

Complete basis function expansion as a tool for machine learning

Jörg Rollbühler

November 22, 2022

Abstract

The expansion of functions into a complete function basis (like, e.g., Fourier series or spherical harmonics) is extensively used in science and engineering. Computer science is no exception, but the technique is rarely mentioned in introductory texts on machine learning. So, a short note describing the basics might be of interest. Expanding densities, regression functions, etc., into complete basis function systems, helps solving supervised and unsupervised tasks. Such expansions are universal in the sense that they can fit (almost) any function. If the series expansion itself converges, then no numerical optimization is required for the fit.

1 Introduction

The original motivation for this work was the need for a very simple and easy-to-implement approach to solve prevalence problems for inhomogeneous data sets with both 'hot spots' of dense data points and other areas with sparse data. For example, earthquakes of large magnitude or high frequency are usually occurring near tectonic plate boundaries, while there are vast areas with much lower activity (by orders of magnitude). This can be an obstacle for a data-driven models of earthquake frequencies and magnitudes.

Another example is the spatial distribution of votes in political elections. In many countries, there are very high densities of votes in urban areas and very low densities of votes in rural areas. But there is much more rural area than urban, such that the total numbers might be comparable. This is crucial, because urban and rural areas usually show different preferences regarding political leanings.

The examples mentioned above are typically regression or classification type problems. However, the first question in approaching such problems is: Where are the data points? I.e., where are high densities and where are low densities. Note that this can quickly become non-obvious in high-dimensional spaces. Once this unsupervised learning task is answered, the focus turns back to classification, regression, etc.¹

A density of observations is needed first, which is an a-priori unknown function from the variable space to the real numbers. The most straightforward approach to express any function in terms of simple known functions seems to be the expansion in a complete basis of functions like Fourier series, spherical harmonics, Bessel functions, etc. This is widely used in various contexts in science. (See, e.g., textbooks of quantum mechanics like [2].) A beautiful example of a application in astrophysics is the expansion of the angular distribution of the cosmic microwave background in spherical harmonics [4, 5]. The most prominent version of complete function basis expansion probably is Fourier

¹In a sense, the problems are mixtures of supervised and unsupervised learning tasks. Usually no labels are provided for the location of hot spots or clusters.

analysis (Fourier transform and Fourier series). There are many numerical techniques available for Fourier analysis and there are no restrictions of the number of dimensions.

Basic machine learning tasks like classification or regression often are built on finding an unknown function of the variables that contains predictive information. This function describes some kind of local mean value. Here, the expansion into a complete function basis comes into play. The function can be approximated by the lowest expansion components, which are usually the slowest varying components. By dropping the higher expansion coefficients, noise is removed. This is the well-known concept of low-pass filtering.

The special case of Fourier expansion mostly appears in the context of machine learning as pre-processing step for image or sound input to deep learning applications. This is not surprising, since Fourier analysis has always been an integral part of image and signal processing. There is some specialized use in machine learning [6], but it is rarely described as a basic machine learning approach. One approach is developed in [7], which is similar to what will be described below for the special case of Fourier analysis.²

I assume that the complete function basis expansion is a widely used method in machine learning. However, in textbooks and tutorials, basis expansion is usually mentioned as using local polynomials, splines, and wavelets (see, e.g., "ESL" [1]). The simpler technique of a global expansion into complete function systems seems not to be part of a typical curriculum. Although these two aspects of 'basis functions' are related to some extent, their relationship might not be obvious. As I could not find literature in the form I needed it for applications, I wrote this short summary.³

I will restrict myself to outlining the method and will demonstrate it using simple toy examples. In the examples, I will only demonstrate Fourier analysis, because this is very generic and has many applications.⁴ But the theoretical framework is given for general complete basis function systems, so replacing the Fourier basis by any other basis is straightforward. Just the basis functions change, not the technique itself, which is universal. The application to actual data sets and the choice of the appropriate complete function system deserve their own treatment.

A data set with many features is equivalent to a set of points in a high-dimensional space. But where in the high-dimensional space are the data points located? How are they distributed?

One way to find this out is to find a smooth density describing the distribution of the data points. One available technique, especially well-developed in low-dimensional spaces, is kernel density estimation. But there is another very simple way: Using complete function basis systems, like those obtained from solving equations of motion (in physics) for a given geometry. We can borrow a lot of such systems from physics and engineering, where plenty of complete sets of orthonormal eigenfunction can be derived. (For the general theory and many of the solutions, we refer to mathematics, of course.) But completeness is more important than orthonormality. The formulas can be generalized for non-orthogonal and non-normalized systems, as we will show later.⁵

Examples of such complete function systems are Fourier series (using orthonormal eigenfunctions of the Laplace operator in Cartesian coordinates in any number of dimensions), spherical harmonics (as eigenfunctions of the angular part of the Laplace operator in spherical coordinates), Bessel functions, Airy functions, the eigenfunction system of the

²In [7], the concept of low-pass filtering is implemented by using a Gaussian filter in Fourier space before back-transformation by discrete Fourier transformation. This provides a smooth cutoff of Fourier components, which is usually a desired feature. This is a powerful technique, but it is a special choice for the approach. Here, we will not apply smoothing of Fourier components and not do discrete Fourier transform, because we want to avoid specialization and want a technique valid for any basis function system.

³Private project, personal opinion.

⁴Again, I would also like to point to [7], which contains some very nice examples, especially in part III.

⁵The projection onto the basis vectors can be done using the inverse of the metric tensor. The metric tensor contains a collection of inner products of the basis vectors.

quantum mechanical harmonic oscillator (in any number of dimensions), the eigenfunction system of the Hydrogen problem (in three dimensions), etc. There is an infinite set of such systems. Many special function systems can be looked up in [3].

Working with densities, as long as the fit can be done reliably, provides a detailed picture of relative probabilities. In classification problems, the focus is often set on achieving the best prediction accuracy. I.e., often a lot of valuable information is ignored by only picking the most likely outcome. Density fits are somewhat like going from black and white to gray scales or even colors.

Similarly, the local distribution shape is often neglected to the point, where tail events can distort mean values. Typically, regression type machine learning algorithms focus on mean values, which minimize the deviation of data points from a local mean value. A wrong distribution assumption can distort this, e.g., in the case of earthquake data or other data with heavy-tailed distributions. Further, a good loss function should also incorporate the local width of the distribution. If the width varies from point to point, a naive squared distance loss function will put wrong weights on the observations. We will discuss how to estimate other local parameters and even allow for general local distribution shapes, which change from point to point.

Like all machine learning algorithms the expansion technique discussed here has its limits. There are hyperparameters, like the number of expansion coefficients taken into account. As I will describe below, keeping only the lowest orders and cutting off 'fast' components corresponds to smoothing. This has to be controlled such that ideally only the noise is smoothed out. Including too many components will result in overfitting. In fact, complete function basis systems can resolve individual data points, in which case the expansions on training, dev, and test sets are completely different. This would be the most extreme version of overfitting. But overfitting can be controlled easily, e.g., by analysis of the spectrum or simply by standard techniques like cross-validation. Especially, with large data sets available nowadays, overfitting is not as limiting as it used to be.

Another limitation is the performance for very high-dimensional datasets, because the number of coefficients can rise quickly with the number of dimensions. This is a matter of performance only. There are a few other limitations. E.g., for Fourier expansions or other wave equation eigenfunctions, there will be wave ripples and the fitted densities might have negative values and meaningless maxima. This can be managed by a smart choice of the threshold levels. Further, depending on the function basis, the approach might not be suited for extrapolation.

2 Theory

The method will now be described in some detail. For simplicity, let's assume we consider a data set with numerical variables. This is not a necessary condition. We can also treat categorical variables, especially ordered ones, e.g., by assigning integer values. In this case we create an artificial spatial structure, and, in the case of unordered variables, we accept artificial ordering. But let us focus on numerical data.

2.1 Density fitting

Suppose we have N observations x_i ($i \in 1, \dots, N$ and $x_i \in \mathbb{R}^d$, where d is the number of variables). We can define an empirical density as

$$\rho_e(x) = \frac{1}{N} \sum_{i=1}^N \delta(x - x_i) . \quad (1)$$

The factor $1/N$ is just for normalization; it is also possible to work with a 'number density' function $n(x) = N\rho_e(x)$, which describes the actually observed numbers in a small volume ΔV via $N_{\Delta V} = \int_{\Delta V} dx n(x)$.

Sometimes, the multiple data points are concentrated in the very same discrete point, but the points themselves are in a continuous space. E.g., the total number of votes n_i in a US county might be mapped to a single coordinate (longitude and latitude) representing a specific geographical point in the county. Then, there are M_i votes all with the same location x_i . We can cover these cases by generalizing

$$\rho_e(x) = \frac{1}{M} \sum_{i=1}^{N'} M_i \delta(x - x_i), \quad M = \sum_{i=1}^{N'} M_i, \quad (2)$$

where N' is the number of locations.

Observations are dots in the space of variables. We can select a space region V within this space, which contains the observations we are interested in. Suppose we have a complete eigenfunction system $\phi_n(x)$ on this volume V . E.g., for flat spaces we can use Fourier eigenfunctions and for geo-data or astronomical data we can use spherical harmonics.

We can expand the empirical distribution into the chosen basis function system,

$$\rho_e(x) = \sum_n c_n \phi_n(x) \quad (3)$$

with expansion coefficients c_n to be determined below. A-priori, the sum is an infinite sum. The key point is to truncate this sum, i.e., restrict it to $n \in \mathcal{I}$, where \mathcal{I} is an appropriate finite index set. By cutting off the fast moving components and keeping only the slowly changing basis functions we can separate off the noise. This yields an approximation of the underlying smooth density, assuming there is one. In Fourier analysis we would say we cut off the high spectral components to create a low-pass filter.

The scale on which the density is smooth is not known a priori. As with many other machine learning methods, this leaves us with unknown hyperparameters. Here it is the index set. There is a lot of freedom to choose it. In many cases it will be conveniently defined by integers, one for each dimension (or even only one number for all dimensions), determining the number of spectral components.⁶ A too low number will not resolve the shape of the distribution enough, but a too high number will overfit the given data set. For oscillatory basis functions (like for Fourier or Bessel functions), a rough rule of thumb is that the smallest 'wavelength' must exceed the average distance of observations in the densest parts.

The expansion coefficients c_n are determined by projections. In case of an orthonormal basis system, the coefficients are obtained by

$$c_n = \int dx \phi_n^*(x) \rho_e(x), \quad (4)$$

i.e., by the scalar product involving the dual space basis vector. This can easily be extended to the non-orthonormal case. We use the scalar product between two functions ϕ and ψ

$$\langle \phi, \psi \rangle := \int dx \phi^*(x) \psi(x) \quad (5)$$

and apply it to the basis function ϕ_m and the function we want to fit $\rho(x) = \sum_n c_n \phi_n$,

$$\langle \phi_m, \rho \rangle = \sum_n c_n \langle \phi_m, \phi_n \rangle = \sum_n g_{mn} c_n. \quad (6)$$

⁶In principle, we can take any selection of components we can fit the data with. E.g., the set can be determined by cross-validation via a grid search.

Here, $g_{mn} := \langle \phi_m, \phi_n \rangle$ is the metric tensor of the basis system. The basis functions are linearly independent, so we can invert the metric tensor and get

$$c_n = \sum_m (g^{-1})_{nm} \langle \phi_m, \rho \rangle = \sum_m (g^{-1})_{nm} \int dx \phi_m^*(x) \rho(x) . \quad (7)$$

This is the generalization of Eq. (4). In practice, the bases will often be orthonormal; whenever the basis function system is chosen as eigenfunction sytème of a Hermitean operator this will be the case. Given orthonormality, $g_{mn} = \delta_{mn}$ (Kronecker symbol), and Eq. (7) will reduce to Eq. (4).

The result of the steps described so far is an approximation for the underlying density

$$\rho(x) \approx \sum_{n \in \mathcal{I}} c_n \phi_n(x) , \quad (8)$$

where (after integration over the δ distributions)

$$c_n = \frac{1}{M} \sum_{i=1}^{N'} M_i \phi_n^*(x_i) . \quad (9)$$

In many cases, the index set will just be defined by $|n| \leq n_{\max}$, keeping only small n values representing the slowly varying components. The fast varying components with $|n| > n_{\max}$ are cut off. The index set (or the cutoff value as hyperparameter, respectively) can be found by cross-validation or similar methods. But we can also interpret $\{|c_n|^2\}_n$ as spectrum, which reflects the importance of the individual components, and use it to find an appropriate cutoff.

Why can this be considered a machine learning algorithm? To give an example, consider a classification problem, where for each vector $x_i \in \mathbb{R}^d$ there is a discrete dependent variable y_i with values in a finite set of classes. If there are N_C categories, let the set be $\{K_m\}$ with $m = 1, \dots, N_C$. Each y_i takes one of the values K_m . For each K_m , we can first determine the expansion coefficients c_n based on the training set and then assign to each test set point $x_j^{(\text{test})}$ the value of the probability density

$$\rho^{(K_m)}(x_j^{(\text{test})}) = \sum_{n \in \mathcal{I}} c_n^{(K_m)} \phi_n(x_j^{(\text{test})}) , \quad (10)$$

and compare the densities for the individual outcomes K_m . A classification algorithm could, e.g., predict the most likely outcome for each test set point. Here, $c_n^{(K_m)}$ has been determined on the training set.

This is very simple, very flexible, and does not need optimization of a cost function.⁷ The method can approximate almost⁸ any shape, at least in principle, and can do so without prior knowledge of the shape. Often, other algorithms require the separation lines to be linear, quadratic, ellipsoids, or similar. Here, we do not need to make any assumptions regarding the shape. There is no need for link functions either. In addition, densities are easily interpretable.

In order to illustrate these last points, let's write the previous function with the coefficients written out (but dropping class labels for simplicity):

$$\rho\left(x_j^{(\text{test})} \middle| x_i^{(\text{train})}\right) = \frac{1}{M} \sum_{i=1}^{N'} \sum_{n \in \mathcal{I}} M_i \phi_n^*(x_i^{(\text{train})}) \phi_n(x_j^{(\text{test})}) . \quad (11)$$

(If $M_i = 1$ for all i , then M and N' both become N .) By using a restricted index set \mathcal{I} to a limited number of coefficients, we obtain a smooth fit to the training set. No other

⁷A cost function only comes in when we quantify the quality of predictions based on densities.

⁸At discontinuities there can arise artifacts, as is well known from Fourier analysis.

adjustment is needed. As this demonstrates, we can immediately apply it to the test set in one go, if we like. This would be ideal for one-shot machine learning problems, where the task is to make an immediate prediction given a newly arriving training and 'test' set for which the prediction is requested. Of course, if we want to fit only once and make predictions for several new data sets, arriving at later times, we will store the coefficients and apply Eq. (10) to new test sets.⁹

The advantages will become self-evident soon, but every approach also has draw-backs. Cutting off the series introduces hyperparameters (often per dimension) and can create artifacts. E.g., eigenfunction systems obtained from wave equations can show ripples, i.e., artificial local optima. This will be visible in the examples shown later. Further, as the scales of data point distances might not be known a priori, the restriction of the index set (or cutoff of an n_{\max}) has to be determined during the analysis. (That said, the volume size and the number of observations allow to determine a useful value a priori.)

Of course, the density can be evaluated anywhere and not just at observations of a test set. The above coarse graining technique smoothes out individual random observations and transforms them into a smooth distribution function. This enables a continuum approximation, which might have potential for machine learning. Why?

Consider a data set that includes temporal information, i.e., some properties which are tracked over time (regularly or randomly). By coarse graining we can derive densities and current densities. This is information about where the points are and in which direction they move.

In order to illustrate this point, let's consider the hypothetical case, where we are given a huge dataset with locations and velocities of all the stars in a region of the Milky Way galaxy. This is a luxury that astrophysicists do not have when they describe other galaxies, so they resort to a use of the collisionless Boltzmann equation. (See, e.g., [8]. It should also be noted that galaxies also contain interstellar matter and dark matter, which would not be part of the hypothetical star table.) This is a continuum description, approaching the limit of very many stars, roughly speaking. The density considered there is the distribution in configuration and velocity space

$$f(x, v, t) = \overline{\sum_{i=1}^N \delta(x - x_i(t)) \delta(v - v_i(t))}, \quad (12)$$

where $x \in \mathbb{R}^3$ is the space vector, $v \in \mathbb{R}^3$ is the velocity, and t is time. The properties with subscript i are location and velocity of the individual data points (stars in a galaxy). The averaging over the fluctuations of the latter, symbolized by the bar, is making f a smooth function over location and velocity space. The collisionless Boltzmann equation is a partial differential equation obtained by taking the time derivative and using equations of motions. (The equations of motions, like Newton's third law, are a luxury that a data scientist usually do not have. The same holds for most conservation laws. However, both astrophysicists and data scientist are in the pleasant situation that the objects of interest rarely scatter. That is the reason for the analogy chosen here.)

The key point is that we can apply the density estimation method described above. The coarse graining by truncating expansion coefficients corresponds to the average symbolized by the bar. Then we get an approximation for $f(x, v, t)$. Or $f(x, v)$ for stationary distributions. We can derive a number density by integrating over velocities and a current density by first multiplying with the velocity and integrating over velocities. Further, it

⁹The immediate 'one-shot' application to the training set without reading out the coefficients might be interesting for (probably far-term) quantum computing as the coefficients can stay concealed. But more interestingly, the transformation is a general 'coarse graining' transformation. It is not restricted to classical density estimation and might be more interesting for the simulation of many-body problems on a quantum computer. A coarse-graining primitive in quantum computing could be interesting (e.g.) for simulating spin arrays or to extract semi-classical approximations. It would have to collect both slow and fast components to stay unitary.

is straightforward to take derivatives, because usually we have differentiable basis functions in an analytical form. The numerical fit is completely encapsulated in the expansion coefficients.

Now imagine any large enough data set containing variables (replacing locations) and the rate of change of the variables (replacing velocities). The rate of change might be measured directly or derived from the locations at two similar but different times. Then we can fit a density and see where the observations are and where they are going. This is different from typical machine learning tasks like clustering, classification, and regression. But let's nevertheless turn to these more typical applications now.

2.2 Classification

2.2.1 Classification using densities

One of the typical supervised learning tasks is classification. Here, the data set contains a categorical dependent variable. There can be multiple classes. Classification is very simple with the basis function expansion. For each category, fit the density or number density. Then use it to make predictions, e.g., by picking the most likely category.

In regions, where densities are comparably large, but similar between the classes, we cannot make reliable predictions, but we can retrieve the maximum amount of information available. E.g., if the probabilities of classes 1-3 are 20%, 50%, and 30%, then we can find this from the densities. We are not able to make a very reliable prediction, because the most likely outcome has only 50% chance of being the correct prediction. In such a scenario, it would be questionable to present one single outcome (like, e.g., the result of a medical test to a patient). But the set of probabilities give a detailed picture of relative probabilities.¹⁰ (In the medical example it can be used to formulate a strategy for suggesting additional tests etc.)

By fitting each class separately, we can determine the unconditional probability densities. If different classes have very different numbers of observation, the estimation errors have different size, but otherwise the densities do not depend on the number of observations. This fact can be used to approach the prevalence problem.

2.2.2 Prevalence

In classification, prevalence is an important topic. The data set might just have more observations for one category than for another. It then depends on what we want to achieve. If the test set can be expected to have the same bias in numbers of observation per class, then a prediction on the test set should be made by a number distribution. If not, i.e., if the prediction is to be made for a balanced data set, then the prediction should be made using the density fit, which yields approximations for the local probabilities of the individual classes. These two cases can also be expressed in terms of unconditional and conditional probabilities.

In a somewhat sloppy notation we can write for the unconditional density in class c in terms of the conditional density $\rho(x|c)$ as

$$\rho_c(x) = \rho(x|c)P_c, \quad (13)$$

where P_c is the probability of an observation belonging to class c . Bayes formula tells us that $\rho(x) = \sum_c \rho_c(x)$.

The prevalence problem can be controlled by estimating the densities for each of the class separately. The number of observations per class only affects the accuracy of each of the fits. But the obtained unconditional densities do not depend on the number of

¹⁰Of course, other machine learning algorithms also provide probabilities, too. Some of them depend on additional assumptions like link functions.

observations in other classes. If the test set is expected to have the same (or similar) P_c , then we can estimate it by

$$P_c \approx N_c^{(\text{train})}/N^{(\text{train})} . \quad (14)$$

If, however, we expect a balanced test set, we use equal probabilities for all classes.

Given an unbalanced data set, we can estimate the number density function $n(x)$, by multiplying the density of class c with the $N^{(\text{test})} \times N_c^{(\text{train})}/N^{(\text{train})}$. But otherwise the densities can be used directly.

2.3 Regression

There are multiple ways to approach regression problems, including stratification of the density fit. But probably the most straightforward and stable way is to just use the classic regression approach, like in linear regression. In that sense it is a well-known approach, except that the basis functions differ from the ones usually used. I will first demonstrate it for Gaussian noise, but will comment on the general case afterwards.

2.3.1 Basic regression approach

The basic assumption is that the data consists of d independent variables x and a (more or less) continuous dependent variable y . (E.g., $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$, but discretized values (due to rounding, e.g.) are also possible.) The y values are distributed as some function f plus noise ϵ ,

$$y = f(x) + \epsilon . \quad (15)$$

Here, we expand the function f into a complete function series, i.e.,

$$f(x) = \sum_n c_n \varphi_n(x) . \quad (16)$$

We only want to fit the smooth function f , not the noise ϵ , but the data does not distinguish between them. So we cut off the series to only fit the smooth part and we assume that this eliminates the noise. In other words, the sum over n is cut off appropriately ($n \in \mathcal{I}$).

The simplest version of regression is to minimize the sum over square distances. I.e., we use the Gauss/Legendre method. Put differently, we can do a log-likelihood fit assuming that the noise is approximately Gaussian, but for now we ignore the width parameter σ , which we will pick up later. We minimize

$$L = \frac{1}{2N} \sum_{i=1}^N \left| y_i - \sum_n c_n \varphi_n(x_i) \right|^2 + \text{const.} , \quad (17)$$

where the observations are labeled by $i = 1, \dots, N$. (We can generalize to multiple observations with the same location in the very same way we did earlier.) The derivative with respect to the complex conjugate coefficient c_n is

$$\frac{\partial L}{\partial c_n^*} = -\frac{1}{N} \sum_{i=1}^N \varphi_n^*(x_i) \left(y_i - \sum_m c_m \varphi_m(x_i) \right) \stackrel{!}{=} 0 , \quad (18)$$

With the abbreviations

$$\begin{aligned} \tilde{y}_n &:= \frac{1}{N} \sum_{i=1}^N \varphi_n^*(x_i) y_i , \\ M_{nm} &:= \frac{1}{N} \sum_{i=1}^N \varphi_n^*(x_i) \varphi_m(x_i) , \end{aligned} \quad (19)$$

this can be expressed as the simple linear equation

$$\tilde{y}_n = \sum_m M_{nm} c_m . \quad (20)$$

If the number of observations is large enough (compared to the number of basis function coefficients), then the matrix is invertible. But note that this can become a practical issue if a small dataset is fitted using a large number of components.¹¹ However, to have a smoothing effect, we keep the number of components small compared to number of observables, so everything should be fine for machine learning applications. The desired basis function coefficients are

$$c_n = \sum_m (M^{-1})_{nm} \tilde{y}_m . \quad (21)$$

Again, making the index set finite, often by using cutoffs of the component indices, can be viewed as hyperparameter selection. Usual techniques like the analysis of the spectrum or cross-validation can be applied to determine an optimal value.

2.3.2 Estimation of the distribution width

This can be generalized for the case, where also the distribution width changes. In fact, if we have a parametric distribution with K parameters $\theta = (\theta_1, \dots, \theta_K)$, we can estimate all these parameters. But for now let's restrict ourselves to just estimating a width parameter. Consider, e.g., the conditional probability distribution for Gaussian noise¹²

$$p(y|x) = \frac{1}{\sqrt{2\pi\sigma^2(x)}} e^{-\frac{(y-\mu(x))^2}{2\sigma^2(x)}} . \quad (22)$$

The log-likelihood function then is

$$L = \frac{1}{N} \sum_{i=1}^N \ln p(y_i|x_i) = \frac{1}{N} \sum_{i=1}^N \left\{ -\frac{|y_i - \mu(x_i)|^2}{2\sigma^2(x_i)} - \frac{1}{2} \ln(2\pi\sigma^2(x_i)) \right\} . \quad (23)$$

We expand

$$\begin{aligned} \mu(x) &= \sum_n a_n \phi_n(x) , \\ \sigma^2(x) &= \sum_n b_n \phi_n(x) . \end{aligned} \quad (24)$$

Note that we can alternatively expand σ as $\sigma(x) = \sum_n c_n \phi_n(x)$ instead of expanding σ^2 . Other approaches are possible.¹³ When using complex basis functions, squares have to be converted into absolute squares, where applicable. We evaluate the log-likelihood function

¹¹E.g., for Fourier series, the number of Fourier components must not exceed the number of linearly independent observations. Duplicates have to be taken into account when counting. If the dataset is not too large, checking for numerically small eigenvalues is an easy method to check, provided it can be performed quickly for the given data set.

¹²To avoid an overloaded notation, we suppress labels showing that this is conditional on the parameters.

¹³E.g., an equivalent ansatz is

$$p(y|x) = \sqrt{\frac{A(x)}{2\pi}} e^{-\frac{(A(x)y+B(x))^2}{2}} , \quad (25)$$

but the normalization factor will always send one function in the denominator, when taking the derivative of the log-likelihood function.

at the observations i and take the derivatives with respect to a_n^* and b_n^* . This results in a system of equations,

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^N \frac{y_i \varphi_n^*(x_i)}{\sigma^2(x_i)} &= \frac{1}{N} \sum_{i=1}^N \frac{\mu(x_i) \varphi_n^*(x_i)}{\sigma^2(x_i)}, \\ \frac{1}{N} \sum_{i=1}^N \frac{(y_i - \mu(x_i))^2 \varphi_n^*(x_i)}{|\sigma^2(x_i)|^2} &= \frac{1}{N} \sum_{i=1}^N \frac{\sigma^2(x_i) \varphi_n^*(x_i)}{|\sigma^2(x_i)|^2}. \end{aligned} \quad (26)$$

The denominators essentially give more weight to the data with smaller variance.¹⁴ These denominators are functions evaluated at the observations, such that they cannot be pulled out in front of the sum and canceled. With the expansion coefficients appearing in the numerator and denominator, this is not a linear systems of equations anymore. This just means that we cannot simply solve it by matrix inversion as in the linear case. But it can be solved numerically (as long as such a solution exist). In many practical problems, we can assume that σ is finite, such that the weights cannot become exceedingly large.

Note that for spatially constant σ , but space-dependent $\mu(x)$, we can just pull out and cancel the denominators on both sides, recovering the previous set of equations,

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^N y_i \varphi_n^*(x_i) &= \frac{1}{N} \sum_{i=1}^N \mu(x_i) \varphi_n^*(x_i), \\ \frac{1}{N} \sum_{i=1}^N (y_i - \mu(x_i))^2 \varphi_n^*(x_i) &= \frac{1}{N} \sum_{i=1}^N \sigma^2 \varphi_n^*(x_i). \end{aligned} \quad (28)$$

The first equation is the one we solved previously by matrix inversion. As long as $\sigma(x)$ is very close to being constant, i.e., just showing small (and ideally smooth) deviations from a globally constant function, we can work with the approximation

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^N y_i \varphi_n^*(x_i) &\approx \frac{1}{N} \sum_{i=1}^N \mu(x_i) \varphi_n^*(x_i), \\ \frac{1}{N} \sum_{i=1}^N (y_i - \mu(x_i))^2 \varphi_n^*(x_i) &\approx \frac{1}{N} \sum_{i=1}^N \sigma^2(x_i) \varphi_n^*(x_i). \end{aligned} \quad (29)$$

This works as approximation, because the main information about the width is contained in the 'residuals' $y_i - \mu(x_i)$. The denominators in the general formula just give different weights areas with smaller width.

The exact solution can be found numerically, e.g., by using optimization or, more easily, by an iterative approach. For the latter, I suggest the following:

- Starting point: First solve Eq. (28) for constant σ or the approximative system Eq. (29). Use these as starting point for the iteration.
- Repeatedly solve the set of linear equations

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^N \frac{y_i \varphi_n^*(x_i)}{\sigma_{\text{prev}}^2(x_i)} &= \frac{1}{N} \sum_{i=1}^N \frac{\mu(x_i) \varphi_n^*(x_i)}{\sigma_{\text{prev}}^2(x_i)}, \\ \frac{1}{N} \sum_{i=1}^N \frac{(y_i - \mu_{\text{prev}}(x_i))^2 \varphi_n^*(x_i)}{|\sigma_{\text{prev}}^2(x_i)|^2} &= \frac{1}{N} \sum_{i=1}^N \frac{\sigma^2(x_i) \varphi_n^*(x_i)}{|\sigma_{\text{prev}}^2(x_i)|^2}, \end{aligned} \quad (30)$$

¹⁴The denominators slightly depend on the explicit choice of expansion. With the alternative expansion $\sigma(x) = \sum_n c_n \phi_n(x)$, the second equation becomes

$$\frac{1}{N} \sum_{i=1}^N \frac{(y_i - \mu(x_i))^2 \varphi_n^*(x_i)}{|\sigma(x_i)|^3} = \frac{1}{N} \sum_{i=1}^N \frac{\sigma(x_i) \varphi_n^*(x_i)}{|\sigma(x_i)|^2}. \quad (27)$$

where "prev" denotes the solution of the previous iteration. The function $\mu_{\text{prev}}(x_i)$ can either be the previous $\mu(x_i)$ or the one already updated using the first equation (which is the preferred version). But in order to keep the second equation linear, we do not use the coefficient we are solving for in the second equation. Below, I will assume that in each iteration we first solve the first equation and then use the μ obtained as given in the second equation. The latter we solve for σ^2 .

- Either set the maximum number of iterations as a hyperparameter upfront or iterate until a reasonable stopping criterion has been reached.

Some experiments showed that the iteration approach worked, but in many cases the initial fit using Eq. (28) was completely sufficient. The more advanced approach of Eq. (26) or Eq. (30) brought an improvement in the example shown in Fig. 4. I will not further discuss convergence of the iteration here, but it seems like a fairly straightforward problem, because it can be seen as a slow adjustment (and hopefully improvement) of weight factors.

In all the linearized cases above we can write

$$\begin{aligned}\tilde{y}_n &= \sum_m M_{nm} a_m, \\ \tilde{r}_n &= \sum_m K_{nm} b_m.\end{aligned}\tag{31}$$

E.g., for the iteration approach, we have

$$\begin{aligned}\tilde{y}_n &:= \frac{1}{N} \sum_{i=1}^N \frac{y_i \varphi_n^*(x_i)}{\sigma_{\text{prev}}^2(x_i)}, \\ M_{nm} &:= \frac{1}{N} \sum_{i=1}^N \frac{\varphi_n^*(x_i) \varphi_m(x_i)}{\sigma_{\text{prev}}^2(x_i)}\end{aligned}\tag{32}$$

and

$$\begin{aligned}\tilde{r}_n &:= \frac{1}{N} \sum_{i=1}^N \frac{(y_i - \mu_{\text{prev}}(x_i))^2 \varphi_n^*(x_i)}{[\sigma_{\text{prev}}^2(x_i)]^2}, \\ K_{nm} &:= \frac{1}{N} \sum_{i=1}^N \frac{\varphi_n^*(x_i) \varphi_m(x_i)}{[\sigma_{\text{prev}}^2(x_i)]^2}\end{aligned}\tag{33}$$

The solution per iteration step is

$$\begin{aligned}a_n &= \sum_m (M^{-1})_{nm} \tilde{y}_m, \\ b_n &= \sum_m (K^{-1})_{nm} \tilde{r}_m.\end{aligned}\tag{34}$$

For the other linear approximations (constant σ or constant σ in denominator), one can just cancel the denominators.

Finally, we note that the log-likelihood estimators are usually not (necessarily) unbiased. One can consider adjusting the variance estimator, but I will not further go into detail here.

2.3.3 General parametric distributions

As mentioned earlier, we can extend this to a parametric distribution with K parameters $\theta(x) = (\theta_1(x), \dots, \theta_K(x))$. Then

$$L = \ln p(y|\theta(x))\tag{35}$$

Each of the parameters can be expanded

$$\theta_k(x) = \sum_n a_{k,n} \phi_n(x) . \quad (36)$$

The log-likelihood approach provides a system of equations

$$\nabla_{a_{k,n}^*} L = 0 \quad (37)$$

which determines the parameters.

2.3.4 Collapsing local distributions

The distribution mean and variance vary from point to point. If the randomness is approximately normal, we can use the estimates obtained above and consider the 'standardized residuals'

$$s_i := \frac{y_i - \mu(x_i)}{\sigma(x_i)} . \quad (38)$$

Of course, if the noise is subject to a different distribution, the estimators and the residuals have to be adjusted. E.g., for an exponential distribution we would just have one parameter. The general approach will still work, of course, as the basis functions just fit the parameter dependence on the observables, whatever the parameter is.

We can then plot or fit the empirical distribution and look whether it is consistent with the assumptions. This is an opportunity to check whether there are outliers and whether the distribution assumptions are justified.

More formally, we should check the overall reduction in variance. We can calculate an R^2 , where we compare the variance of the residual width under the regression model with the variance of the baseline model, in which μ and σ are constants. Usually it is assumed that the mean and variance estimates are not sensitive to the exact distribution as long as the distribution is not heavy-tailed. But there are many practically relevant cases (like, e.g., earthquake statistics), where the mean and variance estimators become unreliable. In the most extreme case the moments are not even defined anymore. But even if the moments exist, the estimation errors can become very large. Then the appropriate distribution shape has to be found and in case of a parametric distribution the log-likelihood estimators have to be derived. From this follows the appropriate residual definition.

However, it is not always practical to use a parametric distribution, e.g., if the distribution is fitted point by point. Then we can still find location and width estimators and check the collapsed data points against the fitted distributions. One way to do so is by applying the fitted (empirical) cumulative distribution function and check against a uniform distribution [9] or other distributions.

2.3.5 Main features extraction

Linear regression gives a direct answer to the question, which feature drives the independent variable by how much. The statistical relevance can be determined. We can do the latter for the complete basis expansion as well, no matter how complicated or non-linear the fit function is. But what about the linear coefficients that linear regression with a linear fitting function provides? As the basis functions (usually) are differentiable, we can determine a local approximation of the global fit by taking the gradient. This yields local linear coefficients, which can be used for a linear approximation in a vicinity of a point of interest.

2.4 Unsupervised learning

The density fit allows finding the hot spots where many data points cluster together. One way of clustering is to first find local maxima of the fitted density function. By finding saddle points and comparing the function values with the maxima, it can be determined if these maxima are connected or not. Doing so it is possible to distinguish extended structures like "ridges" from elliptical blobs. Note that this can be done without defining a distance measure. In some cases, it might nevertheless be desirable to take a distance measure into account.

This method should only be used where the densities at the maxima are high. If they are not, a maximum search will find all the meaningless maxima of artificial wave ripples. Even if densities are high, a "ridge" may turn into a sequence of maxima and saddle points. It is possible to change the number of components or the fitting region to identify such artifacts.

In many cases, this way of clustering will be much less convenient than standard approaches like k-means. However, there are important advantages to do clustering via the basis function density fit. There is no need to introduce a distance measure between data points. In k-means a scale change in one or multiple variables can distort the density and change the cluster assignment. Further, there is no need to search through the whole set of data points repeatedly.¹⁵

3 Example: Fourier analysis

Fourier analysis has been used in information technology for ages, especially in signal processing. In machine learning, it is used for pre-processing image and sound inputs, e.g., in speech recognition. However, it seems to be missing in many texts as a basic method for density estimation, classification, regression, and clustering as described here. There is one set of three papers [7], which describes the use of Fourier transform in machine learning. The author suggests an algorithm using a smooth cutoff of the Fourier components followed by discrete Fourier transform, and then provides very convincing examples, especially for classification. I will describe a more straightforward approach without smooth cutoff and without discrete Fourier transform. This is very easy to implement and works the same way for other basis function systems.

3.1 The Fourier basis functions

Fourier analysis is a special case of the general method described above, which is best suited to explain the basic concept. The eigenfunctions in this case are¹⁶

$$\phi_{\mathbf{n}}(\mathbf{x}) = \frac{1}{N} e^{i\mathbf{k}_{\mathbf{n}} \cdot \mathbf{x}} \quad (39)$$

with a normalization factor N and wave vector \mathbf{k} with components

$$k_{i,n_i} = \frac{2\pi n_i}{m_i L_i} . \quad (40)$$

The volume is rectangular and L_i is the side length in direction i . The integers m_i determine on how many multiples of L_i the function shall be periodic. This simplifies

¹⁵A similar statement can be made about classification or regression described above versus k-nearest neighbors. The complete basis expansion method here, neither needs a distance measure nor is it necessary to search for the nearest neighbor observations. Otherwise the methods are similar in the sense that the predictions is derived from neighboring points.

¹⁶I present the complex Fourier expansion, because it allows for a much more compact code than the real representation with sine and cosine functions.

the treatment if the data comes close to the volume boundary. (This may happen when zooming in into a region of interest.) With $m_i = 1$, we implicitly make the function periodic with period L_i in direction i . If then the values at the two borders of the interval are not the same, then there is a discontinuity, i.e., the fit can suffer. If the data to be fitted is not periodic on the volume, one should pick $m_i = 2$ (or possibly larger) in dimensions i to which the non-periodicity applies.¹⁷ Optionally, we can make the function periodic and continuous by adding a mirror image. Then we can prevent discontinuities. But this is not necessary for the method to work. Especially, if the density falls off to zero near the boundaries it is not needed at all.

The normalization condition is

$$\int d^d x \phi_{\mathbf{n}}^*(\mathbf{x}) \phi_{\mathbf{m}}(\mathbf{x}) = \delta_{nm} . \quad (41)$$

For Fourier series on a finite volume $V = \prod_i L_i m_i$, this translates into

$$\frac{1}{N^2} \int_V d^d x e^{-i\mathbf{k}_n \cdot \mathbf{x}} e^{i\mathbf{k}_m \cdot \mathbf{x}} = \frac{V}{N^2} \delta_{nm} , \quad (42)$$

such that the normalization factor can be chosen as $1/\sqrt{V}$. However, often the Fourier series is defined as

$$f(x) = \sum_n c_n e^{ik_n x} , \quad (43)$$

i.e., without an extra normalization factor. Then the back-transformation has to carry an extra factor of $1/\sqrt{V}$:

$$c_n = \frac{1}{V} \int d^d x f(x) e^{-ik_n x} . \quad (44)$$

We can use this function basis in our density estimation approach, i.e., the coefficients become

$$c_n = \frac{1}{VM} \sum_{i=1}^N M_i e^{-ik_n x_i} \quad (45)$$

for $n \in \mathcal{I}$ (\mathcal{I} is a finite index set) and the evaluation on a test set is

$$\rho \left(x_j^{(\text{test})} \right) = \sum_{n \in \mathcal{I}} c_n e^{ik_n x_j^{(\text{test})}} . \quad (46)$$

Note that the observations are just put in without prior preprocessing. In many applications of Fourier analysis with data, one would first bin the data and then (if the bins have equal size) apply a discrete Fourier transform to it. However, this step is not needed here, which simplifies the implementation considerably.¹⁸

3.2 Illustrative example of a 1D density fitting and classification problem

For illustration purposes I will first demonstrate the method on an artificial data set in one dimension. The training set consists of 10000 simulations in total, with normal densities

¹⁷Note that the average value over the original volume is the respective zero component for $m_i = 1$, but for $m_i = 2$ both the zero and the first component contribute to the average value. This subtle point, however, stays in the background and it is not causing trouble at all.

¹⁸On the other hand, there are algorithms readily available for discrete Fourier transformation. In principle, this even is true for quantum computing ('quantum Fourier transform' or QFT). Here, the approach without binning would require phases to be tunable very accurately, which seems out of range at the moment.

located at (μ) -1, 2, 3, 6, 8 with width (σ) 2, 1, 1, 3, 2 and with relative weights of 4:2:3:2:5. The true density will be plotted as red curves in the graphs below.

This is evaluated on an artificial test set of x -values normally distributed around $\mu = 3$ with width $\sigma = 5$, just to make an arbitrary choice. The 'predictions' on the test set will be shown as blue points in the graphs below. Fig. 1 shows the results of fits with varying number of Fourier components. The upper left graph shows the fit for Fourier components

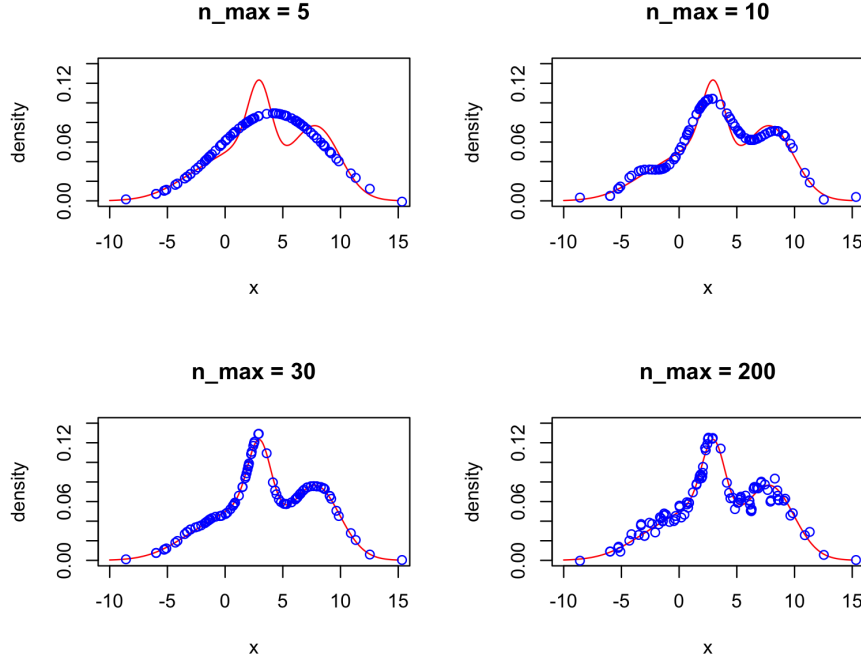


Figure 1: Density fits with different number of Fourier coefficients.

from -5 to 5. This is already showing where most of the data points are located. With -10 to 10 (upper right plot), the fit resolves the detailed structure better, but on the other hand adds more artificial oscillations at first. However, increasing the number further leads to the almost perfect fit for -30 to 30 (lower left figure). For high numbers of Fourier components we expect overfitting. The density fit will then start to resolve the random distances between sample points. Indeed, at about the range of -50 to 50, this becomes visible by short range density fluctuations. But to make it really well visible the lower right plot shows it for -200 to 200.

Note, however, that overfitting is a problem that can be very well controlled, e.g., by n -fold cross validation or similar techniques. It is also helpful to plot the absolute square values of the Fourier components. In the example given, this shows a peak centered at $n = 0$, decaying quickly. At about $n = 20$ to $n = 30$ this reaches a noise level. This is another way of finding the region in which signal is stronger than the noise.

3.3 Illustrative example of a 1D regression problem

As an example, look at the arbitrarily chosen function

$$f(x) = 3 + 0.25x + \sin(x) + 0.5 \sin(2x + 2) + 0.3 \sin(3x - 1) . \quad (47)$$

This function is evaluated on a random set of x -values. For example, take

$$Y = f(X) + \epsilon , \quad X \sim N(0, 1.8) , \quad \epsilon \sim N(0, 0.3) \quad (48)$$

and let the number of observations be 200. Applying the regression to this random training set provides the Fourier components. For example, let the index run between -5

and 5. The corresponding Fourier series can be used to make predictions on a test set, e.g. $X_{\text{test}} \sim N(0, 1.8)$, also 200 points.

3.3.1 Simple regression

In Fig. 2, the training set is plotted as black dots, the predictions as blue points, and the actual function as red line. The 'predictions' points are lined up along the red line as

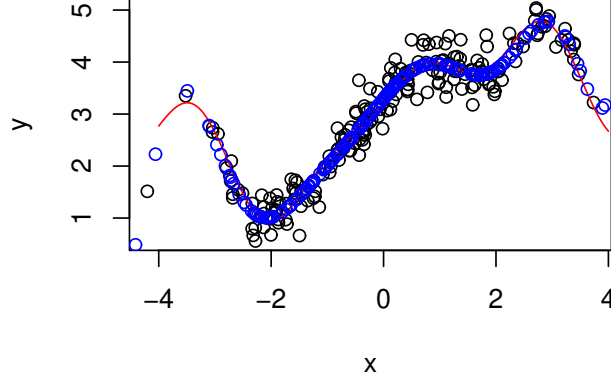


Figure 2: 1D regression example

desired. Of course, the regression piece is the classical method, except that we are using complex basis functions. But even with only 200 data points, the distances between data points is not resolved with the 11 Fourier components chosen here. I.e., overfitting does not set in too early.

The next step is to include the volatility estimate. For example, let σ be a space-dependent function

$$g(x) = 0.1 \text{ abs}(3 + 0.2x - 0.25 * x^2 + 0.7 \sin(2x + 2)) + 0.2 . \quad (49)$$

which is just another arbitrary choice. The regression fit is shown in Fig. 3 for Fourier components from -8 to 8 . It was obtained using the iterative approach with 10 iterations; but an almost undistinguishable plot can be achieved using the approximation Eq. (29). The green line shows the theoretical functions for $\mu(x)$ plus and minus one 'standard deviation' $\sigma(x)$. The blue dots show the fit for $\mu(x)$, the red dots show the fits for $\sigma(x)$ in terms of $\mu(x) \pm \sigma(x)$.

3.3.2 Distribution collapsing

The previous example might not seem impressive yet, because the local standard deviation does not deviate too much from the 'average' standard deviation of the baseline model.

In order to see the emphasize the effect and to illustrate distribution collapsing, let's look at the function

$$f(x) = 3 + 0.125x + \sin(x) + 0.15 * \sin(2 * x + 2) + 0.13 * \sin(3x - 1) \quad (50)$$

for the local mean values and add noise with a strongly varying σ parameter. To make it extreme, let $\sigma(x) = g(x)$

$$g(x) = 0.3 + 2.0 * \rho_{\text{norm}}(x, -2, 0.5) + 1.0 * \rho_{\text{norm}}(x, 1.5, .5) \quad (51)$$

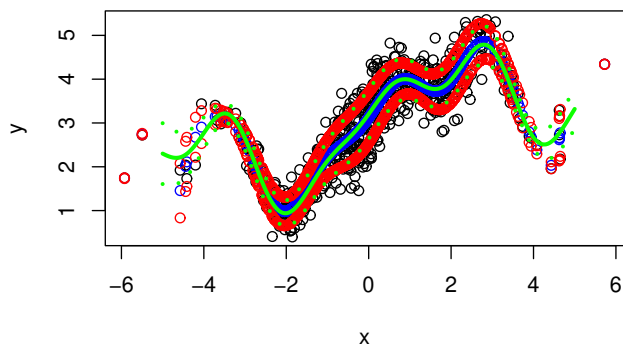


Figure 3: 1D regression example with variance fit

with two pronounced peaks at $x = -2$ and $x = 1.5$. Here, $\rho_{\text{norm}}(x, \mu, \sigma)$ is the Gaussian density function centered at μ with width σ . This example is shown in Fig. 4. In this example, the variance locally deviates considerably from the overall average value (i.e., the variance measured over all data points with equal weights). This distorts the unconditional distribution.

3.3.3 Using density fits to improve regression

A density fit can help identifying low data point densities and to identify data points for which the regression is expected to be inaccurate. I will not demonstrate this here, but mention it as an additional tool that might come in handy.

3.4 Illustrative example of a 2D density fitting and classification problem

Usually, a machine learning problem, even with structured data, is a multi-dimensional problem. I would like to illustrate the density fitting technique in two dimensions, where everything can still be plotted nicely. I create a data set with three classification classes. Two of the classes are "smiley faces" and the third class is just noise. The data points are plotted in Fig. 5 with different colors for the different classes.

The structure of the data is clear immediately, because we work in two dimensions and the shapes are chosen for that purpose. But note how projections on x and y are not straightforward anymore. (In higher dimensions, we mostly work with projections during data inspection.) This is illustrated in Fig. 6, which shows a kernel density plot of the x -values only. Viewed from this angle, this is a very tough classification problem, because the densities of x and y are very similar. At the values -5 and 5 the two classes are not distinguishable. The Fourier method will not fix that problem, but using it for estimating densities makes it very clear, where classification is possible and where it is not. In a situation like the one shown above, predicting a class is not very meaningful. But having an estimate of the probability densities is valuable, because it gives the relative probabilities of the classes. This allows judging where a prediction is useless and where the odds differ enough make classification possible.

3.4.1 Fitting approach

Let's turn to the actual fitting method. Starting with a small number of Fourier components, e.g., from -4 to 4 , helps to locate the data and get a first impression, see Fig. 7.

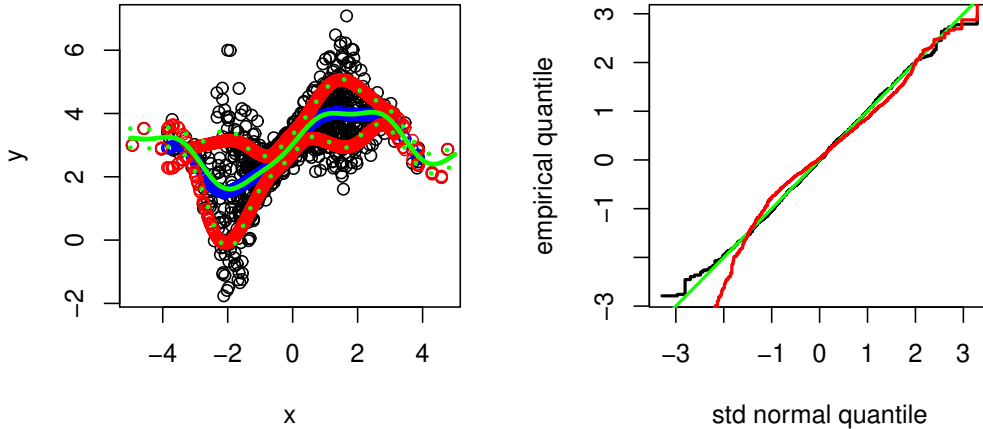


Figure 4: Example with two local variance peaks at $x = -2$ and $x = 1.5$. The left graph shows the fit of mean (the solid green line fits the blue points) and width (The dashed green lines fit the red dots). The graph on the right shows a QQ plot. The local standardization (black curve) lies closer to the standard normal distribution (green curve) than the baseline model (red curve). Note that the color coding in the two graphs is not related to each other.

The density plots for classes '1' and '2' indicate some circular structure.

The more detailed structure becomes apparent, when we go to higher resolution. Fig. 8 shows this for Fourier coefficients from -25 to 25. It is important to note that the prevalence problem of the unbalanced data set is not disturbing much. It is only that the density fit is less accurate if the number of data points is smaller. But each class can be fitted separately. This is unlike the situation in (e.g.) logistic regression.

One very nice feature of the Fourier method is that one can easily zoom in and investigate a region of interest. Here, it makes sense to zoom in onto the region of high density. Looking at the data where x and y are between -7 and 7 , a much smaller set of Fourier components suffices. Fig. 9 shows the case with coefficient labels from -7 to 7 . Again, the resolution can be improved if desired, as shown in Fig. 10 with coefficient indices from -15 to 15 . Note that the noise part already becomes overfitted. Of course, for each class an appropriate resolution should be chosen independently.

3.4.2 A variation: subtraction method for unlabeled data

The data set above was labeled and the smiley faces are already visible after data stratification. Let's now assume the classes are not labeled. For simplicity let there be only one smiley and lots of noise, i.e., we just remove the second smiley data points for this example. Now, without labels, the smiley is completely hidden behind the noise. By Fourier fitting the density with a very small number of components, we will first just see a blob. But we can subtract the initial fit and then go to higher resolution (as often done in astrophysics). Then we start seeing the structure superimposed on the Gaussian blob. Why does it work? We are using the fact that the length scales of structures are very different for the blob and for the smiley. The expansion allows for a separation of length scales and we can recover the smiley also in an unlabeled data set. (The explicit results look visually similar to the ones shown for labeled data.) The second smiley has comparable length scales than the first one. If we add it on top, it cannot be easily separated from the first one, unless it is added with different size.

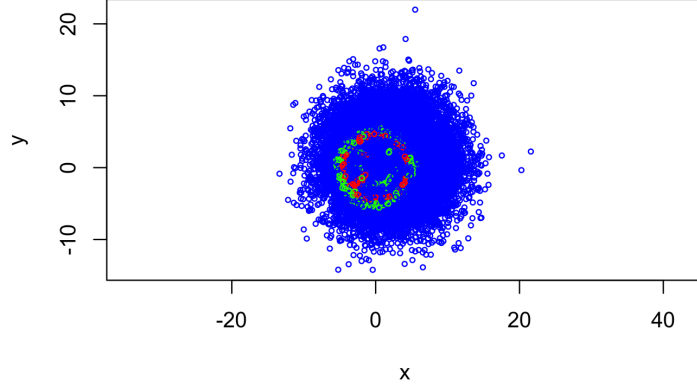


Figure 5: 2D data set with three classification classes

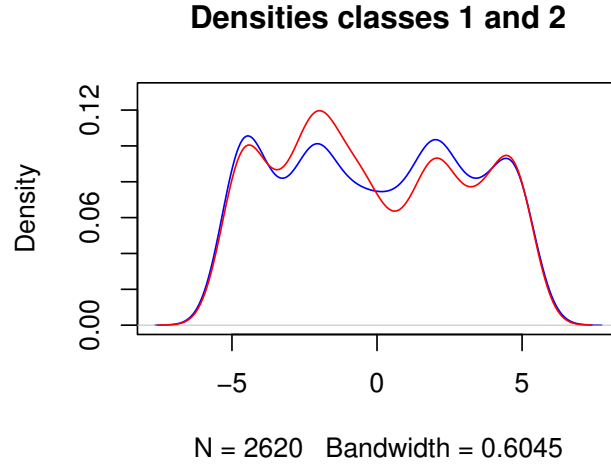


Figure 6: Kernel density after projection onto the x-axis.

References

- [1] T. Hastie, R. Tibhsirani, J. Friedman, *The Elements of Statistical Learning*, Springer, 2nd edition, 2008.
- [2] J. J. Sakurai, *Modern quantum mechanics*, Addison-Wesley, rev. ed., 1994.
- [3] M. Abramowitz, I. A. Stegun, *Handbook of Mathematical Functions*, Dover, ninth Dover printing, 1972.
- [4] C. L. Bennett, D. Larson, J. L. Weiland, et al., *ApJS*, 208, 20 (2013).
- [5] Planck Collaboration, *A&A* 641, A1 (2020).
- [6] N. Linial, Y. Mansour, and N. Nisan, *Journal of the Association for Computing Machinery*, Vol 40, No 3, July 1993.
- [7] S. Mehrabkhani, arxiv:1904.00368v3, arXiv:1904.13241v3, arXiv:2001.06081v2.
- [8] S. Weinberg, *Lectures on Astrophysics*, Cambridge University Press, 2020.
- [9] M. Rosenblatt, *The Annals of Mathematical Statistic* 23, No. 3, 470-472 (1952).

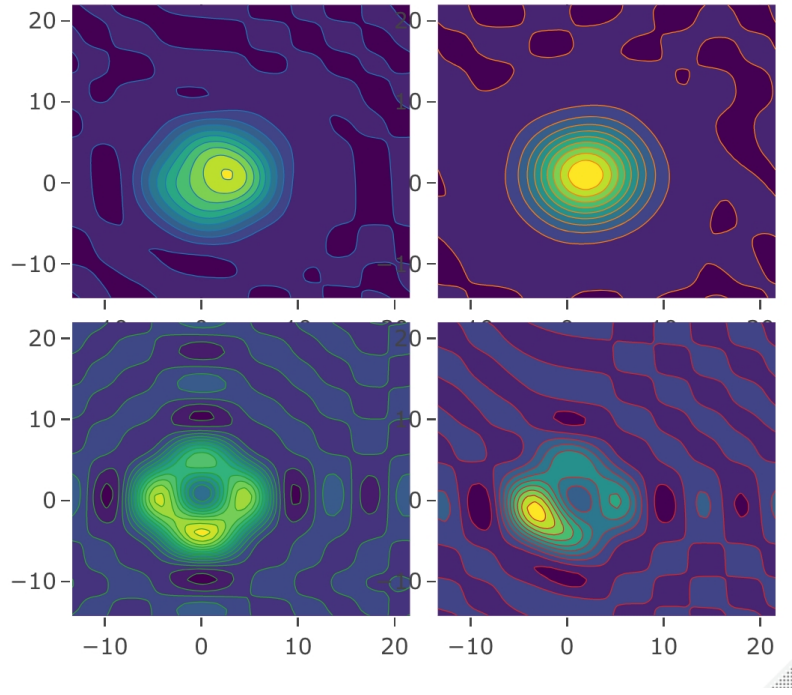


Figure 7: Fourier density fit with low resolution. Top left to bottom right: all data, class '0', '1', '2'.

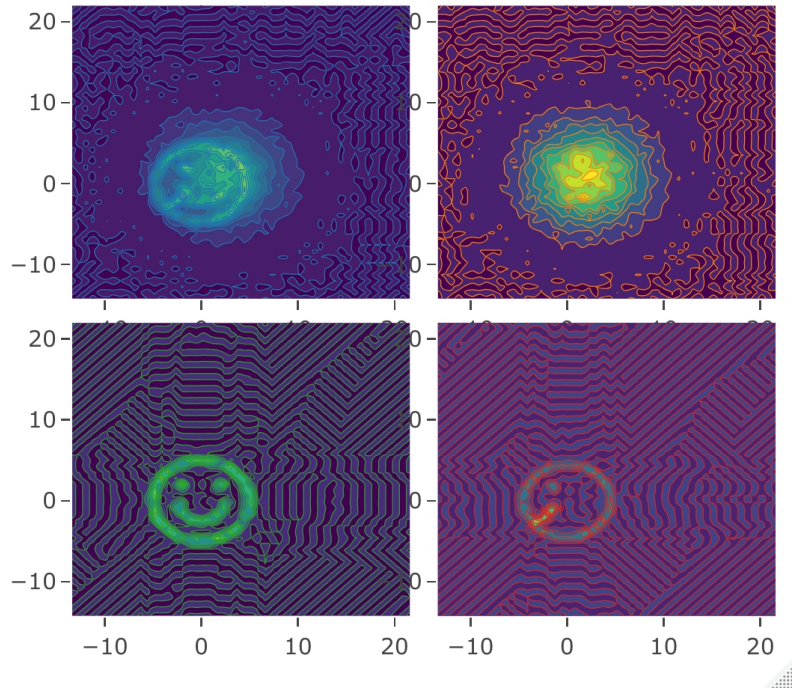


Figure 8: Fourier density fit with high resolution. Graph order as before.

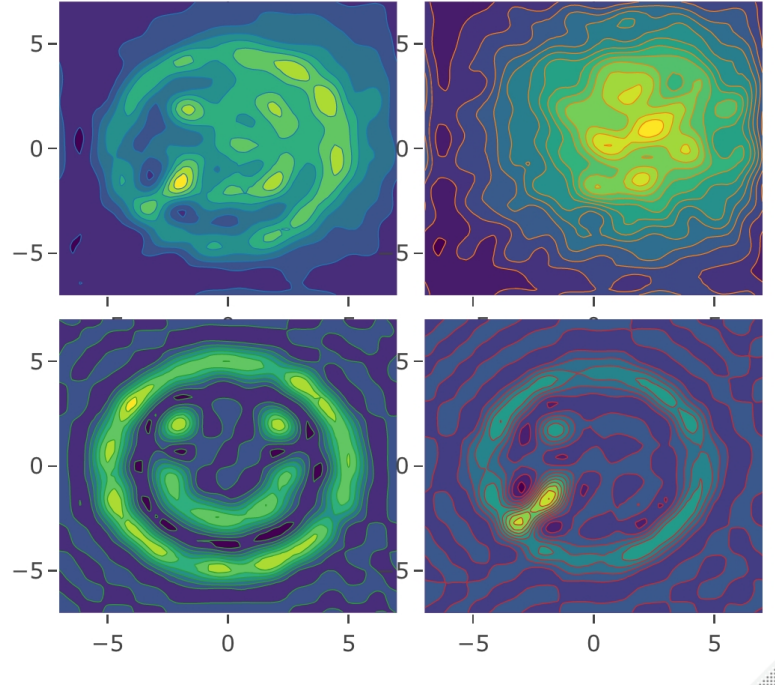


Figure 9: Fourier density fit after zooming in.

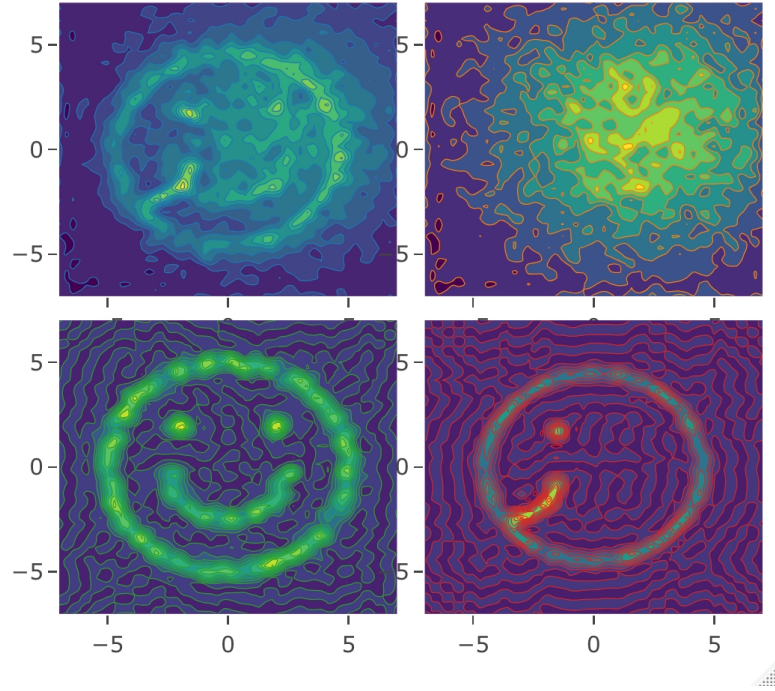


Figure 10: Fourier density fit after zooming in, high resolution.