Introduction to Machine Learning
Unsupervised Learning - Density Estimation

Ramon Fuentes[1,2], Artur Gower[3]

August 9, 2019

[1]Visiting Researcher, Dynamics Research Group
The University of Sheffield

[2]Research Scientist, Callsign Ltd

Unsupervised Learning

Types of unsupervised learning tasks:

- Density estimation
- Clustering
- Feature Extraction / Dimensionality Reduction

Unsupervised Learning

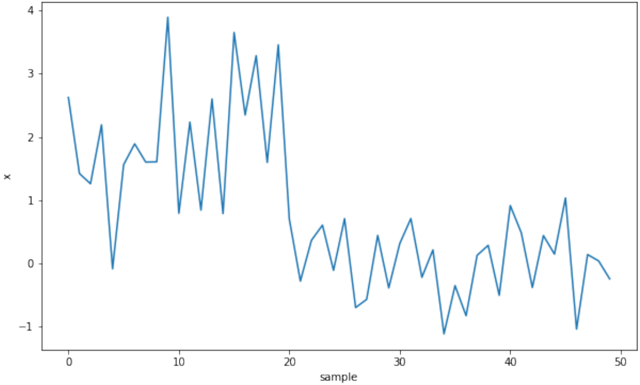
For now, we will focus on **density estimation** (because we don't have infinite time)

So what is it exactly?

Density Estimation

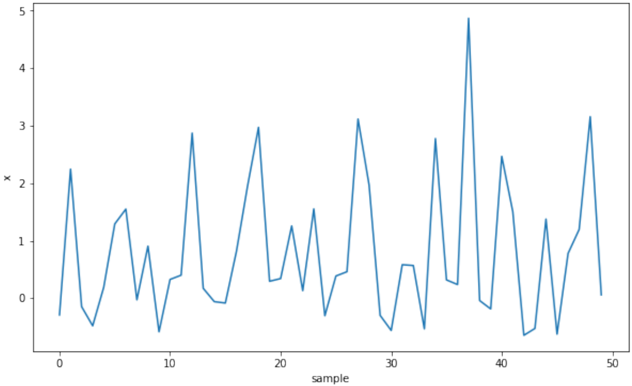
Density estimation seeks to answer the question: how is my data distributed?

Can you spot the pattern on this data set?



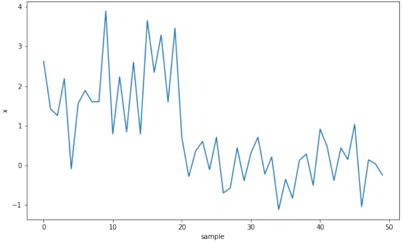
Density Estimation

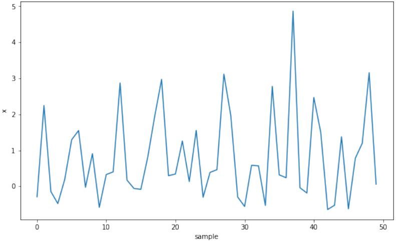
How about this one?



Density Estimation

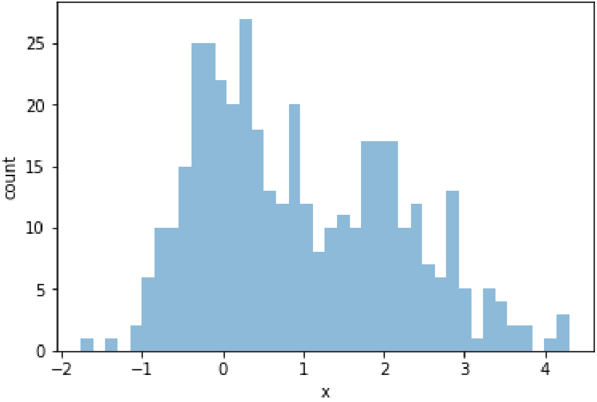
They are the same, we've just reshuffled them!





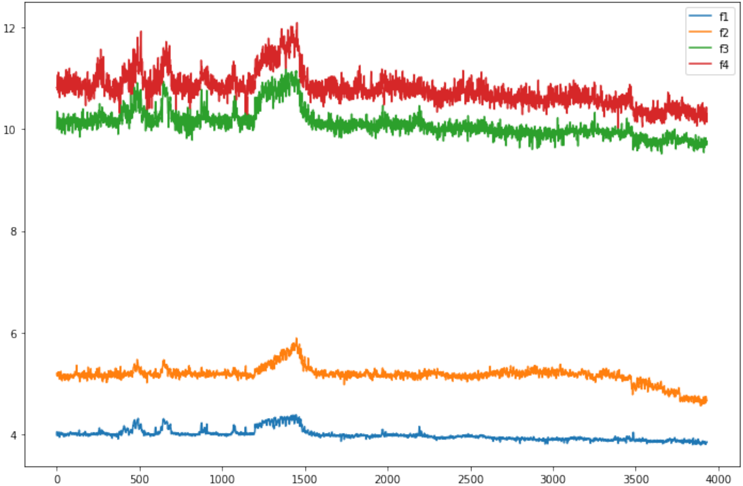
Density Estimation

One of the simplest ways of looking at how data is distributed is through a histogram



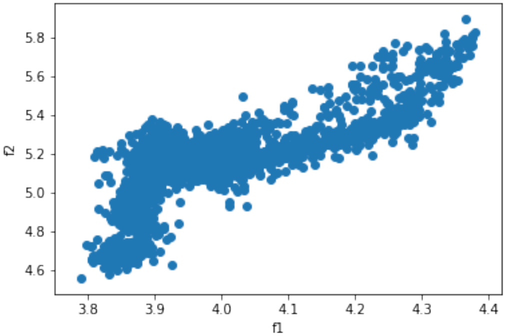
Density Estimation

And what happens when we have more than one dimension? Like in our bridge data...



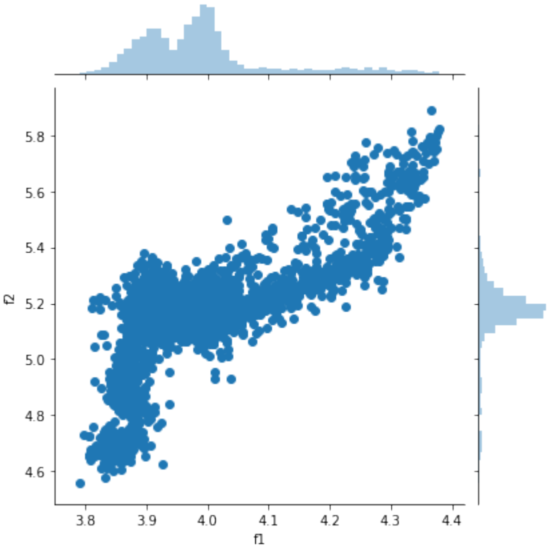
Density Estimation

We could look at 2 dimensions with a scatter plot



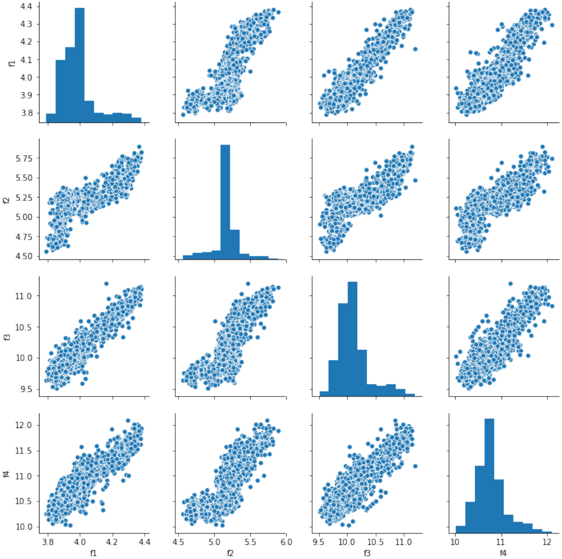
Density Estimation

And we could add histograms...



Density Estimation

We could look at all 4 dimensions at the same time



Density Estimation

- There are many wonderful plots we can create to get insight into our data
- Visualising things is important, but...
  - It does not scale to high dimensions
  - It doesn't **quantify** anything
- Why might we want to quantify the density of our data?
  - To detect abnormal data.
  - To find groupings or clusters in our data.

Density Estimation

There are two kinds of density estimation techniques:

- Parametric: small models but assume a simple shape for the data distribution.
- Non-parametric: large models which can accommodate any data distribution.

We'll be looking at both kinds

Parametric Density Estimation

- In density estimation, we model the data's density with the function
$$p = p(\mathbf{x})$$
- For parametric density estimation the Gaussian distribution is widely used (though not always the most appropriate)

Gaussian distribution

A Gaussian distributions models the *probability density* of data using two free parameters that model the mean location μ and the scatter variance $2\sigma^2$.

In one dimension:

$$p(x^*) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\Big(- \frac{|x^* - \mu|^2}{\sigma^2}\Big),$$

where x^* is the point you want to predict the density $p(x^*)$, and

$$\mu = E[x_i] = \frac{1}{n}\sum_i^n x_i,$$

Gaussian distribution

A Gaussian distributions models the *probability density* of data using two free parameters that model the mean location μ and the scatter variance $2\sigma^2$.

In one dimension:

$$p(x^*) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\Big(- \frac{|x^* - \mu|^2}{\sigma^2} \Big),$$

where x^* is the point you want to predict the density $p(x^*)$, and

$$\mu = E[x_i] = \frac{1}{n} \sum_i^n x_i,$$

$$\sigma^2 = E[(x_i - \mu)^2] = \frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2.$$

Gaussian distribution

In d dimensions we want to predict the density at $\mathbf{x}^* = [x_1^*, x_1^*, \ldots, x_d^*]^T$ using the data $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^T$.

Gaussian distribution

In d dimensions we want to predict the density at $\mathbf{x}^* = [x_1^*, x_1^*, \ldots, x_d^*]^T$ using the data $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^T$.

We can achieve that with a multivariate Gaussian distribution:

$$p(\mathbf{x}^*) = \frac{1}{(2\pi)^{d/2}\det(\mathbf{S})^{1/2}} \exp\Big(-\frac{1}{2}(\mathbf{x}^* - \boldsymbol{\mu})^T \mathbf{S}^{-1} (\mathbf{x}^* - \boldsymbol{\mu})\Big)$$

where now the mean is the vector $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_d]$ and the covariance $\mathbf{S}$ is a $d \times d$ matrix.

Gaussian distribution

In d dimensions we want to predict the density at $\mathbf{x}^* = [x_1^*, x_1^*, \ldots, x_d^*]^T$ using the data $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]^T$.

We can achieve that with a multivariate Gaussian distribution:

$$p(\mathbf{x}^*) = \frac{1}{(2\pi)^{d/2}\det(\mathbf{S})^{1/2}} \exp\Big(-\frac{1}{2}(\mathbf{x}^* - \boldsymbol{\mu})^T \mathbf{S}^{-1}(\mathbf{x}^* - \boldsymbol{\mu})\Big)$$

where now the mean is the vector $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_d]$ and the covariance $\mathbf{S}$ is a $d \times d$ matrix.

$$\boldsymbol{\mu} = E[\mathbf{x}_i] = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i = \begin{bmatrix} (x_{11} + x_{21} + \ldots + x_{n1})/n \\ \vdots \\ (x_{1d} + x_{2d} + \ldots + x_{nd})/n \end{bmatrix},$$

$$\mathbf{S} = E[(\mathbf{x}_i - \boldsymbol{\mu})^T(\mathbf{x}_i - \boldsymbol{\mu})] = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T.$$

Gaussian distribution

To calculate the matrix **S** let's take a closer look at this matrix multiplication in 2D, that is d = 2:

$$
\begin{aligned}
(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T &= \begin{bmatrix} x_{i1} - \mu_1 \\ x_{id} - \mu_d \end{bmatrix} \begin{bmatrix} x_{i1} - \mu_1 & x_{i2} - \mu_2 \end{bmatrix} \\
&= \begin{bmatrix} (x_{i1} - \mu_1)^2 & (x_{i2} - \mu_2)(x_{i1} - \mu_1) \\ (x_{i2} - \mu_2)(x_{i1} - \mu_1) & (x_{i2} - \mu_2)^2 \end{bmatrix}
\end{aligned}
$$

To calculate **S**, we need to sum over i:

$$
\begin{aligned}
\mathbf{S} &= \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T \\
&= \frac{1}{n-1} \sum_{i=1}^{n} \begin{bmatrix} (x_{i1} - \mu_1)^2 & (x_{i2} - \mu_2)(x_{i1} - \mu_1) \\ (x_{i2} - \mu_2)(x_{i1} - \mu_1) & (x_{i2} - \mu_2)^2 \end{bmatrix}.
\end{aligned}
$$

Gaussian distribution

To calculate the matrix **S** let's take a closer look at this matrix multiplication in 2D, that is d = 2:

$$(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T = \begin{bmatrix} x_{i1} - \mu_1 \\ x_{id} - \mu_d \end{bmatrix} \begin{bmatrix} x_{i1} - \mu_1 & x_{i2} - \mu_2 \end{bmatrix}$$
$$= \begin{bmatrix} (x_{i1} - \mu_1)^2 & (x_{i2} - \mu_2)(x_{i1} - \mu_1) \\ (x_{i2} - \mu_2)(x_{i1} - \mu_1) & (x_{i2} - \mu_2)^2 \end{bmatrix}$$

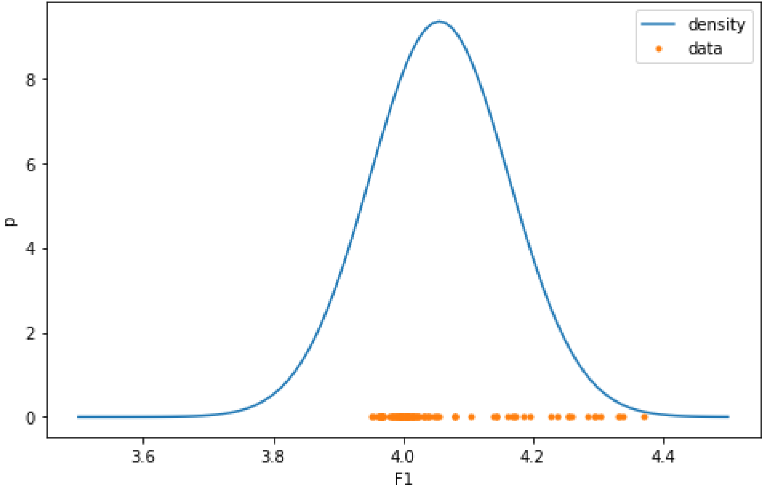
To calculate **S**, we need to sum over i:

$$\mathbf{S} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T$$
$$= \frac{1}{n-1} \sum_{i=1}^{n} \begin{bmatrix} (x_{i1} - \mu_1)^2 & (x_{i2} - \mu_2)(x_{i1} - \mu_1) \\ (x_{i2} - \mu_2)(x_{i1} - \mu_1) & (x_{i2} - \mu_2)^2 \end{bmatrix}.$$

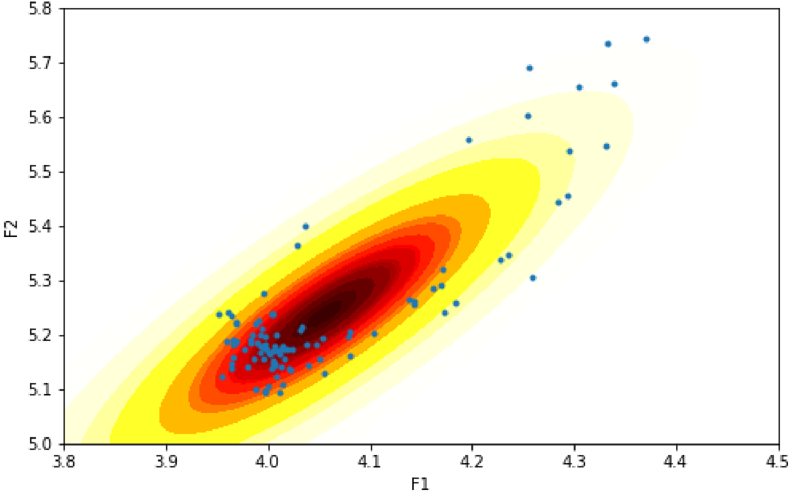
If $\mathbf{x}_i$ had d dimensions then **S** would be a $d \times d$ matrix.

Gaussian distribution - example

Lets fit this to our bridge data - in 1D

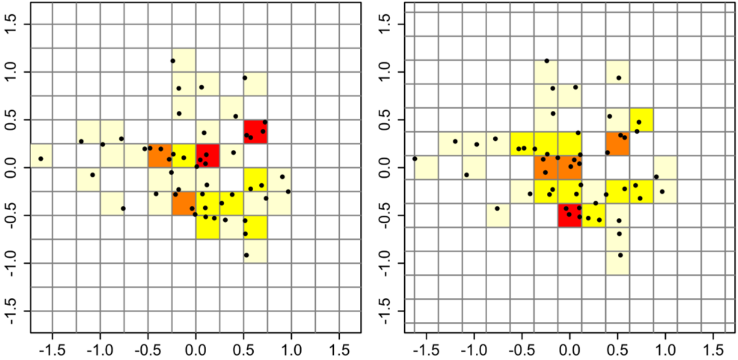


Lets fit this to our bridge - in 2D



Kernel Density Estimation

- To motivate the use of kernels for density estimation, it helps to see some of the shortcomings of histograms[1].
- A histogram can change significantly when changing the position of the bins:



[1] https://en.wikipedia.org/wiki/Multivariate_kernel_density_estimation

Kernel Density Estimation

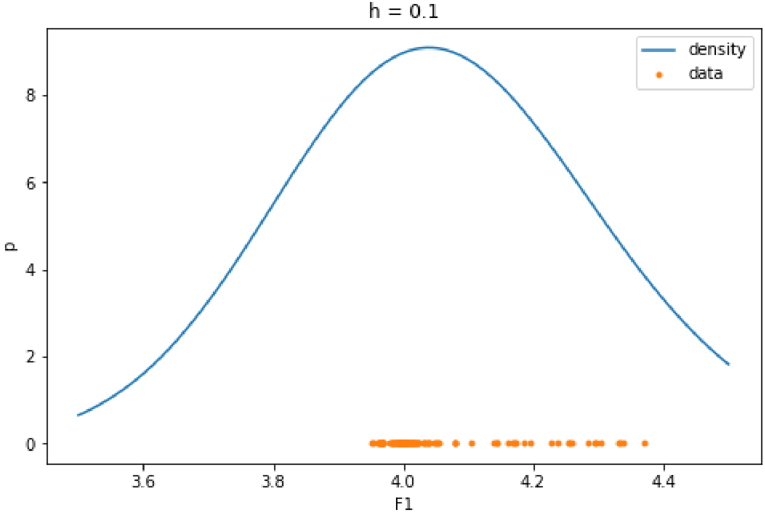
- Kernel methods can accurately estimate any data distribution (non-parametric density estimation).

$$p(\mathbf{x}^*) = \frac{1}{nh} \sum_{i=1}^{n} \kappa(\mathbf{x}^*, \mathbf{x}_i), \tag{1}$$

  where $\kappa(\mathbf{x}^*, \mathbf{x}_i)$ is exactly the same kernel function used for kernel ridge regression!
- For example the Gaussian kernel : $\kappa(\mathbf{x}, \mathbf{x}') = \exp(-\frac{|\mathbf{x}-\mathbf{x}'|^2}{h})$, where h is now called the bandwidth/length-scale hyper-parameter.
- the Gaussian kernel is also a popular choice, and leads to smooth densities.
- and as before, we'll have to tune the hyper-parameter h that controls fit quality.

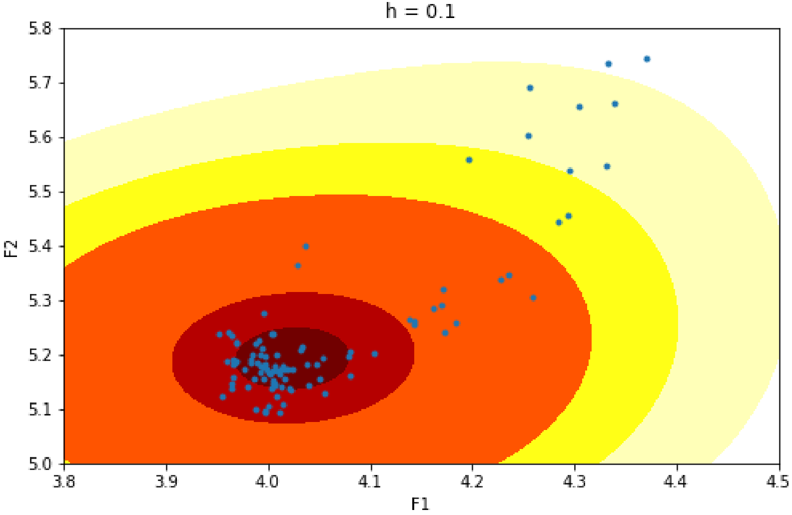
Kernel Density Estimation - example

Lets see how kernel density does on our bridge data... in 1D



Kernel Density Estimation - example

Lets see how kernel density does on our bridge data... in 2D



Conclusions

we have,

- learned about density estimation
- looked at one of the most popular parametric techniques: the Gaussian distribution
- learned about non-parametric density estimation, with kernels
- these both extend easily to multiple dimensions