Kernel Density Estimates and Mean-Shift Clustering

Jonas Spinner* – 1927895
Analytics and Statistics
KIT – Karlsruhe Institute of Technologie

January 21, 2019

*jonas.spinner@student.kit.edu

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Listings

Abstract

In this seminar paper we are describing and evaluating the mean shift algorithm. It is a nonparametric clustering method which does not depend on prior knowledge on the number and shape of the clusters.

The clusters are defined by the modes of the data density and a iteration procedure is used to let a point converge to its cluster centroid. We introduce the kernel density estimate for estimating the data density and then derive the mean shift algorithm from it.

After presenting the methods we also present different applications of the algorithm, mainly its use in image segmentation.

1 Introduction

In this seminar paper I am going to discuss the method of kernel density estimation (KDE) and it's use in the mean-shift clustering algorithm (MSC). Clustering is one of the main tasks in Machine-Learning and MSC has many applications, mainly in image segmentation. The mean-shift algorithm is an non-parametric approach which allows a wide range of data distributions without imposing prior knowledge.

The content of this seminar paper is mainly based on Comaniciu & Meer (2002).

History

The mean shift algorithm was introduced in Fukunaga & Hostetler (1975) where also the term "mean shift" was established. Comaniciu & Meer (2002) and Comaniciu et al. (2003) are responsible for a gain in interest for the algorithm. These publication popularized the use of the algorithm in image segmenation and tracking.

Notation

In this paper I'm going to use the following conventions. Bold symbols $\boldsymbol{x}$ denotes a vector. $\{\boldsymbol{x}_i\}_{i=1}^n$ denotes n samples $\boldsymbol{x}_i$, indexed by i. d is the dimensionality of the samples. $K(\cdot)$ is a multivariate kernel function $\mathbb{R}^d \to \mathbb{R}$ and $k(\cdot)$ denotes a single variable kernel function $\mathbb{R} \to \mathbb{R}$.

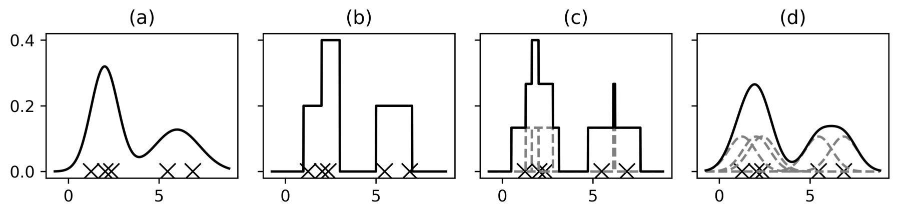

Figure 1: Samples, histogram and kernel density estimates. Author's illustration. (a) The probability density distribution $p(x) = \frac{3}{5}N(x; 2, 0.75^2) + \frac{2}{5}N(x; 6, 1.25^2)$ and five samples drawn from it. (b) A histogram with bins $[i, i+1)$, $i \in \mathbb{Z}$. (c) A kernel density estimate with a uniform kernel and bandwidth $h = 0.75$, $k(x) = \frac{1}{2h}$ for $|x| \leq h$ and 0 else. (d) A kernel density estimate with a gaussian kernel and bandwidth $h = 0.75$, $k(x) = \frac{1}{h}\frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}|\frac{x}{h}|^2)$.

2 Kernel Density Estimation

In this section we are going to introduce the kernel density estimate. It will be used in the derivation of the mean shift algorithm.

We assume that we have n data points $\{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$, which are independently, identically distributed samples from a unknown probabiltiy distribution $p(\boldsymbol{x})$, $\boldsymbol{x}_i \overset{i.i.d.}{\sim} p(\cdot)$. We call the corresponding density function f.

The goal of density estimation is estimating the probality density of p, given samples $\{\boldsymbol{x}_i\}_{i=1}^n$ in $\mathbb{R}^d$, $\boldsymbol{x} \sim p(\boldsymbol{x})$.

Histograms estimate the probability distribution by bucketing the samples and counting the proportion of samples which fall in each bucket.

The **kernel density estimate** is defined as

$$\hat{f}(\boldsymbol{x}) = \frac{1}{nh^d}\sum_{i=1}^{n} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right) \qquad \boldsymbol{x} \in \mathbb{R}^d, \tag{1}$$

where n is the sample size, d is the dimensionality of the data, $K(\boldsymbol{x})$ is a kernel function and h is a bandwidth parameter. The estimate is dependent on the choice of the kernel function and the bandwidth.

A **kernel function** $K(\boldsymbol{x})$ is a function that satisfies $K(\boldsymbol{x}) \geq 0$ and $\int_{\mathbb{R}^d} K(\boldsymbol{x})d\boldsymbol{x} = 1$. These restrictions on the kernel ensure that $\hat{f}(\boldsymbol{x})$ is a valid density. The **bandwidth** h controls the smoothness of the kernel density estimate. We are going

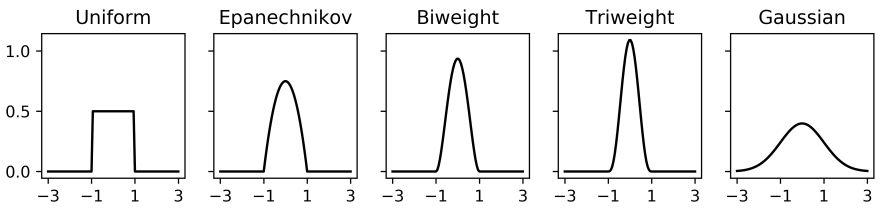

Figure 2: Popular kernels. Author's illustration. The first four kernels all have support $[-1, 1]$. The gaussian kernel has the support $(-\infty, \infty)$.

to discuss the choices for kernel functions and the bandwidth in the next section.

The kernel density estimate can be interpreted as the sum of "bumps" centered on each data point.

For non-negative kernels, one can see that $\hat{f}(\boldsymbol{x})$ is a valid probability density.

2.1 Other methods

Another method for estimating a probability density function are histograms. The samples $\{\boldsymbol{x}_i\}_{i=1}^n$ are put into distinct buckets $b_j \subseteq \mathbb{R}^d$ and the probability is estimated by the sample frequency $p(j) = |\{\boldsymbol{x}_i \mid \boldsymbol{x}_i \in b_j\}/n$ for each bucket.

2.2 Popular Kernels

A general Kernel is a function $K : \mathbb{R}^d \to \mathbb{R}$ satisfying the following properties. See (Comaniciu & Meer 2002) and (Wand & Jones 1995, p. 95).

$$\begin{aligned} \int_{\mathbb{R}^d} K(\boldsymbol{x})\mathrm{d}\boldsymbol{x} &= 1 & \int_{\mathbb{R}^d} \boldsymbol{x}K(\boldsymbol{x})\mathrm{d}\boldsymbol{x} &= \boldsymbol{0} \\ \lim_{\|\boldsymbol{x}\|\to\infty} \|\boldsymbol{x}\|^d K(\boldsymbol{x})\mathrm{d}\boldsymbol{x} &= 0 & \int_{\mathbb{R}^d} \boldsymbol{x}\boldsymbol{x}^T K(\boldsymbol{x})\mathrm{d}\boldsymbol{x} &= c_K \boldsymbol{I} \end{aligned}$$

$$K^P(\boldsymbol{x}) = \prod_{j=1}^{d} k(x_j) \qquad K^S(\boldsymbol{x}) = a_{k,d}k(\|\boldsymbol{x}\|)$$

Although there a many kernel functions, we are going to concentrate us on one class of functions: **radially symmetric kernels**. They are the kernels which can be formulated as

$$K(\boldsymbol{x}) = c_{k,d} k(\|\boldsymbol{x}\|^2), \tag{2}$$

where $k : [0, \infty]) \to [0, \infty)$ is called the **profile** of K, and $c_{k,d}$ is a constant which ensures, that $K(\boldsymbol{x})$ integrate to 1.

The derivation of the mean shift algorithm is going to be easier and nearly all popular kernels already belong to this class. The most popular kernels for mean shift clustering are the uniform, Epanechnikov and the Gaussian kernel. We are going to shortly introduce them and their profile.

The **uniform kernel** is defined as being constant in the unit hypersphere and can be written as

$$K_U(\boldsymbol{x}) = \begin{cases} \operatorname{vol}(S_d)^{-1} & \text{for } \|\boldsymbol{x}\| \leq 1 \\ 0 & \text{else} \end{cases}, \tag{3}$$

where $\operatorname{vol}(S_d) = \pi^{d/2}\Gamma((d+2)/2)^{-1}$ is the volume of the d-dimensional hypersphere with $\Gamma(\cdot)$ being the gamma function. If $d = 1$ then $\operatorname{vol}(S_d) = 2$. The corresponding profile k_U is

$$k_U(u) = \begin{cases} 1 & \text{for } 0 \leq u \leq 1 \\ 0 & \text{for } 1 < u. \end{cases} \tag{4}$$

The **Epanechnikov kernel** is defined as

$$K_E(\boldsymbol{x}) = \begin{cases} \frac{2+d}{2\operatorname{vol}(S_d)}(1 - \|\boldsymbol{x}\|^2) & \text{for } \|\boldsymbol{x}\| \leq 1 \\ 0 & \text{else} \end{cases} \tag{5}$$

and its profile k_E is

$$k_E(u) = \begin{cases} 1 - u & \text{for } 0 \leq u \leq 1 \\ 0 & \text{for } 1 < u. \end{cases} \tag{6}$$

The **gaussian kernel** is defined as

$$K_G(\boldsymbol{x}) = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}\|\boldsymbol{x}\|^2\right) \tag{7}$$

and its profile k_G is

$$k_G(u) = \exp\left(-\frac{1}{2}u\right). \tag{8}$$

Name	Profile support	$k(u)$	$-k'(u)$	$K(\boldsymbol{x})$
Uniform	$u \in [0,1]$	1	0	$\text{vol}(S_d)^{-1}$
Epanechnikov	$u \in [0,1]$	$1-u$	1	$\frac{1}{2}\text{vol}(S_d)^{-1}(d+2)\left(1-\lVert\boldsymbol{x}\rVert^2\right)$
Biweight	$u \in [0,1]$	$(1-u)^2$	$2(1-u)$	$\propto \left(1-\lVert\boldsymbol{x}\rVert^2\right)^2$
Triweight	$u \in [0,1]$	$(1-u)^3$	$3(1-u)^2$	$\propto \left(1-\lVert\boldsymbol{x}\rVert^2\right)^3$
Gaussian	$u \in [0,\infty)$	$\exp\left(-\frac{1}{2}u\right)$	$\frac{1}{2}\exp\left(-\frac{1}{2}u\right)$	$(2\pi)^{-d/2}\exp\left(-\frac{1}{2}\lVert\boldsymbol{x}\rVert^2\right)$

Table 1: Popular kernel functions. The kernel functions listed here are all radially symmetric kernels. By looking at the profiles $k(u)$ and the negated derivative $-k'(u)$ we can see the corresponding shadows (see section 3.3). vol(S_d) is the volume of a d-dimensional hypershere and the normalization constants for the biweight and triweight kernels are omitted.

2.3 Bandwidth Selection

One of the main challenges in using the kernel density estimate in practice is the choice of the bandwidth. The univariate case has a rich theoretical background, but the multivariate case is more problematic. (Comaniciu & Meer 2002, section 3.1) lists some general approaches for bandwidth selection.

- The stability of the clustering in respect to the bandwidth parameter h is analysed. The number of resulting clusters K is stated as a function of the bandwidth, $K(h)$. Then the midpoint of the largest range for which $K(h)$ stays constant is chosen.
- Some objective function is used for evaluating the clustering quality. The bandwidth which maximizes the objective function is chosen.
- When the desired number of clusters is known, the bandwidth can be varied to accomplished the desired number. This approach can be modified to ignore smaller clusters.
- Another approach is to estimate the bandwidth by the mean K nearest neighbor distance between the data points. K is set to be a percentage of the whole dataset, for example 30 %. The Python library scikit-learn from Pedregosa et al. (2011) implements this approach in the method `estimate_bandwidth`. In section 4 we implement this method in Matlab and use it in our experiments. The code can be found in Code Listing 3.

$$\hat{f}_{-i}(\boldsymbol{x}) = \frac{1}{(n-1)h^d}\sum_{j\neq j} K\left(\frac{\boldsymbol{x}_j - \boldsymbol{x}_j}{h}\right) \qquad \boldsymbol{x} \in \mathbb{R}^d \tag{9}$$

$$\begin{aligned}\mathrm{MLCV}(h) &= \frac{1}{n}\sum_{i=1}^{n} \log \hat{f}_{-i}(\boldsymbol{x}_i) && (10)\\ &= \frac{1}{n}\sum_{i=1}^{n} \log\left(\sum_{j\neq j}^{n} K\left(\frac{\boldsymbol{x}_j - \boldsymbol{x}_j}{h}\right)\right) - \log(n-1) - d\log(h) && (11)\end{aligned}$$

3 Mean Shift Clustering

Clustering is one of the main machine learning tasks, concerned with grouping objects $\{x_i\}_{i=1}^n$ into clusters C_j resulting in a clustering $\{C_j\}_{j=1}^K$.

3.1 Clustering algorithms

introduction

What does define a good clustering? One can try to minimize a global criterion like $J = \sum_{j=1}^{K}\sum_{\boldsymbol{x}_i \in C_j} \|\boldsymbol{x}_i - \boldsymbol{m}_j\|^2$, like the K-Means algorithm, or minimize the intra-cluster min, max or average distances between the data points, like in hierarchical clustering. The notion of a "good" clustering brings rise to many different clustering algorithms.

Another criteria to differentiate clustering algorithms, are the assumptions that it makes on the data. K-Means assumes the data is separable into K clusters.

There are many different notions of what defines a good clustering and what shapes a cluster can take. These different notions result in different clustering algorithms. One class of clustering algorithms is centroid based clustering. Each cluster is represented by a representative called centroid. For example the representative of a K-Means clustering is the mean of the cluster $m_j := \frac{1}{|C_j|}\sum_{x_i \in C_j} x_i$.

centroid based

A common concept for several algorithms are **centroids**. A centroid is a point in the data space, which is a representative for a cluster. K-Means defines its centroids as the cluster mean $\boldsymbol{m}_j = \frac{1}{|C_j|}\sum_{\boldsymbol{x}_i \in C_j} \boldsymbol{x}_i$. A shortcoming of this definition is that the cluster mean is not guaranteed to "look like" a typical data object.

The mean shift algorithm defines its cluster representatives by the modes of the data density. This definition solves several of shortcomings of other algorithms. The algorithm does not make prior assum

Another way clustering algorithms can be differentiated is whether of not the amount of clusters is given as an input to the algorithm or not. For example the K-Means algorithm searches for exactly K clusters. But there are also other notions of a cluster center or representative. One is the added restriction, that the center is itself a data object or atleast "looks like" one. The latter is covered by using representatives which are likely also a data object. That's the core of the mean shift clustering algorithm. It estimates the underlying density of the data objects and searches for points which high relative probability. Or in other words it identifies the modes of the estimated density.

3.2 Density gradient estimation

Before we look at the specifics of the mean shift algorithm in section 3.4, we look at an estimation of the density gradient and derive the mean shift vector. This section is based on Comaniciu & Meer (2002) and follows its notation.

By defining clusters with the modes of a density function the clustering task becomes a mode finding task where we search for $\boldsymbol{x}$ where the density gradient is zero, $\nabla f(\boldsymbol{x}) = 0$. As we have no knowledge about the real density $f(\boldsymbol{x})$, we have to use an estimate. Recall that the kernel density estimate is defined as

$$\hat{f}(\boldsymbol{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right). \tag{12}$$

We estimate the density gradient $\nabla f(\boldsymbol{x})$ by the gradient of the density estimate. Formally we write

$$\hat{\nabla} f(\boldsymbol{x}) \equiv \nabla \hat{f}(\boldsymbol{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} \nabla K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right). \tag{13}$$

For simplification we are going to concentrate on the class of **radially symmetric kernels**. These can be written as

$$K(\boldsymbol{x}) = c_{k,d} k(\|\boldsymbol{x}\|^2) \tag{14}$$

where $k : [0, \infty]) \to [0, \infty)$ is called the **profile** of K, and $c_{k,d}$ is a constant which ensures that $K(\boldsymbol{x})$ integrate to 1. The most common kernels belong to this class of functions.

Because we are going to use different kernel density estimates, we explicit state the used bandwidth h and kernel K of the kernel density estimate as $\hat{f}_{h,K}(\boldsymbol{x})$. The kernel density estimate can now be formulated as

$$\hat{f}_{h,K}(\boldsymbol{x}) = \frac{c_{k,d}}{nh^d} \sum_{i=1}^{n} k\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_i}{h} \right\|^2 \right) \tag{15}$$

and its gradient as

$$\nabla \hat{f}_{h,K}(\boldsymbol{x}) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (\boldsymbol{x} - \boldsymbol{x}_i) k'\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_i}{h} \right\|^2 \right). \tag{16}$$

With the substitution $g(u) = -k'(u)$ and reordering, we can formulate the gradient as

$$\nabla \hat{f}_{h,K}(\boldsymbol{x}) = \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_i}{h} \right\|^2 \right) \right] \left[\frac{\sum_{i=1}^{n} \boldsymbol{x}_i g\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_i}{h} \right\|^2 \right)} - \boldsymbol{x} \right]. \tag{17}$$

We can analyse the terms separately. The last term is called **mean shift vector**

$$\boldsymbol{m}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} \boldsymbol{x}_i g\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_i}{h} \right\|^2 \right)} - \boldsymbol{x}. \tag{18}$$

We derived an estimate for the density gradient and introduced the mean shift vector $\boldsymbol{m}(\boldsymbol{x})$. In section 3.3 we are going to further analyse the properties of $\boldsymbol{m}(\boldsymbol{x})$. In section 3.4 we are finally presenting the mean shift algorithm and a short discussion follows in section 3.5.

3.3 Properties of the mean shift vector and kernel shadows

We can further analyse the equation 17 and relate the mean shift vector to an kernel density estimate. The first two factors can be decomposed into a scaling factor and a kernel density estimate with the kernel $G(\boldsymbol{x}) = c_{g,d} g(\|\boldsymbol{x}\|^2)$

$$\frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_i}{h} \right\|^2 \right) \right] = \frac{2c_{k,d}}{h^2 c_{g,d}} \left[\frac{c_{g,d}}{nh^d} \sum_{i=1}^{n} g\left(\left\| \frac{\boldsymbol{x} - \boldsymbol{x}_i}{h} \right\|^2 \right) \right] \tag{19}$$

$$= \frac{2c_{k,d}}{h^2 c_{g,d}} \hat{f}_{h,G}(\boldsymbol{x}). \tag{20}$$

With these equations we can reformulate the density gradient estimate $\nabla \hat{f}_{h,K}(\boldsymbol{x})$ into

$$\nabla \hat{f}_{h,K}(\boldsymbol{x}) = \frac{2c_{k,d}}{h^2 c_{g,d}} \hat{f}_{h,G}(\boldsymbol{x}) \boldsymbol{m}(\boldsymbol{x}). \tag{21}$$

That allows us to write the mean shift vector as a normalized gradient of the kernel density estimate $\nabla \hat{f}_{h,K}(\boldsymbol{x})$

$$\boldsymbol{m}(x) = \frac{h^2 c_{g,d}}{2c_{k,d}} \frac{\nabla \hat{f}_{h,K}(\boldsymbol{x})}{\hat{f}_{h,G}(\boldsymbol{x})}. \tag{22}$$

This observation means that the mean shift vector calculated with the kernel G points in the gradient direction of the kernel density estimate calculated with the kernel K. For the kernels $G(\boldsymbol{x}) \propto g(\|\boldsymbol{x}\|^2)$ and $K(\boldsymbol{x}) \propto k(\|\boldsymbol{x}\|^2)$ if

$$-k'(u) \propto g(u) \tag{23}$$

then we call the kernel K a **shadow** of kernel G.

The gaussian kernel is its own shadow. Recall that the profile of the gaussian kernel is $k_G(u) = \exp\left(-\frac{1}{2}u\right)$. It holds that $-k'_G(u) \propto k_G(u)$.

A shadow of the uniform kernel is the Epanechnikov kernel, because the profile of the Epanechnikov kernel is $k_E(u) = (1-u)\mathbb{1}\{u \in [0,1]\}$ and $-k'_E(u) = k_U(u) = \mathbb{1}\{u \in [0,1]\}$.

The mean shift vector $\boldsymbol{m}(\boldsymbol{x})$ calculated with the kernel G is therefore not the gradient direction of an kernel density estimate calculated with G, but of the kernel density estimate calculated with a more “complex” kernel, the shadow of G.

The mean shift vector $\boldsymbol{m}(\boldsymbol{x})$ inhibits some nice properties that makes it easy to compute. In section 2 we required that the kernel must integrate to 1. This requirement can be dropped because the normalizing constants cancel in the expression.

3.4 The mean shift algorithm

With the provided prerequisites the mean shift algorithm can be quickly stated. Recall that the mean shift vector is

$$\boldsymbol{m}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} \boldsymbol{x}_i g\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right\|^2\right)} - \boldsymbol{x}. \tag{24}$$

For each $\boldsymbol{x}_i$ we perform the iterative procedure

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \boldsymbol{m}(\boldsymbol{x}^{(t)}), \qquad t = 1, 2, ... \tag{25}$$

where $\boldsymbol{x}^{(1)} = \boldsymbol{x}_i$. The procedure is stopped when $\boldsymbol{x}^{(t)}$ converges or at least the difference $\left\| \boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)} \right\|$ is small enough. $\boldsymbol{z}_i$ gets assigned the value $\boldsymbol{x}^{(t)}$ is converged to.

After calculating $\boldsymbol{z}_i$ for each $\boldsymbol{x}_i$, the points where $\boldsymbol{z}_i = \boldsymbol{z}_j$ are assigned to the same cluster. As before we relax this by assigning the same cluster when $\|\boldsymbol{z}_i - \boldsymbol{x}_j\| < \varepsilon$ holds.

The pseudo code for the mean shift algorithm can be found in Algorithm 1.

Algorithm 1 Mean-shift algorithm.

```
1:  function MeanShift($\boldsymbol{x}_1, ..., \boldsymbol{x}_n : \mathbb{R}^d$, $h : \mathbb{R}^+$, $\varepsilon : \mathbb{R}^+$)
2:      for $i = 1..n$ do
3:          $\boldsymbol{x}^{(1)} = \boldsymbol{x}_i$, $t = 0$
4:          repeat
5:              $t = t + 1$
6:              $\forall j : w_j = \exp(-\frac{1}{2} \left\| (\boldsymbol{x}^{(t)} - \boldsymbol{x}_j)/h \right\|^2)$
7:              $\boldsymbol{x}^{(t+1)} = (\sum_{j=1}^n w_j)^{-1} \sum_{j=1}^n w_j \boldsymbol{x}_j$
8:          until $\left\| \boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)} \right\| < tol$
9:          $\boldsymbol{z}_i = \boldsymbol{x}^{(t+1)}$
10:     return ConnectedComponents($\{\boldsymbol{z}_i\}_{i=1}^n, \varepsilon$)
```

When G is the uniform kernel, the procedure is guaranteed to converge. For the samples $D = \{\boldsymbol{x}_i\}_{i=1}^n$ the number of weighted means of the subsets $D' \subseteq D$ is 2^n. For each iterate $\boldsymbol{x}^{(t)}$ there exists some t_0 so that $\boldsymbol{x}^{(t_0+t)} = \boldsymbol{x}^{(t_0)}$ for $t \geq 0$.

For the gaussian kernel only convergence for $t \to \infty$ is guaranteed. (citation needed)

The core of the mean shift algorithm is an iterative procedure which acts on a single point $\boldsymbol{x}^{(0)}$. For the given point the algorithm finds the corresponding mode of the kernel density estimate ($\hat{\nabla} f(\boldsymbol{x}) = \boldsymbol{0}$). Each iteration step the point moves in the direction of the steepest ascent. The iteration stops when the point has converged to a fixed point $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)}$. For the clustering of a dataset, the iteration procedure is performed on each point. The samples used for the kernel density estimate are still fixed to the original dataset.

A variant of the algorithm is the blurring mean shift algorithm. There the kernel density estimate is based on the data points from the previous iteration step.

Algorithm 2 Mean-shift algorithm in matrix form.

```
1: function MeanShift($\{\boldsymbol{x}_i\}_{i=1}^n \subset \mathbb{R}^d$, $h : \mathbb{R}^+$, $\varepsilon : \mathbb{R}^+$)
2:     $\boldsymbol{Z} = \boldsymbol{X} : \mathbb{R}^{d\times n}$
3:     repeat
4:         $\boldsymbol{W} = (\exp(-\frac{1}{2}\left\|(\boldsymbol{x}_i - \boldsymbol{x}_j)/h\right\|^2))_{i,j=1..n}$
5:         $\boldsymbol{D} = \mathrm{diag}(\sum_i \boldsymbol{W}_{ij})$
6:         $\boldsymbol{Q} = \boldsymbol{W}\boldsymbol{D}^{-1} : \mathbb{R}^{n\times n}$
7:         $\boldsymbol{Z} = \boldsymbol{X}\boldsymbol{Q} : \mathbb{R}^{d\times n}$
8:     until stop
9:     return ConnectedComponents($\{\boldsymbol{z}_i\}_{i=1}^n, \varepsilon$)
```

Algorithm 3 Mean-shift algorithm in iterative form.

```
1: function MeanShift($\boldsymbol{x}_1, ..., \boldsymbol{x}_n : \mathbb{R}^d$, $h$, $\varepsilon$)
2:     for $i = 1..n$ do
3:         $\boldsymbol{x}^{(1)} = \boldsymbol{x}_i$, $t = 0$
4:         repeat
5:             $t = t + 1$
6:             $\forall n : p(i \mid \boldsymbol{x}^{(t)}) = \frac{\exp(-\frac{1}{2}\left\|(\boldsymbol{x}^{(t)} - \boldsymbol{x}_i)/h\right\|^2)}{\sum_{j=1}^n \exp(-\frac{1}{2}\left\|(\boldsymbol{x}^{(t)} - \boldsymbol{x}_j)/h\right\|^2)}$
7:             $\boldsymbol{x}^{(t+1)} = \sum_{i=1}^n p(i \mid \boldsymbol{x}^{(t)})\boldsymbol{x}_i$
8:         until $\left\|\boldsymbol{x}^{(t+1)} - \boldsymbol{x}^{(t)}\right\| < tol$
9:         $\boldsymbol{z}_i = \boldsymbol{x}^{(t+1)}$
10:    return ConnectedComponents($\{\boldsymbol{z}_i\}_{i=1}^n, \varepsilon$)
```

3.5 Discussion

Problems

Performance

The algorithm inhabits several problems. Some of them are dependent on the domain and data, like the choice of bandwidth and kernel, others are regarding the performance.

Methods for speedup

4 Application

image segmentation

The main application for the mean shift clustering is image segmentation. Although the raw image data in the RGB color space can be used most problem domains benefit from a transformed feature space. A human perceives distances of colors differently than the RGB-colorspace indicates. For that reason the image is often transformed in a more suitable colorspace, for example the L*u*v*-colorspace.

Another aspect is the spatial coherence of the clusters in the image. Two pixels with the same color, but in widely different parts of the image would be put in the same cluster. That can be prevented by adding imagespace information to the pixels.

4.1 Experiments

In this section we apply the mean shift algorithm to different tasks. For this we implemented the mean shift algorithm in Matlab. The code for the method `mean_shift` can be found in Code Listing 1.

Basic clustering

basic clustering task

image segmentation – details

dataset description

Image segmentation

We perform an image segmentation task. The dataset is the "segmentation evaluation database" from Alpert et al. (2012). The dataset consists of 100 images with gray level and color sources. Also human segmentation results are provided.

For the experiment we apply several transformations before running the mean shift procedure. To transform the image data into a suitable feature space we implemented the method `image_transform`. The code can be found in Code Listing 4.

1. **Image rescaling**. This reduces the number of pixels and minimizes the run time.
2. **Color space transformation**. As discussed at the introduction of section 4, the RGB color space is not suitable for an euclidean distance metric. The CIE 1976 L*,u*,v* color space is build to approximate a perceptually uniform color space. Matlab provides the functionality to perform color space transformations.
3. **Adding spacial features**. Image segmentation benefits from connected patches, but without spacial features, there's no way to respect pixel distance. We add two new features representing the position of the pixel in the image.

In our experiment we rescale the image so that they have height of 50 pixels. That transformation is needed so that the total number of pixels is small enough for segmentation in reasonable time.

Next to the image rescaling we are also performing other transformations to an feature space which is more suitable for image segmentation. Our approach closely resembles the procedure in Comaniciu & Meer (2002). A Matlab function `image_transform` is provided in the Code Listing 4. The input images are provided in RGB color space. The problem with this color space is that the euclidean distance in RGB values do not translate to the visual distance perceived by humans. The CIE 1976 L*,u*,v* color space is build to approximate a perceptually uniform color space. Matlab provides the functionality to perform color space transformations. Image segmentation tasks also benefit from spacial

4.2 Evaluation and Comparison

To evaluate segmentation results is a difficult task. We are going to present the segmentation results themselves.

5 Summary

We introduced general kernel density estimates and presented popular choices for kernels. We classify the mean shift clustering algorithm by it's understanding of a cluster and derive the iteration step coming from the kernel density estimate. We present different applications of the algorithm, mainly classic clustering tasks and

References

Alpert, S., Galun, M., Brandt, A. & Basri, R. (2012), 'Image segmentation by probabilistic bottom-up aggregation and cue integration', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **34**(2), 315–326.

Comaniciu, D. & Meer, P. (2002), 'Mean shift: a robust approach toward feature space analysis', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(5), 603–619.

Comaniciu, D., Ramesh, V. & Meer, P. (2003), 'Kernel-based object tracking', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25**(5), 564–577.

Fukunaga, K. & Hostetler, L. (1975), 'The estimation of the gradient of a density function, with applications in pattern recognition', *IEEE Transactions on Information Theory* **21**(1), 32–40.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011), 'Scikit-learn: Machine learning in python', *Journal of Machine Learning Research* **12**, 2825–2830.

Wand, M. P. & Jones, M. C. (1995), *Kernel Smoothing*, Vol. 60 of *Monographs on Statistics and Applied Probability*, Springer US, Boston, MA and s.l.
URL: *http://dx.doi.org/10.1007/978-1-4899-4493-1*

A Code

```
function [A, C, T] = mean_shift(X, kernel, h, epsilon)
    % Performes the mean shift algorithm with the
    % specified kernel and bandwidth h.
    %
    % INPUTS:
    %          X - an d by n matrix of data points
    %     kernel - the kernel used. 'gaussian', 'uniform'
    %          h - the bandwidth used
    %    epsilon - the parameter for cluster pruning
    % OUTPUTS:
    %          A - an n by 1 vector of cluster indices
    %          C - an d by n matrix of cluster centroids
    %          T - an n by 1 vector of iterations needed
    %              for each data point
    tol = h * 1e-2; max_iter = 100;

    [d, n] = size(X);
    Z = zeros(d, n);
    T = zeros(n, 1);

    for i = 1:n
        x = X(:, i);
        for t = 1:max_iter
            u = sum(((X - x) / h).^2, 1);
            switch kernel
                case 'gaussian'
                    w = exp(-0.5 * u);
                case 'uniform'
                    w = u < 1;
            end

            x_next = X * w' / sum(w);

            if (norm(x_next - x) < tol)
                break
            end
            x = x_next;
        end
        Z(:, i) = x;
        T(i) = t;
    end

    [A, C] = connected_component(Z, epsilon);
end
```

Code Listing 1: Mean shift

```
function [A, C] = connected_component(X, epsilon)
    % Performes data pruning. An implicit neighborhood
    % graph is constructed in which x_i and x_j are
    % connected if ||x_i - x_j|| < epsilon. Then points
    % are randomly chosen and all its neighbors are
    % assigned to the same cluster.
    %
    % INPUTS:
    %         X - a d by n matrix with data points
    %   epsilon - the distance in which points are merged
    % OUTPUTS:
    %         A - a n by 1 vector with cluster indices
    %         C - a d by max(A) matrix with cluster
    %             representatives
    [d, n] = size(X);
    A = zeros(n, 1);
    C = zeros(d, n);
    % the number of connected components found yet
    c = 0;

    while any(A == 0)
        c = c + 1;
        % find i for which x_i has not yet been assigned
        i = find(~A, 1);
        x = X(:, i);
        C(:, c) = x;
        d = sum((X - x).^2, 1).^0.5;
        A(d < epsilon) = c;
    end
    C = C(:, 1:c);
end
```

Code Listing 2: Connected component

```
function h = estimate_bandwidth(X, quantile)
    % Bandwidth estimation for kernel density estimates.
    % The mean distance of each data point to its
    % (quantile * n)-nearest neighbor is used. A quantile
    % value of 0.5 means the median value of all pairwise
    % distances is used.
    %
    % INPUTS:
    %          X - a d by n matrix with data points
    %   quantile - the quantile used; must be in [0, 1]
    %              the default value is 0.3
    % OUTPUTS:
    %          h - the estimated bandwidth
    %
    % Reference:
    %   See: scikit-learn implementation: https://github.com/scikit-learn/scikit-learn/blob/7389dba/sklearn/cluster/mean_shift_.py#L31
    % Example:
    %   X = [randn(3, 20) (randn(3, 40) + [2;0;6])];
    %   estimate_bandwidth(X)
     if nargin < 2
         quantile = 0.3;
    end

    [~, n] = size(X);
    K = ceil(n * quantile);
    [~, D] = knnsearch(X', X', 'K', K, 'Distance','euclidean');
    h = mean(D(:, K));
end
```

Code Listing 3: Estimate bandwidth

```
function [X, n_rows, n_cols, upvpl_mean, upvpl_std] =
    image_transform(I)
    % Transform image to data matrix. It performs the
    % following steps:
    %   1. Convert rgb to 1976 CIE u'v'Y color space
    %      (also known as L*U*V* color space)
    %   2. Flatten image data into matrix
    %   3. Standardize luv coords to have zero mean
    %      and unit variance
    %   4. Add dimensions representing row/col
    %      coordinates with range [0, 1]
    %
    % INPUTS:
    %            I - an n_rows by n_cols by 3 image
    %                array in rgb color format
    % OUTPUTS:
    %            X - a 5 by n_rows * n_cols matrix
    %       n_rows - the number of image rows
    %       n_cols - the number of image columns
    %   upvpl_mean - a 3 by 1 vector. The mean vector
    %                of upvpl coordinates
    %    upvpl_std - a 3 by 1 vector. The std vector
    %                of upvpl coordindates
    %
    % Reference:
    %   See: https://en.wikipedia.org/wiki/CIELUV
    T = rgb2xyz(I);
    T = applycform(T, makecform('xyz2upvpl'));

    [n_rows, n_cols, ~] = size(T);
    n = n_rows * n_cols;
    X = reshape(T, n, 3)';

    upvpl_mean = mean(X, 2);
    upvpl_std = std(X, 1, 2);
    X(1, :) = (X(1, :) - upvpl_mean(1)) / upvpl_std(1);
    X(2, :) = (X(2, :) - upvpl_mean(2)) / upvpl_std(2);
    X(3, :) = (X(3, :) - upvpl_mean(3)) / upvpl_std(3);

    row_vals = repmat(linspace(0,1,n_rows), 1, n_cols);
    col_vals = repmat(linspace(0,1,n_cols), n_rows, 1);
    X(4, :) = reshape(row_vals, n, 1)';
    X(5, :) = reshape(col_vals, n, 1)';
end
```

Code Listing 4: Image transform

Statutory Declaration

I hereby declare, that I have written this seminar paper with the title **Kernel Density Estimates and Mean-Shift Clustering** by myself and have not used other sources without declaration.

Karlsruhe, January 21, 2019

Jonas Spinner