Exercises for
Pattern Analysis Programming
Lina Felsner, Mathias Seuret, Dalia Rodriguez-Salas
Task 1: April 21 – April 29

Pattern Recognition Lab

# Warmup: Sampling from a Closed-Form Distribution

Our goal in Pattern Analysis is to analyze data. However, let us first look into ways to *create* data, that we can represent then in later steps. Hence, this exercise is about drawing sample data from a closed-form probability density function (PDF). This process is oftentimes just called "sampling".

Sampling has different applications. In the fields of computer graphics and computer vision, a currently popular example is to invent visual data like human faces. A nice demo for artificial faces is at `http://thispersondoesnotexist.com`. A cool drive through their distribution for sampling (the "latent space") is shown in `https://www.youtube.com/watch?v=6E1_dgYlifc`.

This exercise is intended as a warmup. It does not involve a lot of code, but it is an opportunity to setup your exercise environment:

(a) Make sure that your python environment works

(b) Get familiar with the type of problem statement in this exercise

(c) Set up your communication logistics, establish contact to your group members

**Exercise 1** Implement a function that draws 1000 samples from an univariant gaussian distribution with mean 1 and a standard deviation of 0.2. Plot the experimental distribution using 30 bins. In order to compare it with the ground truth distribution, generate also the plot of such. An example is shown in Fig. 1

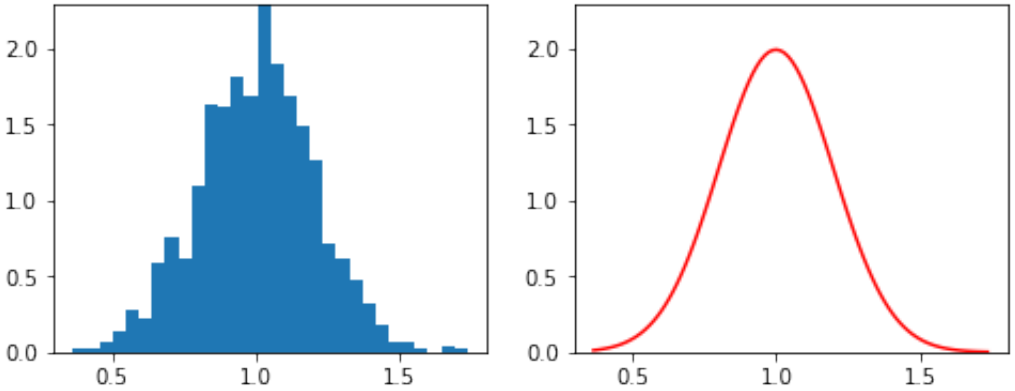


Figure 1: Experimental and Ground Truth Distribution in 1D.

**Exercise 2** Implement a function that draws 10000 samples from a multivariant gaussian distribution (2D in this case) with mean mean vector $[0.5, -0.2]$ and covariance matrix $[[2.0, 0.3], [0.3, 0.5]]$. Plot the experimental distribution using 30 bins (30x30). In order to compare it with the ground truth distribution, generate also the plot of such. An example is shown in Fig. 2

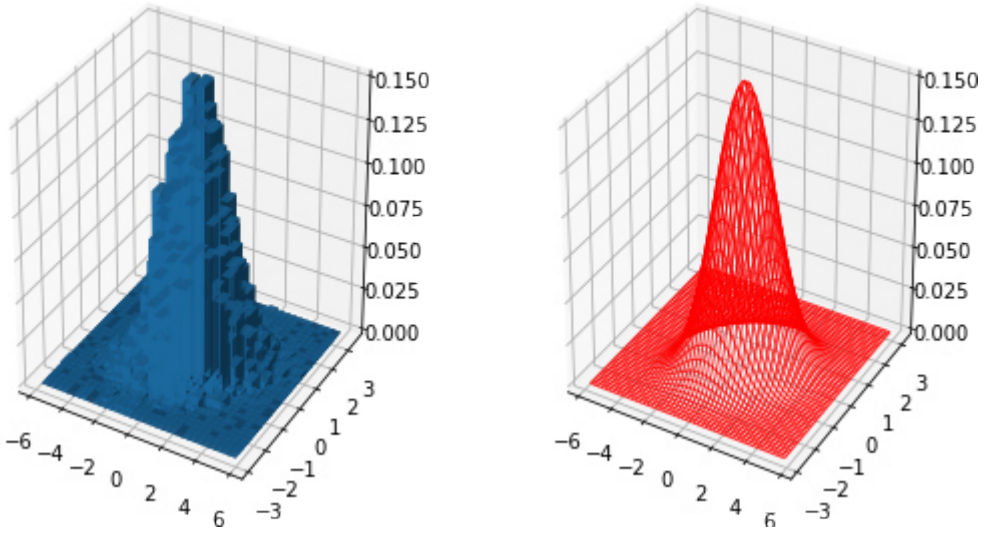


Figure 2: Experimental and Ground Truth Distribution in 2D.

**Exercise 3** Play with the amount of drawn samples and of the bins. What happens when you increase/decrease both? What happens when you increase one and decrease the other? Are observations from these past two questions the same for the 1D and 2D cases? What about changing the parameters of the ground truth distributions? Observe the plots and select the one which turned the most interesting for you; Please create a figure with the respective 1D and 2D distributions and post it on the forum (Thread: 'Warmup') and comment why you chose it.

*Comments:*
*We ask for only one figure per group. Please also state your group number. Bring your code to the joint meeting on April 27 or April 28 for a potential little extra experiment.*

Exercises for
Pattern Analysis Programming
Lina Felsner, Mathias Seuret, Dalia Rodriguez-Salas
Task 2: April 29 – May 6

Pattern Recognition Lab

# Density Estimation

**Our First Empirical Distribution: Sampled from a Non-parametric Ground Truth Distribution**

A central challenge in statistical machine learning is the discrepancy between a true, ideal data distribution and the actual, observed distribution. The purpose of statistical machine learning is to deduce the ideal data distribution from the observed distribution. We will write some first code here to better understand the relationship between both distributions.

In this task, we create a ground truth distribution, and draw an empirical distribution from it. We can "misuse" a gray-scale image for this task, namely if we imagine that the $x$- and $y$-coordinates of a pixel are feature dimensions $x_1$ and $x_2$, and the grayscale intensity represents the relative density at its respective $(x, y)$-location.

**Exercise 1** Use `scipy.misc.face(gray=true)` to load a gray-scale image of a raccoon face. Apply a Gaussian filter (e.g., with $\sigma = 3$) to slightly smooth the image. Use `matplotlib` to visualize the image.

Implement a function that draws new samples from this density, using the approach with the cumulative density function from the lecture. Maybe you can re-use some code from the warmup exercise? To visualize the drawn samples, it is most convenient to create a new (empty) image with the same dimensions as the raccoon, and to draw a point at every location that is sampled from the density. An example is shown in Fig. 1.

Draw 10.000, 50.000, 100.000, 200.000, and 400.000 samples. What do you observe?

**Exercise 2** Use the Parzen Window Estimator to "reconstruct" the image. Thus, use the sampled density from the previous task as input, and output a density from the Parzen window estimator.

More in detail, Implement a Parzen window estimator with a box kernel. Use our empirical raccoon densities from the previous task as an input. Vary the window size of the Parzen estimator. What do you observe?

**A First Taste of the Model Selection Problem**

Which kernel size is best suited for our raccoon density? Implement a cross-validation to automatically determine the best kernel size from a reasonably large list of candidate kernel sizes (with at least 3 candidate sizes). The objective function shall be chosen analogously to the lecture.

If you are familiar with cross-validation, then please just go ahead and implement it. Otherwise, you may follow these steps: to perform $k$-fold cross-validation, split the samples $\mathcal{S}$ into a test set $\mathcal{S}_j$ of size $|\mathcal{S}_j| = N/k$ and a training set $\mathcal{T}_j = \mathcal{S}$

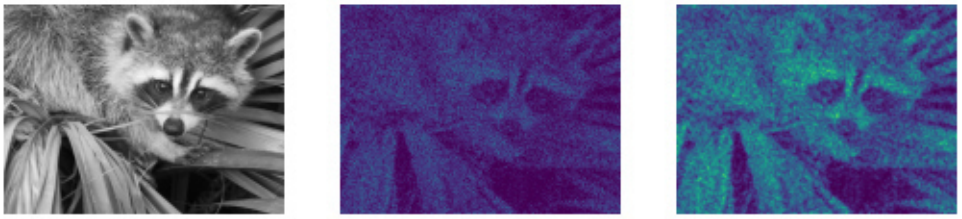

Figure 1: Ground truth distribution (raccoon) as well as sampled density (100000 samples) and a reconstruction with a Parzen Window kernel of width 9.

$\mathcal{S}_j$. Build a density $p(\mathbf{x})_\theta^j$ from $\mathcal{T}_j$ where $\theta$ denotes the candidate kernel size. Choose the best $\theta$ with an ML estimation across all folds, by choosing $\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \sum \log p(\mathbf{x})_\theta^j$ on all $\mathbf{x} \in \mathcal{T}_j$ across all folds $j$.

Play with it! Which observations are new or surprising to you? In other words, what do you experience in the experiments that was not obvious to you from the theoretical derivation of the model selection problem?

Please create a figure with the raccoon or any other image of your choice as well as a sampling and a reconstruction with the best kernel size (example in Fig. 1). Please post it to the forum and add a short text about one observation you made.

*Comments:*
*We ask for only one figure per group. Please also state your group number. Bring your code to the joint meeting on Mai 6 or Mai 7 for a potential little extra experiment.*

Exercises for
Pattern Analysis Programming
Lina Felsner, Mathias Seuret, Dalia Rodriguez-Salas
Task 3: May 6 – May 13

Pattern Recognition Lab

# Dealing with Highly Imbalanced Sample Distributions

In the previous task, we interpreted the intensities in our raccoon picture as relative frequencies in a raccoon PDF. However, we could also interpret these intensities as function values, and then aim to approximate these values via regression. This can also be done with a kernel, as shown, in Sec. 6.1 of our textbook by Hastie, Tibshirani, and Friedman.

The main advantage of regression via kernel smoothing is its simplicity. On the downside, this simplicity can be the source of systematic errors in the regression result.

The distribution of samples is always imbalanced at the boundary of the domain: here, samples can only be inside of the domain, but not outside. Thus, to regress a value $x_0$ close to the boundary, the kernel can only make use of samples on one side of the kernel window. Similar situations can also arise within the domain whenever the samples are very unevenly distributed within a local neighborhood.

In this exercise, we will investigate the special case at the boundary of the domain. This case is illustrated in Fig. 6.3 and Fig. 6.4 of Hastie, Tibshirani, Friedman (→ pages 195 and 196). The goal of this exercise is to reproduce this experiment, and to obtain a Figure that is similar to Fig. 6.3 and the left part of Fig. 6.4. An example how this could look link is shown in Fig. 1.

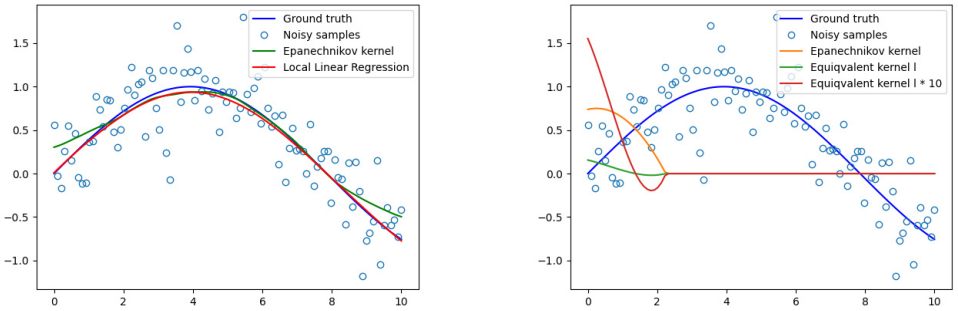


Figure 1: Left: Ground truth and noisy samples together with the regression with an Epanechnikov kernel and a a kernel-weighted local linear regression. Right: Kernels plotted for $x_0 = 0.25$. Note that the local linear equivalent kernel may become negative.

**Exercise 1** Create the dataset. First, you will need a ground truth function (dark blue in Fig. 6.3). In our implementation, clamping the domain of $y = \sin(0.4x)$ looked qualitatively quite similar, but feel free to choose your own function if you like.

Draw noisy random samples along that function. In our implementation, we use additive Gaussian noise, which is a typical simple baseline.

**Exercise 2** Perform kernel smoothing regression on the noisy samples to obtain the equivalent to the green curve in the left plot of Fig. 6.3. Use an Epanechnikov kernel, like Tibshirani, Hastie, Friedman, with a sufficiently large window size. Can you observe a similar error at the boundary? If not, wiggle on your function until a similar error occurs.

**Exercise 3** Perform a kernel-weighted local linear regression, as described in Eqn. (6.8) and Eqn. (6.9), and plot this estimate. Hopefully, the boundary region is better extrapolated now? How well are the other areas of your function represented? Please also plot the Epanechnikov kernel and the local linear equivalent kernel (shown in Fig. 6.4). Do you observe a similar difference in both kernels as shown in the book?

**Please note:** Equations (6.8) and (6.9) are directly derived from a standard linear regression task. We find this derivation instructive, and we will tomorrow upload a short video on it to sec. A in studOn.

*Comments:*
*We ask for only one figure per group. Please also state your group number. Bring your code to the joint meeting on May 13 or May 14 for a potential little extra experiment.*

Exercises for
Pattern Analysis Programming
Lina Felsner, Mathias Seuret, Dalia Rodriguez-Salas
Task 4: May 13 – May 20

Pattern Recognition Lab

# Transforming an Unsupervised Task to a Supervised Task

**Preliminary Remarks: Density Estimation as a Regression Task**
There are unsupervised models that operate on the empirical probability density distribution of the data, and there are supervised models to perform classification and regression. Supervised methods can be further split up into a generative probability model for the joint density $p(\mathbf{x}, \mathbf{y})$ of features $\mathbf{x}$ and labels $\mathbf{y}$, or a discriminative model that directly target at $p(\mathbf{y}|\mathbf{x})$.

There is a good reason to have specialized methods for different learning tasks: task-specific performance. However, from a theoretical perspective, it is quite interesting to note how closely related the seemingly disparate tasks of unsupervised learning, regression, and classification are.

This exercise is loosely inspired by Sec. 14.2.4 in "The Elements of Statistical Learning" by Hastie, Tibshirani, Friedman: let us convert the density estimation task into a regression task.

The key idea is to define a second density that models the background. Assign high values to the actual density samples (e.g., the ones from our raccoon), and low values to the background density. Then, the regression task is to fit one function that represents both distributions, with high values in regions where our actual samples dominate, and with low values in regions where the background dominates.

## Exercise 1

Reuse the raccoon sampler from the previous exercise to create a density of the samples. We also need a background distribution: what might be a good distribution for representing the background? Hastie, Tibshirani, Friedman suggest that unless we have a better idea, we can certainly choose a uniform distribution, so let us work with a uniform distribution. Note that from a theoretical perspective, you will need at least as many samples in the background distribution as in the foreground distribution.

Use a random forest to fit the regression model. That way, we can also earn some practical experience with parameterization of such a forest. Random forests are implemented in `scikit.learn.ensemble`, specifically in the class `RandomForestRegressor`.

You can check the performance of your regressor visually, by making a regressor prediction for each pixel of the raccoon image: does the output look like a raccoon? A more quantitative approach that also allows to tune the forest parameters would be to reuse the cross-validation from the previous exercise.

## Exercise 2

Repeat your experiments using now `ExtraTreesRegressor`. What exactly is the difference between a `RandomForestRegressor` and an `ExtraTreesRegressor`? Play with the amount of samples as well as with the amount of trees and maximum height. Which parameter settings provide satisfying results? By tendency, are the requirements for the amount of training data larger or smaller than for our Parzen window estimator? Choose a set of parameters to create a figure with your results for both regressors (example in Fig. 1). Please post it to the forum

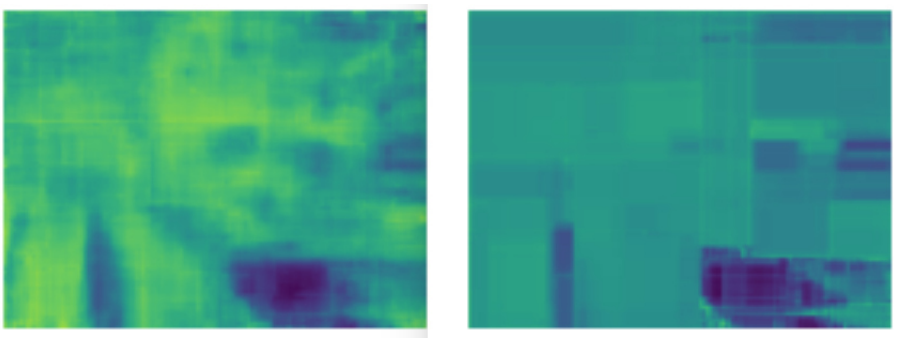

Figure 1: Left side images correspond to ExtraTreesRegressor. Right side images correspond to RandomForestRegressor. Amount of samples = 100000 (50000 foreground + 50000 background), Amount of trees = 20, Maximum depth = 10

and add a short text about one observation you made.

**Optional: A Second Dataset**
Use the function sklearn.datasets.make moons to create a synthetic den- sity of two interwoven half moons. Repeat your experiments with the RandomForestRegressor and the ExtraTreeRegressor. Are your observations consistent with your observations on the previous dataset? If you like, you can post a figure of the moons instead or additional to the raccoon image to the forum. Please add a short text about one observation you made.

*Comments:*
*We ask for only one figure per group. Please also state your group number. Bring your code to the joint meeting on May 20 (or May 21 (not sure yet, we will keep you informed)) for a potential little extra experiment.*

Exercises for
Pattern Analysis Programming
Lina Felsner, Mathias Seuret, Dalia Rodriguez-Salas
Task 5: May 20 – May 27

Pattern Recognition Lab

# Clustering: K-Means versus Mean Shift

Let us have a closer look at the inner workings of K-Means and Mean Shift. Mean Shift clustering assigns each point to its associated mode in the density. K-Means clustering assigns each point to the nearest mean vector. The purpose of this exercise is to understand under which circumstances these two approaches lead to different results.

## Exercise 1

For this exercise, we will resort to a synthetic distribution that consists of random samples from three 2-D Gaussian distributions. Use K-Means and Mean Shift algorithms to cluster the samples.

We believe it is a good exercise if you implement these algorithms yourself. The implementation is not difficult, and it helps to deepen your understanding of the algorithms. However, it is also not forbidden to use already implemented versions, such as `sklearn.cluster.MeanShift` or `sklearn.cluster.KMeans`.

For details on the K-Means algorithm, please have a look at the book by Hastie/Tibshirani/Friedman Sec. 14.3.6 or in the book by Bishop Sec. 9.1. Details for the Mean Shift algorithm can be found in the paper by Comaniciu/Meer (uploaded to studOn Sec. A), together with the lecture video and slides 02b.

You can quickly test if both algorithms work by having the center of each of the three Gaussians well separated from each other. An example is shown in Fig. 1.

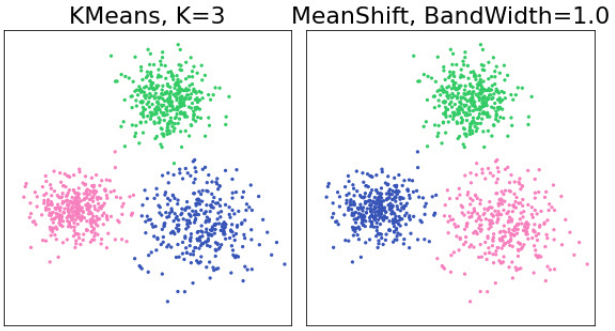


Figure 1: Example with three Gaussian distributions. The clustering result for K-Means (left) and Mean Shift (right) are color coded.

## Exercise 2

From the results of the first part of the exercise (e.g. Fig. 1), the difference between the algorithms is not clear. Both give the same clustering result. However, from the theory we can expect that there is a difference. Think about and design your own density, where the two algorithms give different results. Any density is valid, and there are certainly different possibilities and approaches to create such a density. Maybe think about a density that is particularly simple, or that realizes the difference between the methods in a particularly instructive way; or

just report an unexpected case. There is an infinite amount of different possible densities you can try, so have fun exploring!

Please post a figure of your density with the respective results of K-Means and Mean Shift clustering on the forum and add a short text describing what made you choose that specific density.

*Comments:*
*We ask for only one figure per group. Please also state your group number. Bring your code to the joint meeting on May 27 or May 28 for a potential little extra experiment.*

Exercises for
Pattern Analysis Programming
Lina Felsner, Mathias Seuret, Dalia Rodriguez-Salas
Task 6: May 27 – June 3

Pattern Recognition Lab

# K-Means and the Gap-Statistics

In this exercise, we will play with the gap statistics for model selection of k-means clusters. If you like to look at the literature, it can be found in Sec. 14.3.11 of Hastie/Tibshirani/Friedman. If you like to read the original source, you can find the original work by Tibshirani in the literature section of the studon class.

**Exercise 1** Implement Tibshirani's gap statistic to automatically select the correct number of components of the clustering. In case that you compare Sec. 14.3.11 of Hastie/Tibshirani/Friedman with the original paper, you will notice that the original paper proposes to optionally align the sampling of the reference density with the principal components of the original data. Although this is certainly a smart move for cases of ill-shaped clusters, you may leave it out for this exercise and just follow Sec. 14.3.11. For each experiment, please produce four plots as shown in Fig. 1. These plots are:

(a) the input clusters, color-coded by their membership to the multivariate Gaussians

(b) the k-means result after selecting k with the gap statistics, color-coded by their membership to the k clusters.

(c) the gap statistics, including the standard deviation of the reference clusters (i.e., the right part of Fig. 14.11 in Hastie/Tibshirani/Friedman)

(d) the decrease of the within-cluster distance of the original data (i.e., the left part of Fig. 14.11 in Hastie/Tibshirani/Friedman)

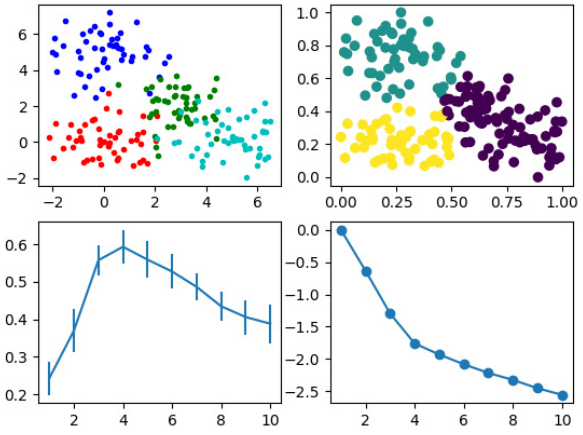

Figure 1: Top left: color-coded input clusters. Top right: color-coded k-means result after selecting k with the gap statistics. Bottom left: the gap statistics, including the standard deviation of the reference clusters. Bottom right: the decrease of the within-cluster distance.

Play with your new method! In which cases is the gap statistic likely to underestimate the number of clusters k? One aspect that complicates the analysis is that both our data generation and also the k-means clustering are randomized. What might be a reasonable approach to nevertheless measure the performance of the gap statistics? What are the issues with our raccoon density? Extra brain teaser: can you think of a case where the gap statistic overestimates the number of clusters? Extra brain teaser: assume that all we want from our raccoon is to cluster some of the bright parts of the original image – what might be a reasonable approach to make this happen?

Please select a density of your choice and post a figure similar to the one in Fig. 1 to the forum and add a short text about it.

*Comments:*
*We ask for only one figure per group. Please also state your group number. Bring your code to the joint meeting on June 3 or June 4 for a potential little extra experiment.*

Exercises for
Pattern Analysis Programming
Lina Felsner, Mathias Seuret, Dalia Rodriguez-Salas
Task 7: June 10 – June 17

Pattern Recognition Lab

# Dimensionality Reduction

Not only the human intuition breaks down in higher dimensions, but also for some algorithms (e.g. clustering) it is favourable to bring the data to a lower dimension. That means, that we want to find a good representation in a lower dimensional space, that reduces the dimensions of the feature vectors in such a way, that the (possible) data-structure is preserved. In this exercise, we will look at different dimensionality reduction methods and compare them with each other.

**Exercise 1** In this exercise we will compare the following algorithms with each other:

(a) Principal Component Analysis (PCA) (Lecture 2h, Bishop Sec. 12.1.1)

(b) Multidimensional Scaling (MDS) (Lecture 2i)

(c) Iso Map (IM) (Lecture 2j)

(d) Laplacian Eigenmaps(LE) (Lecture 2j)

You can implement them by yourself, or just use the corresponding function from the `sklearn` python-library.

To get a *feeling* for the algorithms and be able to visualize the results, we will restrict ourselves in this task to 2D and 3D densities / feature vectors and reduce them by one dimension only. Please test and compare the algorithms qualitatively on the following 2D and 3D data sets:

- One Gaussian distribution (2D or 3D)
- Multiple Gaussian Distributions in 2D
- Multiple Gaussian Distributions in 3D
- Two circles in 2D (`sklearn.datasets.make_circles`)
- Swiss roll in 3D (`sklearn.datasets.make_swiss_roll`)

Figure. 1 gives an example for each data set.

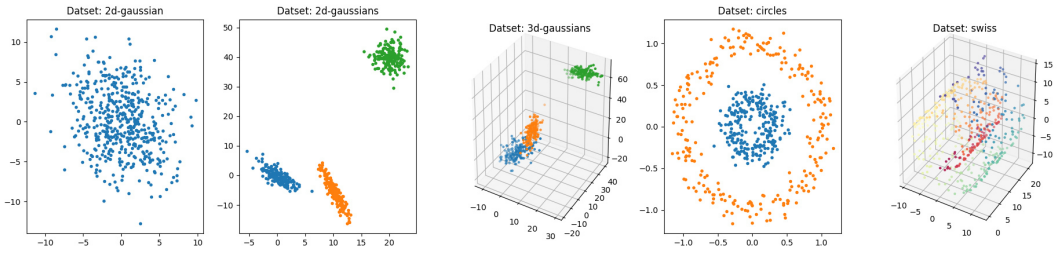


Figure 1: Five different data sets in 2D and 3D.

What results do you get by applying the different algorithms to the different data sets? What result (distribution combined with algorithm) might have surprised you? Play around with the parameters (e.g. number of samples, amount of noise, etc.). Are all algorithms stable?

Advanced python skills: a very nice representation of the dimensionality reduction would be, to visualize a representation of the lower dimension in the original data set (e.g. line, plane, or higher dimensional surface)...

Please post a figure of one of the data sets and the respective lower dimensional representations to the forum and add a short text about it.

*Comments:*
*We ask for only one figure per group. Please also state your group number. Bring your code to the joint meeting on June 17 or June 18 for a potential little extra experiment.*