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 of 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 we are going to discuss the method of kernel density estimation and it's use in the mean shift clustering algorithm. Clustering is one of the main tasks in machine learning and the mean shift algorithm has many applications, mainly in computer vision tasks. The mean shift algorithm is an nonparametric approach which allows a wide range of data distributions without imposing any prior knowledge.

The content of this seminar paper is mainly based on Comaniciu & Meer (2002). First we introduce the kernel density estimate in section 2. In section 3 we derive the mean shift algorithm by estimating the density gradient. Then we finally state the mean shift algorithm. In section 4 we take a look at applications of the algorithm and describe our implementation in Matlab.

History

The mean shift algorithm was introduced in Fukunaga & Hostetler (1975) where also the term "mean shift" was established. Later 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 segmentation, image filtering and object tracking.

Notation

In this paper we use the following conventions. Bold symbols like $\boldsymbol{x}$ denote a vector or a vector valued function like $\boldsymbol{m}(\cdot)$. $\{\boldsymbol{x}_i\}_{i=1}^n$ denotes the set with n samples $\boldsymbol{x}_i$, indexed by i. Throughout the seminar paper d is the dimension of the samples. Upper case functions like $K(\cdot)$ denote a multivariate kernel function $\mathbb{R}^d \to \mathbb{R}$ and $k(\cdot)$ denotes a single variable function $[0, \infty]) \to [0, \infty)$.

2 Kernel Density Estimation

In this section we are going to introduce the kernel density estimate. It is the main concept behind the mean shift algorithm and will be used in its derivation.

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 distribution with the probability density function $f(\boldsymbol{x})$. The goal of density estimation is to estimate the $f(\boldsymbol{x})$.

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

$$K(\boldsymbol{x}) \geq 0 \quad \text{and} \quad \int_{\mathbb{R}^d} K(\boldsymbol{x}) d\boldsymbol{x} = 1. \tag{2}$$

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 to discuss the choices for kernel functions and the bandwidth in section 2.1.

The kernel density estimate can be interpreted as the sum of “bumps” centered on each data point. In Figure 1 the process of kernel density estimates is illustrated. The original density function is estimated based on five samples. For density estimation three methods are compared: a histogram, a kernel density estimate with a uniform kernel and a kernel density estimate with a gaussian kernel.

2.1 Popular Kernels

In this section we are going to present popular kernel functions and introduce the concept of radially symmetric kernels. This class of kernels is useful in the derivation of the mean shift algorithm.

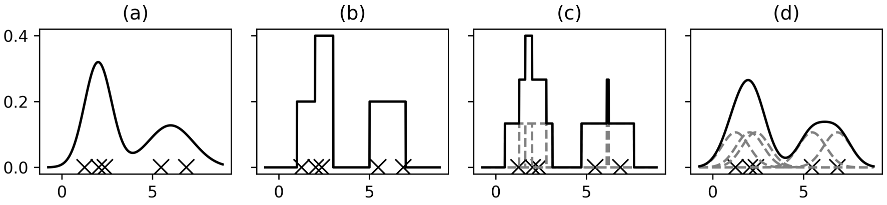

Figure 1: Samples, histogram and kernel density estimates. Author's illustration. (a) The probability density distribution $\frac{3}{5}\mathcal{N}\left(2, 0.75^2\right) + \frac{2}{5}\mathcal{N}\left(6, 1.25^2\right)$ and five samples drawn from it. (b) A histogram with bins $[i, i+1)$, $i \in \mathbb{Z}$. (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 $h = 0.75$, $K(x) = \frac{1}{h}\frac{1}{\sqrt{2\pi}}\exp(-\frac{1}{2}|\frac{x}{h}|^2)$.

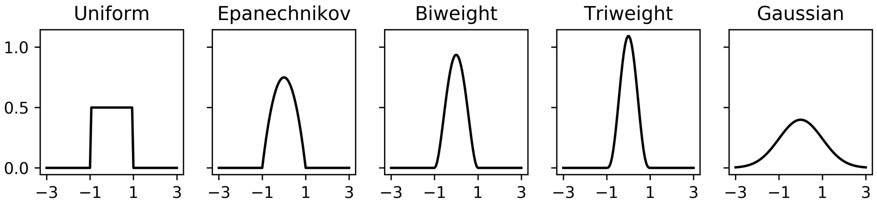

Figure 2: Popular kernels. Author's illustration. The function definitions are listed in Table 1. The first four kernels all have support $[-1, 1]$. The gaussian kernel has the support $(-\infty, \infty)$.

Radially symmetric kernels are kernel functions which can be formulated as

$$K(\boldsymbol{x}) = c_{k,d} k(\|\boldsymbol{x}\|^2), \tag{3}$$

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

The literature regarding mean shift clustering only uses a few distinct kernels. The most popular kernels for mean shift clustering are the uniform,the Epanechnikov and the gaussian kernel which are all radially symmetric kernels. We are going to shortly introduce them and their profile.

The **uniform kernel** is defined by being proportional to the indicator function of the unit hypersphere and can be written as

$$K_U(\boldsymbol{x}) = \begin{cases} \mathrm{vol}(S_d)^{-1} & \text{for } \|\boldsymbol{x}\| \leq 1 \\ 0 & \text{else} \end{cases}, \tag{4}$$

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

$$k_U(u) = \begin{cases} 1 & \text{for } 0 \leq u \leq 1 \\ 0 & \text{for } 1 < u. \end{cases} \tag{5}$$

The **Epanechnikov kernel** is defined as

$$K_E(\boldsymbol{x}) = \begin{cases} \frac{2+d}{2\mathrm{vol}(S_d)}(1 - \|\boldsymbol{x}\|^2) & \text{for } \|\boldsymbol{x}\| \leq 1 \\ 0 & \text{else} \end{cases} \tag{6}$$

and its profile k_E is

$$k_E(u) = \begin{cases} 1 - u & \text{for } 0 \leq u \leq 1 \\ 0 & \text{for } 1 < u. \end{cases} \tag{7}$$

The Epanechnikov kernel is an optimal kernel in the sense that it minimizes the quality measure AMISE[1].

The **gaussian kernel** is defined as

$$K_G(\boldsymbol{x}) = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}\|\boldsymbol{x}\|^2\right) \tag{8}$$

[1] AMISE stands for approximation of the mean square error between the density and the estimate integrated over the domain (Comaniciu & Meer 2002, p. 5).

Name	**Profile support**	$k(u)$	$-k'(u)$	$K(\boldsymbol{x})$
Uniform	$u \in [0,1]$	1	0	$\mathrm{vol}\left(S_d\right)^{-1}$
Epanechnikov	$u \in [0,1]$	$1-u$	1	$\frac{1}{2}\mathrm{vol}(S_d)^{-1}(d+2)\left(1-\lVert\boldsymbol{x}\rVert^2\right)$
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.

and its profile k_G is

$$k_G(u) = \exp\left(-\frac{1}{2}u\right). \tag{9}$$

The gaussian kernel is one of the few kernels which has a non compact support. That can pose a problem because every point $\boldsymbol{x} \in \mathbb{R}$ gets assigned a non zero weight $K_G(\boldsymbol{x})$. In practical applications the kernel is often truncated and the very small weights are ignored.

We present the kernels, their support and profile in Table 1 for future reference.

2.2 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 and many “rule of thumbs” are formulated, but the multivariate case lacks those. In (Comaniciu & Meer 2002, section 3.1) some general approaches for bandwidth selection are listed.

- 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 %.

More bandwidth selection techniques are proposed in the literature, but are not covered by us.

For our seminar paper we are implemented the K nearest neighbor approach. The Python library scikit-learn from Pedregosa et al. (2011) implements this approach as `estimate_bandwidth`. Our code can be found in Code Listing 3.

3 Mean Shift Clustering

In this section we are going to give a general overview of the clustering task and different clustering algorithms in section 3.1. Then we will look at density gradient estimation in section 3.2 and derive properties of the mean shift vector. Equipped with this perquisites we are formulate the mean shift algorithm in section 3.4.

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 $\mathcal{C} = \{C_j\}_{j=1}^K$.

3.1 Clustering algorithms

What does define a "good" clustering? One can try to minimize a global criterion. For example the K-Means algorithm tries to minimize the intra cluster variance $J = \sum_{j=1}^K \sum_{\boldsymbol{x}_i \in C_j} \|\boldsymbol{x}_i - \boldsymbol{m}_j\|^2$ and hierarchical clustering minimizes the intra-cluster min, max or average distances between the data points. 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. That can be the number and shapes of clusters, e.g. K-Means assumes the data is separable into K clusters.

Centroid based algorithms

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 means $\boldsymbol{m}_j = |C_j|^{-1} \sum_{\boldsymbol{x}_i \in C_j} \boldsymbol{x}_i$. A problem with this approach is that the cluster mean is not guaranteed to "look like" a typical data object, meaning that the centroid can lie in a environment with no real data points in its neighborhood.

"Low density centroids" can be avoided by defining the centroids to be the **modes of the probability density function**. That's the core idea 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 searches for the modes of the estimated density.

This approach has some nice benefits. The number of clusters is defined by the number of modes and clusters can have an arbitrary shape.

For the mode finding procedure we need to work with gradient of the density and find the points $\boldsymbol{x}$ where $\nabla f(\boldsymbol{x}) = 0$. In the next section we are take a look on how to approximate the density gradient.

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

$$\hat{f}(\boldsymbol{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right). \tag{10}$$

We estimate the density gradient $\nabla f(\boldsymbol{x})$ by the gradient of the density estimate:

$$\hat{\nabla} f(\boldsymbol{x}) \equiv \nabla \hat{f}(\boldsymbol{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} \nabla K\left(\frac{\boldsymbol{x} - \boldsymbol{x}_i}{h}\right). \tag{11}$$

For simplification we are going to concentrate on **radially symmetric kernels** that we have introduced in section 2.1. Recall that these kernels can be written as

$$K(\boldsymbol{x}) = c_{k,d} k(\|\boldsymbol{x}\|^2) \tag{12}$$

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.

Because we are going to use different kernel density estimates, we now 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

$$\hat{f}_{h,K}(\boldsymbol{x}) = \frac{c_{k,d}}{nh^d}\sum_{i=1}^{n} k\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right\|^2\right) \tag{13}$$

and its gradient as

$$\nabla\hat{f}_{h,K}(\boldsymbol{x}) = \frac{2c_{k,d}}{nh^{d+2}}\sum_{i=1}^{n}(\boldsymbol{x}-\boldsymbol{x}_i)k'\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right\|^2\right). \tag{14}$$

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

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

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

$$\boldsymbol{m}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n}\boldsymbol{x}_i g\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right\|^2\right)} - \boldsymbol{x}. \tag{16}$$

We introduced an estimate for the density gradient and 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 15 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 $G(\boldsymbol{x}) = c_{g,d}g(\|\boldsymbol{x}\|^2)$

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

$$= \frac{2c_{k,d}}{h^2 c_{g,d}}\hat{f}_{h,G}(\boldsymbol{x}). \tag{18}$$

With this formulation we can rewrite the density gradient estimate $\nabla\hat{f}_{h,K}(\boldsymbol{x})$ into

$$\nabla\hat{f}_{h,K}(\boldsymbol{x}) = \frac{2c_{k,d}}{h^2 c_{g,d}}\hat{f}_{h,G}(\boldsymbol{x})\boldsymbol{m}(\boldsymbol{x}). \tag{19}$$

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

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

We can observe 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.

Shadows of kernels

For the kernels $G(\boldsymbol{x}) \propto g(\|\boldsymbol{x}\|^2)$ and $K(\boldsymbol{x}) \propto k(\|\boldsymbol{x}\|^2)$, if $-k'(u) \propto g(u)$ then we call the kernel K a **shadow** of kernel G.

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

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

3.4 The mean shift algorithm

With the previously introduced concepts the mean shift algorithm can be quickly stated. Recall that the mean shift vector is

$$\boldsymbol{m}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n}\boldsymbol{x}_i g\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right\|^2\right)} - \boldsymbol{x}. \tag{21}$$

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

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \boldsymbol{m}\left(\boldsymbol{x}^{(t)}\right), \qquad t = 1, 2, ... \tag{22}$$

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 for which $\boldsymbol{z}_i = \boldsymbol{z}_j$ 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 mean shift algorithm is formulated in pseudo code as Algorithm 1.

Algorithm 1 The mean shift algorithm with a gaussian kernel.

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

Convergence

For 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. Each iterate $\boldsymbol{x}^{(t)}$ converges after finitely many steps.

The mean shift trajectories are illustrated in Figure 3. The dataset is a mixture of two gaussian distributions and the paths $\boldsymbol{x}^{(t)}$ are plotted. One can see that the trajectories of the data points meet at the two modes.

3.5 Discussion

The mean shift algorithm has several advantages and disadvantages compared to other clustering algorithms. We are going to present possible solutions for some of them. This section is mainly based on Carreira-Perpiñán (2015).

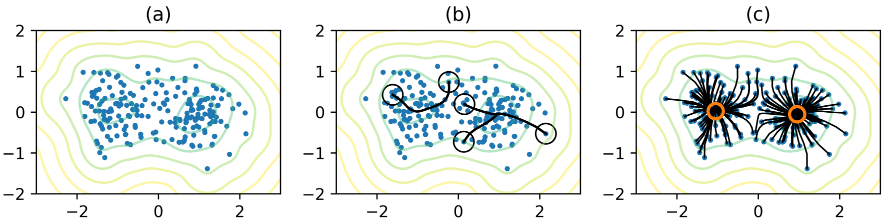

Figure 3: Mean shift trajectories. Author's illustration. (a) The dataset is sampled from $\frac{1}{2}\mathcal{N}\left((1,0)^T, 0.5^2\right) + \frac{1}{2}\mathcal{N}\left((-1,0)^T, 0.5^2\right)$. The contour lines of $\log \hat{f}(\boldsymbol{x})$ are plotted. (b) The trajectories $\boldsymbol{x}^{(t)}$ for selected data points are plotted. A gaussian kernel was used with the bandwidth $h = 0.25$. The circles are representing the bandwidth. (c) Trajectories for all data points. The cluster centroids are marked with orange circles.

Advantages

- One of the main benefits is that the algorithm makes no prior assumptions about the number and shape of the clusters. This allows complex clusters with non convex shapes.
- Outliers do not affect the clustering of the other data points. As outliers lie in low density regions their weight assigned by the kernel is either zero or very small.
- The algorithm only has one parameter, the bandwidth. The bandwidth has a easily understood interpretation and determines the resulting cluster uniquely. That can be used to formulate functions dependent on the bandwidth, e.g. some clustering criteria (see section 2.2).
- The cluster centroids can be interpreted as the modes of the estimated density function. That has the benefit that, ignoring the outliers, the centroids are "typical" data objects in that they are from a locally high density environment.

Disadvantages

The algorithm also inhibits several problems, mainly the scaling properties in the size and dimension of the dataset and choice of bandwidth.

- The algorithm has complexity $\mathcal{O}(Tn^2)$, with T being the number of iterations. This is especially problematic for large datasets.
- A problem originating from the underlying multivariate kernel density estimation is a poor scaling with the dimension of the feature space d.

- As we have already discussed in section 2.2, it is not easy to find a fitting bandwidth, especially in the multivariate case.
- Although the modes are local maxima, it is not guaranteed that they lie in a large density environment. This can easily fixed by a post processing step in which low density modes are classified as outliers.
- In some use cases it is useful to specify the number of clusters a priori. This is only possible by searching for the desired number of clusters through changing the bandwidth.

Methods for speedup

There are several ways to speed up the algorithm for practical usage.

- **Parallelize iteration**. The iterations of the data points are independent and can easily be performed in parallel.
- **Sample the dataset**. Instead of performing the mean shift algorithm on all data points, only perfom it on a smaller subsample. Then assign the remaining points to their nearest cluster.
- **Bucketing the dataset**. Before working on the dataset, put the data points into buckets and perform the algorithm for the buckets only.
- **Spacial data structures**. For kernels with compact support only the data points in the neighborhood are relevant. Spacial data structures can be used to find those relevant points.
- **Trajectory merging**. When two trajectories get close to each other it is likely that they converge to the same point. By merging the trajectories the number of total iterations can be minimized.

4 Application

The main application for the mean shift clustering is image segmentation. In Comaniciu & Meer (2002) and Comaniciu et al. (2003) the use of the algorithm for image segmentation, image filtering and object tracking is introduced and popularized.

4.1 Experiments

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

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. Our approach closely resembles the procedure in Comaniciu & Meer (2002). 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. The transformation steps are:

1. **Image rescaling**. This reduces the number of pixels in order to reduce the run time of the algorithm.
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. **Feature standardization**. We standardize the color features by their mean and standard deviation.
4. **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.

4.2 Evaluation

The mean shift algorithm performs mostly well on the "segmentation evaluation database" dataset. An overview of the results can be seen in Figure 4. The algorithm performs well on large segments with nearly uniform color.

We could also see that the chosen bandwidth has an significant effect on the clustering result. In Figure 5 two images are segmented with a wide range of bandwidth values. Low bandwidth values result in a large number of segments and chaotic segmentation result. High values result in only a few segments.

Automatically choosing the right bandwidth value is not an easy task. The experiment values were chosen by hand.

Although the algorithm has performed quite well on some input images, others do not result in a satisfying segmentation. In Figure 6 we present some “bad” segmentation results. Some of these images have a busy texture and the segments are distributed across the image, e.g. the frog at position (1, 3) or the owl at position (2, 2). The plane at position (1, 4) is easily visually segmented, but the algorithm struggles to find a good segmentation.

To summarize the experiment results, the mean shift algorithm as stated in Code Listing 1 and with the image transformation in Code Listing 4 performs well on a wide range of test images. The results are highly dependent on the chosen bandwidth and finding a good bandwidth is hard. Also there are some images for which the algorithm fails to provide a good segmentation.

5 Summary

We introduced general kernel density estimates and presented popular choices for kernel functions. We derived the theory behind the mean shift algorithm and define its cluster definition. In the section 4 we implemented the mean shift algorithm in Matlab and applied it to a image segmentation task.

In conclusion, the mean shift algorithm is an interesting density based clustering algorithm which is unique in its cluster definition. It has scaling problems in the dimension of the feature space. In theory the number of samples is a problem for the run time, but several speed up techniques can be implemented to effectively counter that effect. The algorithm has found its main applications in computer vision tasks but can be used as a general purpose clustering algorithm for lower dimensional problems.

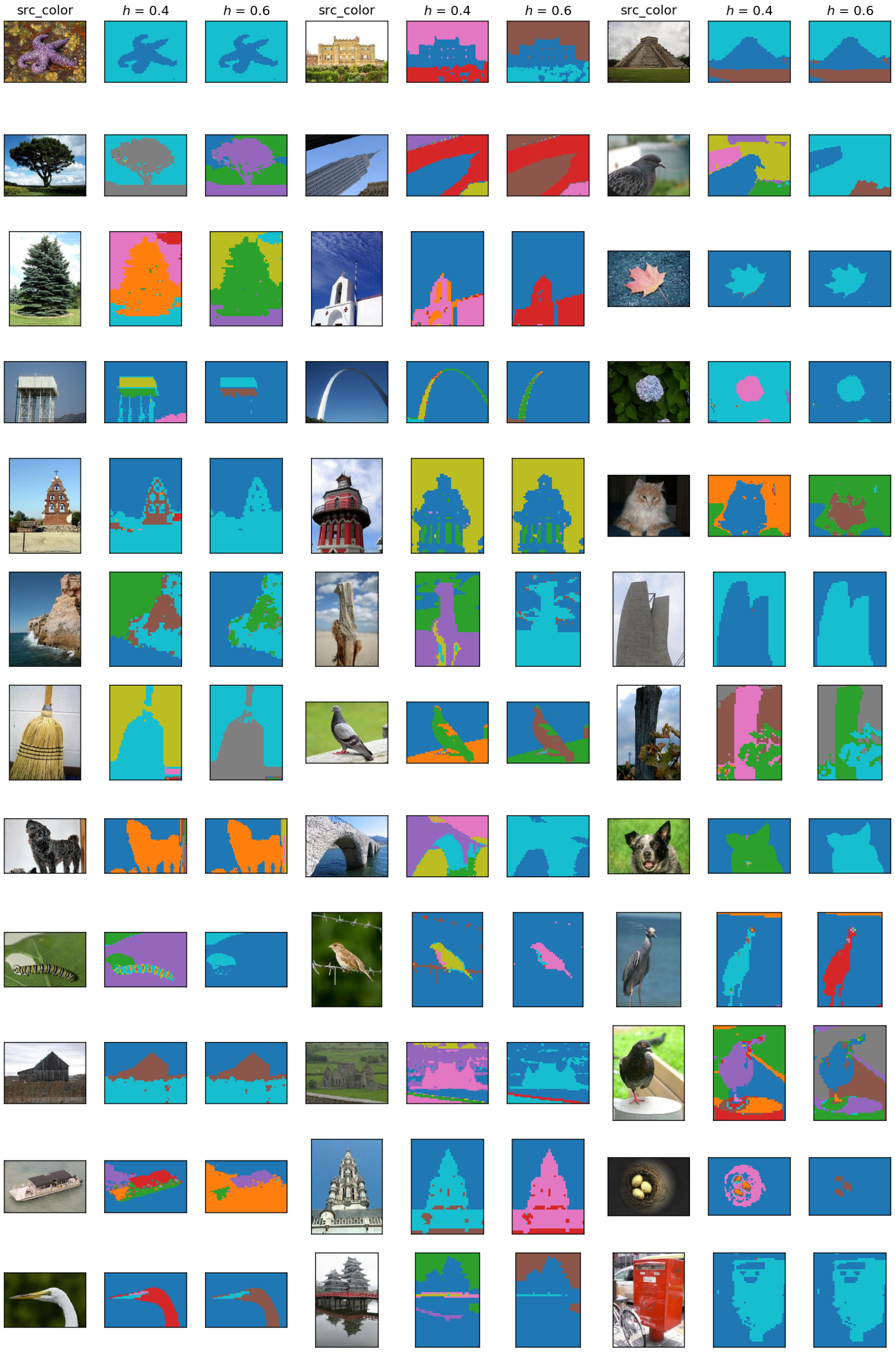

Figure 4: Image segmentation results. Author's illustration. Selected images from the "segmentation evaluation database" dataset. The segmentation is performed with a gaussian kernel with bandwidth 0.4 and 0.6.

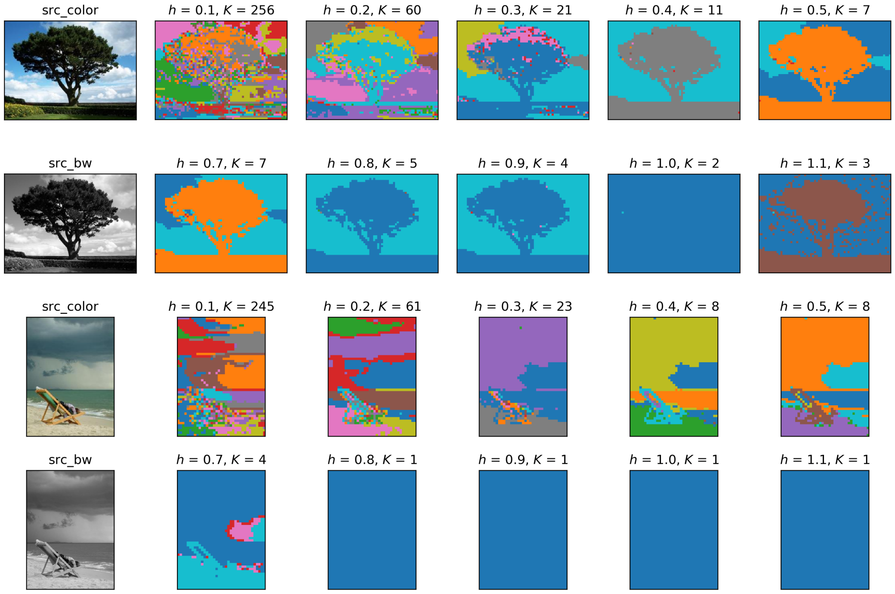

Figure 5: Bandwidth effects. Author's illustration. Two images from the "segmentation evaluation database" dataset are segmented with the mean shift algorithm and a gaussian kernel. The color and gray scale sources are presented. Then the clustering results with varying bandwidths h from 0.1 to 1.1 are plotted. The number of clusters are noted as K.

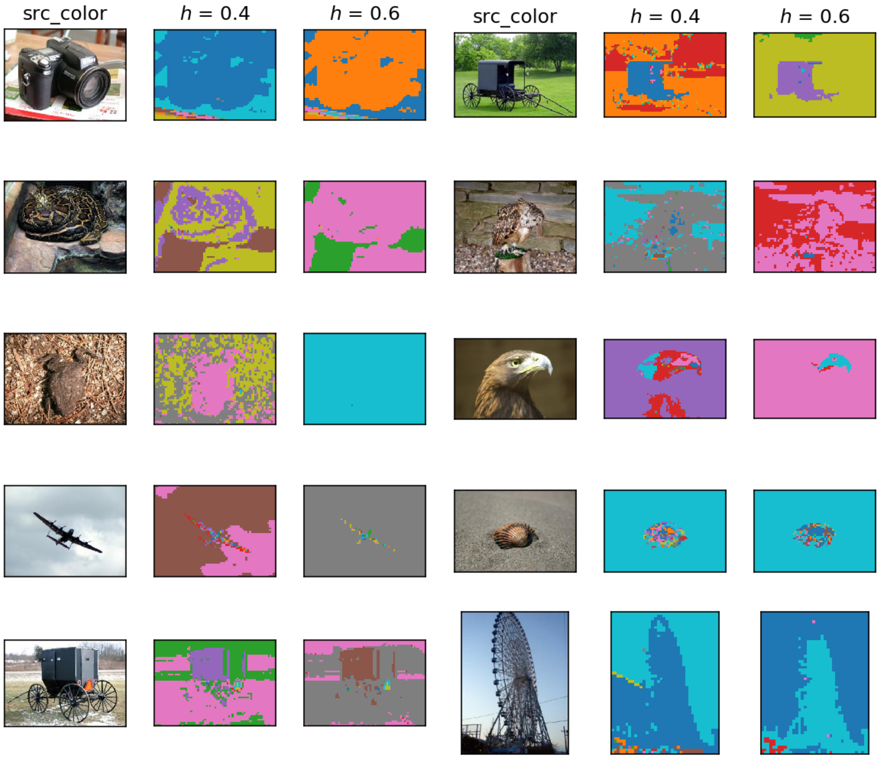

Figure 6: Bad image segmentation examples. Author's illustration. Selected images from the "segmentation evaluation database" dataset. The segmentation is performed with a gaussian kernel with bandwidth 0.4 and 0.6. These images are not well segmented by the mean shift algorithm.

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.

Carreira-Perpiñán, M. Á. (2015), 'A review of mean-shift algorithms for clustering'.
URL: *https://arxiv.org/abs/1503.00687v1*

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.

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

```matlab
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, 0, 2);
    X(1:3,:) = (X(1:3,:) - upvpl_mean) ./ upvpl_std;

    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