Skript Pattern Analysis Sommersemester 2017

Nils Häusler, Moritz Grand, Adrian Meyer

June 6, 2017

1 Density Estimation

Let $p(\vec{x})$ denote a probability density function pdf then:

1. $p(\vec{x}) \geq 0$
2. $\int_{-\infty}^{\infty} p(\vec{x})\, d\vec{x} = 1$
3. $p(\vec{a} \leq \vec{x} \leq \vec{b}) = \int_{\vec{a}}^{\vec{b}} p(\vec{x})\, d\vec{x}$

The task of density estimation is to obtain a continous representation of the underlying pdf from a set of discrete samples (massumants). Note: that if we have the pdf we can do statistical analysis.

Parametric density estimation (mostly Pattern Recognition)
Make an assumption about the underlying distribution (e.g. Gausian, GMM) and determine the best fitting distribution parameters from the data. (ML estimation, MAP estimation)

Non-parametric density estimation We make no assumption of the underlying Model.

1.1 Parzen-Rosenblatt estimator

???Idea: Quantify the number of samples with a window

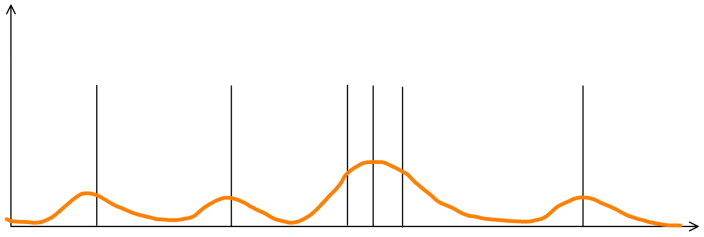


Figure 1: A PDF describing the distribution of measurements

The Parzen window estimator interpolates the pdf from the observations in the neighbourhood of a position x, using an appropriate kernel/window function.

Short derivtion: Let p_R denote the probability that $\vec{x}$ lies within region R:

$$p_R = \int_R p(\vec{x})\, d\vec{x}$$

Now assume that $p(\vec{x})$ is approximately constant in R.

$$p_R \approx p(\vec{x}) \int_R d\vec{x}$$

For example, let R be a d-dimensional hypercube with side length h, then its volume[1][2] is h^d

$$p_R \approx p(\vec{x}) V_R$$

Let $p_R = \frac{k_R}{N}$, we determine the probability of making observations in region R by counting the samples in R ($= k_R$) and dividing by the total number of samples. Note: p_R is also called the "relative frequency"

$$p(\vec{x}) = \frac{p_R}{V_R} = \frac{k_R}{V_R N}$$

Let's write the parzen window estimator as a function of a kernel[3] $k(\vec{x}; \vec{x_i})$, then

$$p(\vec{x}) = \frac{1}{h^d N} \sum_{i=1}^{N} k(\vec{x}; \vec{x_i})$$

where[4]

$$k(\vec{x}; \vec{x_i}) = \begin{cases} 1 & \text{when } \frac{|x_{i,k} - x_k|}{h} \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases}$$

equivalently, if we use a (multivariate) gaussian kernel:

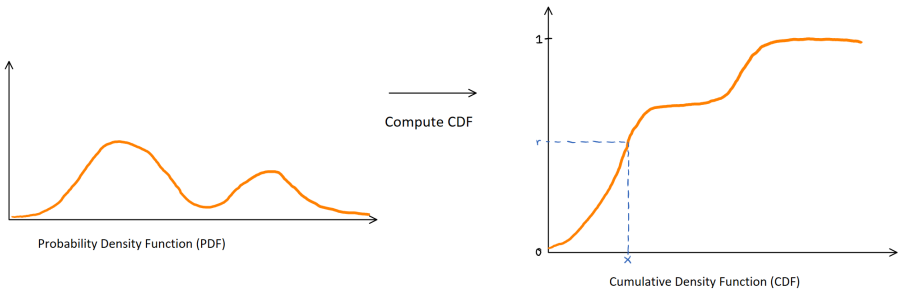
$$k(\vec{x}; \vec{x_i}) = \frac{1}{(2\pi)^d |\Sigma|} e^{-(\vec{x}-\vec{x_i})^T \Sigma^{-1} (\vec{x}-\vec{x_i})}$$

A note on applications

- General remark: We obtain a continuous pdf, i.e. desity estimation converts a list of measurments to a statistical model
- Specific example: We can sample from a pdf. This means that we have a princeple way of generating new / more / ... data that behaves / looks / ... similary to the observations.

[1] $\int_R d\vec{x}$ is just the volume of R
[2] We also write V_R for the volume
[3] Omit h^d if the kernel is gaussian
[4] $\vec{x_i}$ and $\vec{x}$ are not father apart then $0.5h$ in any dimension k

Q: How can we (practically) sample from a pdf?



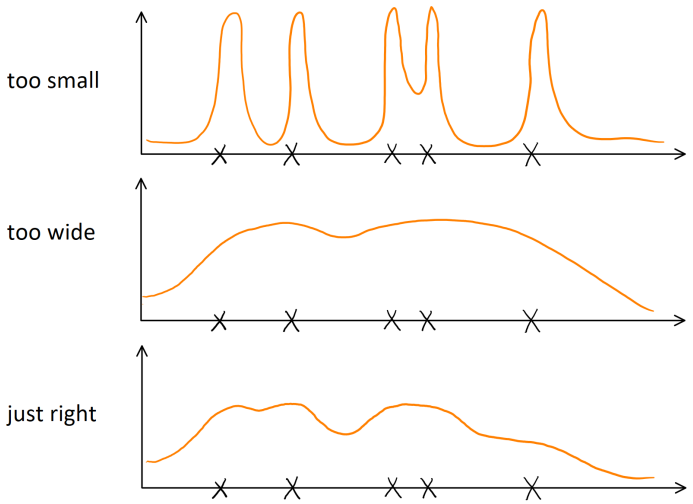
Compute through discretisation of the pdf $cdf[i] = cdf[i-1] + pdf[i]$. Then draw a uniformly distributed number (r) between 0 and 1. The sampled value is x where $cdf[x] = r$

Q: How can we determine a good window / kernel width h?
Let's do ML est. with a cross-vaslidation (cv) (e.g. leave-one-sample-out cv)

$$p^j_{h,N-1}(\vec{x}) = \frac{1}{h^d N} \sum_{i=1(i\neq j)}^{N} k(\vec{x}; \vec{x_i})$$

We estimate the pdf from all samples except $\vec{x_j}$. $\vec{x_i}$ will be used to evaluate the quality of the pdf using window siye h.

Q: How do the results change with varing window size?



$$\hat{h} = \arg\max_h L(h) = \arg\max_h \prod_{j=1}^{N} p^j_{h,N-1}(\vec{x_j}) = \arg\max_h \sum_{j=1}^{N} \log p^j_{h,N-1}(\vec{x_j})$$

The position of the maximum does not change, because the logarithm is a strictly monotonic function.

2 Mean Shift Algorithm

Purpose: Find maximum in pdf without actually performing a full density estimation. [5]

Potential applications: Clustering, segmentation, ...

Idea: Maxima can be found, where the gradient of the pdf is zero. (Assume that we have a full density estimator.)

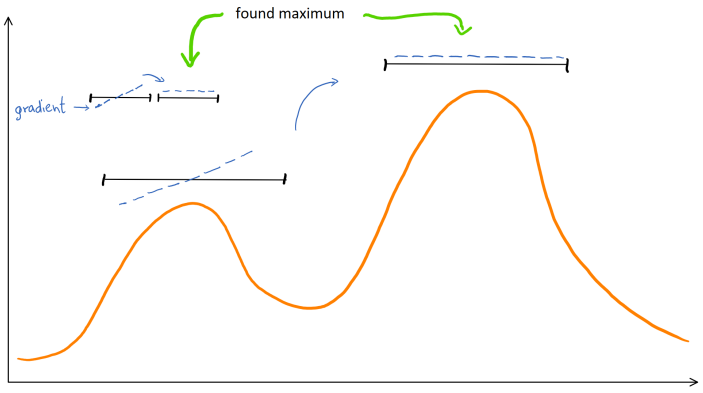


Figure 2: The kernel size indirectly controls the number of indentified maxima

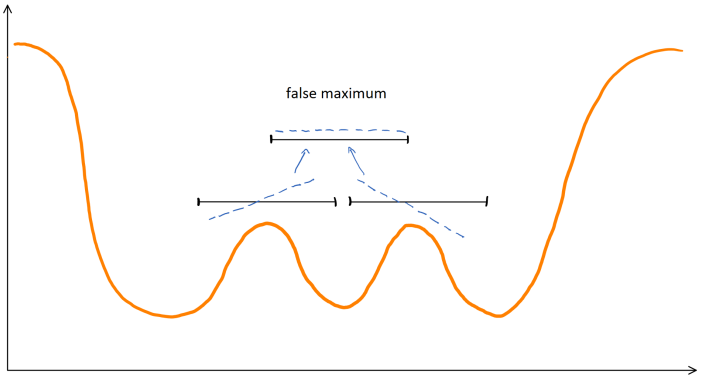


Figure 3: One of the issues is, the case when a zero gradient is just between two finer maxima

[5] This applies only in some cases, e.g. quickly finding clusters through particle tracing or with a downsampled PDF (see section 2)

Let

$$p(\vec{x}) = \frac{1}{N}\sum_{i=1}^{N} k_h(\vec{x};\vec{x_i})$$

denote the multivariate kernel density estimation. A local maxium of the pdf can be assumed where the gradient vanishes $\nabla p(\vec{x}) = 0$.

$$\nabla p(\vec{x}) = \nabla(\frac{1}{N}\sum_{i=1}^{N} k(\vec{x};\vec{x_i})) = \frac{1}{N}\sum_{i=1}^{N} \nabla k(\vec{x};\vec{x_i})$$

Let's assume that k_h is a radially symmetric kernel, i.e.

$$k(\vec{x};\vec{x_i}) = c_d k_h(||\vec{x_i} - \vec{x}||^2)$$

$$\frac{\partial k_h(S)}{\partial S} = k_h'(S)$$

$$\frac{\partial S}{\partial \vec{x}} = \frac{\partial(\vec{x_i} - \vec{x})^T(\vec{x_i} - \vec{x})}{\partial \vec{x}} = -2(\vec{x_i} - \vec{x})$$

$$\nabla p(\vec{x}) = \frac{1}{N}\sum_{i=1}^{N} c_d k_h(||\vec{x_i} - \vec{x}||^2)(-2(\vec{x_i} - \vec{x})) \doteq \vec{0}$$

$\frac{1}{N}$ and c_d can be dropped then multiply out

$$\sum_{i=1}^{N} k_h'(||\vec{x_i} - \vec{x}||^2)\vec{x_i} - \sum_{i=1}^{N} k_h'(||\vec{x_i} - \vec{x}||^2)\vec{x} = \vec{0}$$

Then we get the mean shift vector

$$\frac{\sum_{i=1}^{N} k_h'(||\vec{x_i} - \vec{x}||^2)\vec{x_i}}{\sum_{i=1}^{N} k_h'(||\vec{x_i} - \vec{x}||^2)} - \vec{x} = \vec{0} \tag{1}$$

To perform a gradient acsend, compute the gradient, walk one step, re-compute the gradient, walk a step, ...

Mean shift algorithm (formalized)

1. Compute the mean shift vector $m(\vec{x}^{(t)})$ (see 1)
2. Update $\vec{x} : \vec{x}^{(t+1)} = \vec{x}^{(t)} + m(\vec{x}^{(t)}) = \vec{x}^{(t)} + \frac{\sum_{i=1}^{N} k_h'(||\vec{x_i} - \vec{x}||^2)\vec{x_i}}{\sum_{i=1}^{N} k_h'(||\vec{x_i} - \vec{x}||^2)} - \vec{x}^{(t)} = \frac{\sum_{i=1}^{N} k_h'(||\vec{x_i} - \vec{x}||^2)\vec{x_i}}{\sum_{i=1}^{N} k_h'(||\vec{x_i} - \vec{x}||^2)}$

Q: Why is it called "mean shift"?
If we plug in for k_h the Epanechnikov kernel. Then the computation breaks down to the mean of the samples in a circular (hyperspherical) around $\vec{x}^{(t)}$

Epanechnikov kernel

$$k_E(\vec{x}) = \begin{cases} c(1 - \vec{x}^T\vec{x}) & \text{when } \vec{x}^T\vec{x} \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Abstract example: Assume we have a 2-D feature space

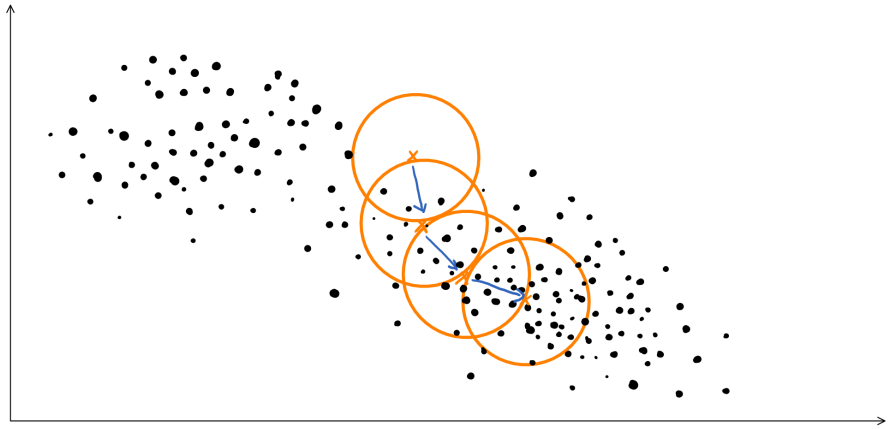


Figure 4: Mean Shift iterations with an Epanechnikov kernel

Specific example:

- (Color) quantization
  - Note: the RGB colorspace is not perceptually uniform (Lab or Luv are used in practice)
- (Color) segmentation
  - Similar in result to a super pixel segmentation: operate locally in the image
  - Incorporate the position of each pixel

Remarks on the found maxima:

- Different trajectories typically coverage only to **almost** the same peak, thus, we will have to post process the peaks and somehow reduce them.

- We don't have a guarentee to sit on top of a maximum when reaching a 0-gradient. This is due to the finite window size and the discrete representation of our density (see figure 2).
- If the amount of data is large, then it may become extremly costly to iteratively evaluate the "neighbourhood finder". In that case we have to help ourself, either with a smart data structure (oct-tree or a generalisation for many dimensions) or locality sensitve hashing (LSH).

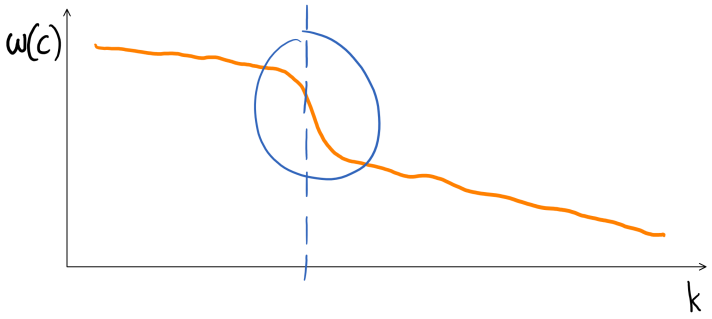
3 Clustering

Variants of k-means: For clustering see section 14 of "The Elements of Statistical Learning" (Hastie, Tibshirani, Friedman - 2009). Section 14.3 "Cluster Analysis" introduces different flavours of k-means. Points of interest are:

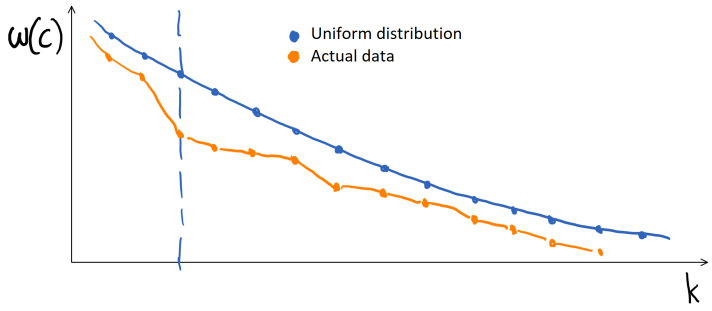
- Proximity Matrices
- K-means
- Gaussian Mixtures as Soft K-means Clustering
- Vector Quantization

Q: How can we determine k?
Let $w(c)$ be distance from samples to cluster centers within clusters. We use this as a metric of quality in cluster analysis.

Approach 1: Track the rate of change of a quality metric (like $w(c)$). Proposed in "Pattern Classification" (Duda, Hart, Stork).



Approach 2: Let $w'(c)$ be a metric on a uniform distribution of samples. Relate change of $w(c)$ to change of $w'(c)$. Proposed in TEoSL.



??? is something missing here ???
We have several options for clustering data. K-means and its variants are straight forward and easy to understand. But if the proper number of clusters is part of the unknowns, it may be a suboptimal choice. One alternative is mean shift clustering. Here is the number of clusters determined implicitly by the kernel type and size plus the type of bump post process.
Now we will look into a Dirichlet Process to model infinite gaussian mixtures.

The idea for infinite mixture models: Instead of fitting a specific distribution to the data (the GMM), we define a meta-distribution from which the actual distribution is drawn. specifically we can draw a GMM from a dirichlet process[6]. However this is a top-down perspective, i.e. if we randomly draw a GMM, we will certainly not draw one that fits nicely to our data. From a bottom up perspective, we need a fitting algorithm that works with the available data points and finds a GMM in such a way, that it could also have been drawn from a distribution of distributions (the dirichlet process).

Chinese restaurant process: To illustrate the model behind the fitting algorithm, we usually talk about the "Chinese restaurant process".

Extensions to a "normal" GMM:

1. If a customer prefers to sit at a new table, this is possible we can add arbitrary many tables

[6]Because uniform ??? parameters for drawing the GMM

2. The more people sit on a table the more attractive it is to the new customers (rich get richer)

The chinese restaurant process (CRP) provides a constructive way of sampling from a dirichlet process.

Gibbs sampling: A straight forward clustering algorthm based on the CRP
while(true)
???
Tis is a probailistic algorithm, bcause we sample from the list ..???.

Back to the top-down view: Using the CRP, we are effectivly drawing a GMM using a dirichlet process. Implicitly, the mixture weihts β_i match a $Beta(\alpha)$ distribution and the normal distributions are drawn from a hyper-distribution, i.e. $\vartheta_i \sim H(\lambda)$

Stick breaking process: (Illustration of a $Beta$ distribution)

Qualitatively, this is what the $Beta$ distribution looks like:

The expansion parameter α is a prior that influences the number of clusters that will be created.

Observation 1: The larger the expansion parameter α is, the more likely it is to draw a number that is close to 0 form $Beta(1, \alpha)$

The number that has been drawn form $Beta(1, \alpha)$ is used in the stick breaking process to determine the weight of the i-th mixture component β_i

Stick breaking process (again):

$$b_i \sim Beta(1, \alpha)$$

$$\beta_i = b_i \prod_{i=1}^{i-1} (1 - b_i) = b_i (1 - \sum_{i=1}^{i-1} \beta_i)$$

Manifold Learning

Principal idea: Reduce the dimensions of the Data, gut preserve the structure of the data / underling manifold.

Curse of Dimensionality

"Human intuition breaks in high dimensional spaces :("

Let's illustrate this: Consider a d-dimensional feature vector $\vec{x_1}, \vec{x_2}, \ldots, \vec{x_N} \in \mathbb{R}^d$ where $0 \leq x_{i,k} \leq 1$ (??? wo kommt das k her) uniformly distributed in a d-dimensional hyperspace of volume 1. Let's say we would like to cover enough volume of the cube to collect 1% of the data. Let's say we also use a cube for this task. What is the required edge length s of the cube to obtain that 1% of space?

Example: A 10 dimensional hypercube

$$V = s^d \Rightarrow s = V^{\frac{1}{d}} = 0,01^{\frac{1}{10}} = 0.63$$

Another way of thinking about this is, that in a very high dimensional space, virtually every feature point is located at the boundary of the feature space[7]. This leads to the effect, that common distance measures loose their effectivity. E.g. the *median* distance for the nearest neighbour to the origin.

Techniques for dimensionality reduction

Recap: "Notorious" example for high dimensional data:

- hyper-spectral remote sensing image classification
- satellite image: perform e.g. agricultural monitoring classify type of vegetation from hyper-spectral (many color channels) image.

[7] Because in at least on dimension, we draw a very low of very high value

Our known approch is PCA.

Orthogonal basis that is aligned with the "maximum spread" (w.r.t. the covariance) of the data. It is a global unsupervised method[8]. Consider that PCA os a linear method.

The Kernel PCA performs a non-linear mapping of the data, then perform a standard (linear) PCA on the result. With the kernel-trick we can do that in one step. The Objective function: (the non-linear mapping is part of ϕ)

$$\delta = sum_{i,j=1}^{N}(\phi(\vec{x_i}) - \phi(\vec{x_j}))^T(\phi(\vec{x_i}) - \phi(\vec{x_j})) + \lambda(\phi^T\phi - 1)$$

Some offsprings of the Kernel PCA idea are:

1. Perform some preprocessing / mapping that operates non-linearly
2. The dimensionality reduction itself is an operation that operates linearly.

[8]Operating on all data points, no labels, find one global optimal solution

Multi Dimensional Scaling (MDS)

ISOMAP Algorithm

Locally Linear Embedding (LLE)

Laplacian Eigenmaps

Random Forrest