# Introduction To Mean Shift Algorithm

April 1, 2010 by Saravanan Thirumuruganathan

Its been quite some time since I wrote a Data Mining post . Today, I intend to post on Mean Shift – a really cool but not very well known algorithm. The basic idea is quite simple but the results are amazing. It was invented long back in 1975 but was not widely used till two papers applied the algorithm to Computer Vision.

I learned this algorithm in my Advanced Data Mining course and I wrote the lecture notes on it. So here I am trying to convert my lecture notes to a post. I have tried to simplify it – but this post is quite involved than the other posts.

It is quite sad that there exists no good post on such a good algorithm. While writing my lecture notes, I struggled a lot for good resources🙂 . The 3 "classic" papers on Mean Shift are quite hard to understand. Most of the other resources are usually from Computer Vision courses where Mean Shift is taught lightly as yet another technique for vision tasks (like segmentation) and contains only the main intuition and the formulas.

As a disclaimer, there might be errors in my exposition – so if you find anything wrong please let me know and I will fix it. You can always check out the reference for more details. I have not included any graphics in it but you can check the ppt given in the references for an animation of Mean Shift.

## Introduction

Mean Shift is a powerful and versatile non parametric iterative algorithm that can be used for lot of purposes like finding modes, clustering etc. Mean Shift was introduced in Fukunaga and Hostetler [1] and has been extended to be applicable in other fields like Computer Vision.This document will provide a discussion of Mean Shift , prove its convergence and slightly discuss its important applications.

# Intuitive Idea of Mean Shift

This section provides an intuitive idea of Mean shift and the later sections will expand the idea. Mean shift considers feature space as a empirical probability density function. If the input is a set of points then Mean shift considers them as sampled from the underlying probability density function. If dense regions (or clusters) are present in the feature space , then they correspond to the mode (or local maxima) of the probability density function. We can also identify clusters associated with the given mode using Mean Shift.

For each data point, Mean shift associates it with the nearby peak of the dataset's probability density function. For each data point, Mean shift defines a window around it and computes the mean of the data point . Then it shifts the center of the window to the mean and repeats the algorithm till it converges. After each iteration, we can consider that the window shifts to a more denser region of the dataset.

At the high level, we can specify Mean Shift as follows :
1. Fix a window around each data point.
2. Compute the mean of data within the window.
3. Shift the window to the mean and repeat till convergence.

# Preliminaries

## Kernels :

A kernel is a function that satisfies the following requirements :

1. $\int_{R^d} \phi(x) = 1$

2. $\phi(x) \geq 0$

Some examples of kernels include :

1. Rectangular $\phi(x) = \begin{cases} 1 & a \leq x \leq b \\ 0 & else \end{cases}$

2. Gaussian $\phi(x) = e^{-\frac{x^2}{2\sigma^2}}$

3. Epanechnikov $\phi(x) = \begin{cases} \frac{3}{4}(1 - x^2) & if\ |x| \leq 1 \\ 0 & else \end{cases}$

**Kernel Density Estimation**

Kernel density estimation is a non parametric way to estimate the density function of a random variable. This is usually called as the Parzen window technique. Given a kernel K, bandwidth parameter h , Kernel density estimator for a given set of d-dimensional points is

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

## Gradient Ascent Nature of Mean Shift

Mean shift can be considered to based on Gradient ascent on the density contour. The generic formula for gradient ascent is ,

$$x_1 = x_0 + \eta f'(x_0)$$

Applying it to kernel density estimator,

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

$$\nabla \hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K'\left(\frac{x - x_i}{h}\right)$$

Setting it to 0 we get,

$$\sum_{i=1}^{n} K'\left(\frac{x - x_i}{h}\right) \overrightarrow{x} = \sum_{i=1}^{n} K'\left(\frac{x - x_i}{h}\right) \overrightarrow{x_i}$$

Finally , we get

$$\overrightarrow{x} = \frac{\sum_{i=1}^{n} K'\left(\frac{x - x_i}{h}\right) \overrightarrow{x_i}}{\sum_{i=1}^{n} K'\left(\frac{x - x_i}{h}\right)}$$

## Mean Shift

As explained above, Mean shift treats the points the feature space as an probability density function . Dense regions in feature space corresponds to local maxima or modes. So for each data point, we perform gradient ascent on the local estimated density until convergence. The stationary points obtained

via gradient ascent represent the modes of the density function. All points associated with the same stationary point belong to the same cluster.

Assuming $g(x) = -K'(x)$, we have

$$m(x) = \frac{\sum_{i=1}^{n} g\left(\frac{x-x_i}{h}\right) x_i}{\sum_{i=1}^{n} g\left(\frac{x-x_i}{h}\right)} - x$$

The quantity $m(x)$ is called as the mean shift. So mean shift procedure can be summarized as : For each point $x_i$

1. Compute mean shift vector $m(x_i^t)$

2. Move the density estimation window by $m(x_i^t)$

3. Repeat till convergence

Using a Gaussian kernel as an example,

1. $y_i^0 = x_i$
2. $y_i^{t+1} = \frac{\sum_{i=1}^{n} x_j e^{\frac{-|y_i^t - x_j|^2}{h^2}}}{\sum_{i=1}^{n} e^{\frac{-|y_i^t - x_j|^2}{h^2}}}$

## Proof Of Convergence

Using the kernel profile,

$$y^{t+1} = \frac{\sum_{i=1}^{n} x_i k(||\frac{y^t - x_i}{h}||^2)}{\sum_{i=1}^{n} k(||\frac{y^t - x_i}{h}||^2)}$$

To prove the convergence , we have to prove that $f(y^{t+1}) \geq f(y^t)$

$$f(y^{t+1}) - f(y^t) = \sum_{i=1}^{n} k(||\frac{y^{t+1} - x_i}{h}||^2) - \sum_{i=1}^{n} k(||\frac{y^t - x_i}{h}||^2)$$

But since the kernel is a convex function we have ,

$$k(y^{t+1}) - k(y^t) \geq k'(y^t)(y^{t+1} - y^t)$$

Using it ,

$$f(y^{t+1}) - f(y^t) \geq \sum_{i=1}^{n} k'(||\frac{y^t - x_i}{h}||^2)(||\frac{y^{t+1} - x_i}{h}||^2 - ||\frac{y^t - x_i}{h}||^2)$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} k'(||\frac{y^t - x_i}{h}||^2)(y^{(t+1)^2} - 2y^{t+1}x_i + x_i^2 - (y^{t^2} - 2y^t x_i + x_i^2))$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} k'(||\frac{y^t - x_i}{h}||^2)(y^{(t+1)^2} - y^{t^2} - 2(y^{t+1} - y^t)^T x_i)$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} k'(||\frac{y^t - x_i}{h}||^2)(y^{(t+1)^2} - y^{t^2} - 2(y^{t+1} - y^t)^T y^{t+1})$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} k'(||\frac{y^t - x_i}{h}||^2)(y^{(t+1)^2} - y^{t^2} - 2(y^{(t+1)^2} - y^t y^{t+1}))$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} k'(||\frac{y^t - x_i}{h}||^2)(y^{(t+1)^2} - y^{t^2} - 2y^{(t+1)^2} + 2y^t y^{t+1})$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} k'(||\frac{y^t - x_i}{h}||^2)(-y^{(t+1)^2} - y^{t^2} + 2y^t y^{t+1})$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} k'(||\frac{y^t - x_i}{h}||^2)(-1)(y^{(t+1)^2} + y^{t^2} - 2y^t y^{t+1})$$

$$= \frac{1}{h^2}\sum_{i=1}^{n} -k'(||\frac{y^t - x_i}{h}||^2)(||y^{t+1} - y^t||^2)$$

$$\geq 0$$

Thus we have proven that the sequence $\{f(j)\}_{j=1,2...}$is convergent. The second part of the proof in [2] which tries to prove the sequence $\{y_j\}_{j=1,2,...}$ is convergent is wrong.

## Improvements to Classic Mean Shift Algorithm

The classic mean shift algorithm is time intensive. The time complexity of it is given by $O(Tn^2)$ where $T$ is the number of iterations and $n$ is the number of data points in the data set. Many improvements have been made to the mean shift algorithm to make it converge faster.

One of them is the adaptive Mean Shift where you let the bandwidth parameter vary for each data point. Here, the $h$ parameter is calculated using kNN algorithm. If $x_{i,k}$is the k-nearest neighbor of $x_i$ then the bandwidth is calculated as

$$h_i = ||x_i - x_{i,k}||$$

Here we use $L_1$or $L_2$ norm to find the bandwidth.

An alternate way to speed up convergence is to alter the data points during the course of Mean Shift. Again using a Gaussian kernel as an example,

1. $y_i^0 = x_i$
2. $y_i^{t+1} = \frac{\sum_{i=1}^{n} x_j e^{\frac{-|y_i^t - x_j|^2}{h^2}}}{\sum_{i=1}^{n} e^{\frac{-|y_i^t - x_j|^2}{h^2}}}$
3. $x_i = y_i^{t+1}$

# Other Issues

1. Even though mean shift is a non parametric algorithm , it does require the bandwidth parameter h to be tuned. We can use kNN to find out the bandwidth. The choice of bandwidth in influences convergence rate and the number of clusters.
2. Choice of bandwidth parameter h is critical. A large h might result in incorrect clustering and might merge distinct clusters. A very small h might result in too many clusters.

3. When using kNN to determining h, the choice of k influences the value of h. For good results, k has to increase when the dimension of the data increases.
4. Mean shift might not work well in higher dimensions. In higher dimensions , the number of local maxima is pretty high and it might converge to a local optima soon.
5. Epanechnikov kernel has a clear cutoff and is optimal in bias-variance tradeoff.

# Applications of Mean Shift

Mean shift is a versatile algorithm that has found a lot of practical applications – especially in the computer vision field. In the computer vision, the dimensions are usually low (e.g. the color profile of the image). Hence mean shift is used to perform lot of common tasks in vision.

**Clustering**

The most important application is using Mean Shift for clustering. The fact that Mean Shift does not make assumptions about the number of clusters or the shape of the cluster makes it ideal for handling clusters of arbitrary shape and number.

Although, Mean Shift is primarily a mode finding algorithm , we can find clusters using it. The stationary points obtained via gradient ascent represent the modes of the density function. All points associated with the same stationary point belong to the same cluster.

An alternate way is to use the concept of Basin of Attraction. Informally, the set of points that converge to the same mode forms the basin of attraction for that mode. All the points in the same basin of attraction are associated with the same cluster. The number of clusters is obtained by the number of modes.

**Computer Vision Applications**

Mean Shift is used in multiple tasks in Computer Vision like segmentation, tracking, discontinuity preserving smoothing etc. For more details see [2],[8].

# Comparison with K-Means

Note : I have discussed K-Means at **K-Means Clustering Algorithm** (https://saravananthirumuruganathan.wordpress.com/2010/01/27/k-means-clustering-algorithm/). You can use it to brush it up if you want.

K-Means is one of most popular clustering algorithms. It is simple,fast and efficient. We can compare Mean Shift with K-Means on number of parameters.

One of the most important difference is that K-means makes two broad assumptions – the number of clusters is already known and the clusters are shaped spherically (or elliptically). Mean shift , being a non parametric algorithm, does not assume anything about number of clusters. The number of modes give the number of clusters. Also, since it is based on density estimation, it can handle arbitrarily shaped clusters.

K-means is very sensitive to initializations. A wrong initialization can delay convergence or some times even result in wrong clusters. Mean shift is fairly robust to initializations. Typically, mean shift is run for each point or some times points are selected uniformly from the feature space [2] . Similarly, K-means is sensitive to outliers but Mean Shift is not very sensitive.

K-means is fast and has a time complexity $O(knT)$ where k is the number of clusters, n is the number of points and T is the number of iterations. Classic mean shift is computationally expensive with a time complexity $O(Tn^2)$.

Mean shift is sensitive to the selection of bandwidth, $h$. A small $h$ can slow down the convergence. A large $h$ can speed up convergence but might merge two modes. But still, there are many techniques to determine $h$ reasonably well.

**Update [30 Apr 2010] :** I did not expect this reasonably technical post to become very popular, yet it did ! Some of the people who read it asked for a sample source code. I did write one in Matlab which randomly generates some points according to several gaussian distribution and the clusters using Mean Shift . It implements both the basic algorithm and also the adaptive algorithm. You can download my **Mean Shift code here** (https://code.google.com/p/saravanant/source/browse/trunk/datamining/meanshift/meanshift.m). Comments are as always welcome !

# References

1. Fukunaga and Hostetler, "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition", IEEE Transactions on Information Theory vol 21 , pp 32-40 ,1975
2. Dorin Comaniciu and Peter Meer, Mean Shift : A Robust approach towards feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence vol 24 No 5 May 2002.
3. Yizong Cheng , Mean Shift, Mode Seeking, and Clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence vol 17 No 8 Aug 1995.
4. Mean Shift Clustering by Konstantinos G. Derpanis
5. Chris Ding Lectures CSE 6339 Spring 2010.
6. Dijun Luo's presentation slides.
7. cs.nyu.edu/~fergus/teaching/vision/12_segmentation.ppt

8. Dorin Comaniciu, Visvanathan Ramesh and Peter Meer, Kernel-Based Object Tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence vol 25 No 5 May 2003.
9. Dorin Comaniciu, Visvanathan Ramesh and Peter Meer, The Variable Bandwidth Mean Shift and Data-Driven Scale Selection, ICCV 2001.




Posted in Data Mining, machine learning | Tagged clustering, computer vision, convergence, kmeans, matlab, mean shift | 88 Comments

# 88 Responses

**Koustubh Sinkar** *on May 5, 2010 at 1:51 am* | *Reply*
thank u for demystifying meanshift filtering

> **Saravanan Thirumuruganathan** *on May 5, 2010 at 7:31 pm* | *Reply*
> Hello Koustubh , Glad that you found the article useful !

**jacouille** *on May 13, 2010 at 1:33 pm* | *Reply*
Hi Saravanan !
thanks for this algorithm ! Ive already tested it successfully with my own vectors as input.
Im tring to change it a little bit in order to make image segmentation (with RGB images if possible).
Do you have any clue to modify the function sqdist ?

++

**jacouille** *on May 13, 2010 at 2:58 pm* | *Reply*
hi Saravanan!
thanks for the code!
would you give me a clue to extent this code to images ?
thanks tons!!

> **Saravanan Thirumuruganathan** *on May 15, 2010 at 1:10 am* | *Reply*
> Hello jacouille,
>
> Looks like wordpress categorized your comment as a spam and hence the delay in the response. I do not have much exposure to computer vision and that was the reason I did not emphasize that aspect of mean shift – You can check this reference for more details on applying it to images – Dorin Comaniciu and Peter Meer, Mean Shift : A Robust approach towards feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence vol 24 No 5 May 2002.
>
> You can also check this pdf : pages.cs.wisc.edu/~dyer/cs766/hw/hw2/hw2.pdf and this zip file for more details : http://www.cs.ucf.edu/vision/public_html/source.html

**jacouille** *on May 17, 2010 at 8:55 am* | *Reply*
thanks Saravanan !

**John** *on May 25, 2010 at 11:05 am* | *Reply*
Hi Saravanan,

Thankyou very much for the tutorial.

I found it very useful!

I have a question. How can you replace -'k(x) with g(x).

I am confused when you give an example using the gaussian kernel, do you not have to use the derivative of the gaussian kernel?

Any help would be appreciated.

Thanks,

John

**Saravanan Thirumuruganathan** *on May 25, 2010 at 1:02 pm | Reply*

Hello John,

Replacing -K'(x) with g(x) is just a notational convenience and nothing else.

Wrt the gaussian kernel , the basic idea is that derivative of e^x is again e^x . Again some paper use the L1 norm and many other use L2 norm. In the post , I obtained the derivative of the kernel and then use the L2 norm on the result. That is the reason it has power of two on it.

You can refer either Meer or Cheng et al for more details.

**John** *on May 26, 2010 at 6:20 am*

Saravanan,

Thankyou for the quick response.

I think iam missing something obivous. I understand that the derivative of e^x is again e^x.

However, the derivative of a gaussian is not e^x. Rather it is -2*e^(-x^2)*x.
This is where iam getting confused.

thanks again for the help.

John

**Marcel** *on July 28, 2012 at 1:31 pm | Reply*

Hi John,

I am confused with this, too. The derivative of the gaussian kernel is -2*e^(-x^2)*x, which gives us another weight compared to the gaussian kernel.

Did you get the point meanwhile? Could you explain? Or can anyone else?

Marcel

**Mariusz** *on August 29, 2012 at 5:22 pm*

Hi!
There's an error in the article. The g(x) function is wrongly definied. It shoud be this way: g(x) = -k'(x), where k is kernel profile (not a kernel K). The profile for the Gaussian kernel is exp(-(1/2) * x)) and then the formula for calculating a next point in the mean shift algorithm using Gaussian kernel is correct. More info in the paper [2].

**Ridha** *on May 27, 2010 at 5:02 am* | *Reply*

please, explain to me how to run this code???which one the main program???
thanks

**Saravanan Thirumuruganathan** *on May 27, 2010 at 10:24 pm* | *Reply*

Hello,

The way to call it is meanshift(numDimensions,useKNNToGetH) . Dimensions is the feature vector's dimension and useKNNToGetH is a boolean – If true it uses adaptive mean shift (see above) else uses the classical mean shift.

A sample invocation is
meanshift(2,false);
meanshift(2,true);

**SC Kim** *on September 2, 2010 at 2:18 am* | *Reply*

would you like to let me know where can I find paper or documents containing below "Mean shift might not work well in higher dimensions. In higher dimensions , the number of local maxima is pretty high and it might converge to a local optima soon." ?

**Saravanan Thirumuruganathan** *on September 2, 2010 at 10:15 am* | *Reply*

Kim,

For any optimization problem, the number of local maxima increases as dimensions increases. There are two scenarios : If the bandwidth is small, the update per iteration is small and hence converges to the nearest local maxima. If the bandwidth is very large it will converge to a single point which probabilistically may not be the global maxima. Basic details can be found in Peter Meer's paper.

**dinesh** *on November 22, 2010 at 4:30 am* | *Reply*

Thanks for the clear explanation of the concept.

**Burak** *on December 24, 2010 at 8:24 am* | *Reply*

About this line of the code clusterCentroids(closestCentroid,:) = 0.5 * (curDataPoint + clusterCentroids(closestCentroid,:)); I have a question. The cluster centroids change in each data is processsed. But, you take average of the recent data point that is inside of that cluster

and previous cluster centroid. However, I think we should find the average of all cluster centroids. If we call the cluster centroids in time by Xi, for example (X1+X2+X3)/3 is not equal to 1/2* [(X1+X2)/2+X3] which is cluster centroid computed in your code. Am I right?

**Saravanan Thirumuruganathan** *on December 27, 2010 at 1:57 am* | *Reply*

Burak,

To be honest, I do not remember exactly why I alter centroids in each step. If my memory is correct, it helps to speed up the convergence. Also, if one of the point is slightly far away from the "actual" centroid, this helps to bring it closer to the actual centroid faster. And you are right in that what I do is not the averaging that happens in other clustering algorithms like K-means .

**nadiafezai@yahoo.fr** *on December 30, 2010 at 8:06 am* | *Reply*

please, this code is available for segmentation of images TEP?
AND this code containt the version adaptative of mean shift(variable bandwith )?
please , explain me
thanks

**Saravanan Thirumuruganathan** *on December 30, 2010 at 10:18 am* | *Reply*

Nadia,

I do not have code for image segmentation. You can use the second parameter to force it to be adaptive. if the useKNN variable is true, it becomes adaptive mean shift.

**Preben** *on February 1, 2011 at 6:38 am* | *Reply*

Hi Saravanan

I've read your post and is a bit confused.
In the section "Gradient Ascent Nature of Mean Shift" you write the generic formular for gradient ascent:
x_{1}=x_{0}+\eta f'(x_{0})
which is fine…
Then you write, that you apply this to the kernel density estimator. Ok, so you differentiate (gradient) the function \hat{f} and get:
\bigtriangledown{\displaystyle \hat{f}(x)=\frac{1}{nh^{d}}\sum_{i=1}^{n}K'\left(\frac{x-x_{i}}{h}\right)}

Then you write "Setting it to 0 we get", but what are you exactly setting to zero, and how do you get the following formula?
{\displaystyle \sum_{i=1}^{n}K'\left(\frac{x-x_{i}}{h}\right)\overrightarrow{x}=\sum_{i=1}^{n}K'\left(\frac{x-x_{i}}{h}\right)\overrightarrow{x_{i}}}

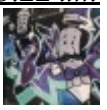
**Danny** *on March 1, 2011 at 6:55 pm* | *Reply*

I'm wondering about this exact question too.

Could someone shed some light on it?

Thanks,
Danny

**Raingo** *on March 27, 2011 at 3:22 am*



Hello Danny,

maybe I can help you with the question.

The question is originated from the previous expression: \bigtriangledown{\displaystyle \hat{f}(x)=\frac{1}{nh^{d}}\sum_{i=1}^{n}K'\left(\frac{x-x_{i}}{h}\right)}

which is fault as a matter of fact.

The correct form of the derivative of \hat{f}(x) is shown below:

\bigtriangledown{\displaystyle \hat{f}(x)=\frac{1}{nh^{d}}\sum_{i=1}^{n}K'\left(\frac{x-x_{i}}{h}\right) \left( 2*x-2*x_i \right)}

**weixi** *on February 7, 2011 at 2:01 pm* | *Reply*

Hi Saravanan,

First of all, thank you for this post and the Matlab code.

I have a question with regard to your code. The following lines seem to be inconsistent with a published paper:

```
kernelDist = exp(-euclideanDist ./ (bandwidth^2));
numerator = kernelDist * origDataPoints;
denominator = sum(kernelDist);
```

In the paper "Mean Shift Based Clustering in High Dimensions: A Texture Classification Example", which is last paper published by Peter Meer's lab on the subject, the formula (5) used an additional weight 1/h_i^(d+2). I do not understand how you can omit this additional weight, especially for the KNN case, in which h_i are different for each x_i and therefore cannot be canceled out.

**Saravanan Thirumuruganathan** *on February 15, 2011 at 11:13 am* | *Reply*

Weixi,

The factor h_i^(d+2) is needed to handle higher dimensional data. Since I was only using 2 D points, I ignored it. The other reason was that I was also using gaussian kernel which smoothes the effect.

But you are right and a proper implementation would have added this factor.

**weixi** *on February 10, 2011 at 10:56 am* | *Reply*

Hi Saravanan,

Could you also comment on the choice of the kernel?
For example, what is the difference between using e^(-dist/(bandwidth^2)) and e^(-1/2*dist/(bandwidth^2)). Are they exactly the same kernel, or actually somewhat different? I understand that because of the division some constants can be canceled out, but I am not sure if that's the case with different Gaussian kernels.

**Saravanan Thirumuruganathan** *on February 15, 2011 at 11:15 am* | *Reply*

Weixi,

No they are not the same kernels. I wrote this code long back and it seems to me now that it might be a careless mistake.

**weixi** *on February 10, 2011 at 4:54 pm* | *Reply*

One more question with regard to the condition you chose to merge cluster centers:
if (distToCentroid < 4 * H) % H = 1 closestCentroid = j;

I understand errors in floating point calculations may lead to different data points being shifted to slightly different positions even if they are meant to converge to the same mode. However, Matlab does arithmetic with double-precision numbers, I do not understand how the final error can be as large as 4. Why 4? why not 0.001, or 1, or 10?

**Saravanan Thirumuruganathan** *on February 15, 2011 at 11:20 am* | *Reply*

Weixi,

Honestly I dont remember why had that constant. I derived the number based on the mean and std of the normal distribution that generated the points.

I remember using some theorem that gave a tail error bound for gaussian distribution and then rounded it as a nice number . I dont recall the exact theorem/idea I used☹

If I get any more details, I will shoot you an email !

BTW, thanks for catching the mistakes in my code ! I will try to write a better code when I get some time – or alternatively, if you have some code in your college website, I can link to yours !

**TatkaSmoula** *on March 1, 2011 at 6:44 pm* | *Reply*

Hi, thx for nice article.

I am trying to understand the mean shif algorithm but i am stucked in a mathematical expression. Can you please explain me, what means "double brackets" in kernel profile?

For example d-dimensional point x has a condition as follows: ||x|| < 1. Does it have anything common with the point's dimmensions?

My interest is in an article: D. Comaniciu and V. Ramesh: Real-Time Tracking of Non-Rigid Objects using Mean Shift.

Thanks for reply.

**Saravanan Thirumuruganathan** *on March 1, 2011 at 10:36 pm* | *Reply*

Hello TatkaSmoula,

The ||x|| stands for the norm operator. It can be either the L1 or L2(euclidean) norm. For eg in a vector , L1 norm is 1+2+3 = 6 and L2 norm is sqrt(1^2+2^2+3^2).

**Arjun Jain** *on March 29, 2011 at 11:27 pm* | *Reply*

thank you

**Yassmin** *on April 1, 2011 at 8:43 am* | *Reply*

Thank you this article really helped me🙂

**wpükgnwegpüer** *on April 4, 2011 at 4:23 am* | *Reply*

What do you think: How many points does the adaptive algorithm (your code) in MATLAB work readonably fast with?

Thanks.

**Saravanan Thirumuruganathan** *on April 7, 2011 at 11:19 am* | *Reply*

@wpükgnwegpüer,

It depends on lot of factors – for eg #dimensions , #data and so on. But even in pathological cases, I got an answer in around 20 iterations.

**weixi** *on April 4, 2011 at 12:48 pm* | *Reply*

Hi Saravanan,

thanks again for this excellent post.

With regard to the proof of convergence, you mentioned “The second part of the proof in [2] which tries to prove the sequence \{y_{j}\}_{j=1,2,…} is convergent is wrong.”

Could you please elaborate on this claim? I found an errata of the paper, but it only mentioned typographical errors.

**Saravanan Thirumuruganathan** *on April 7, 2011 at 11:28 am* | *Reply*

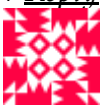
Weixi,

It has almost been an year since I wrote this post – so my memory is a bit hazy.

But I my memory serves right, it is not possible to prove that the sequence \{y_{j}\}_{j=1,2,…} . I found some resource for the same and I also discussed with my prof.Chris Ding about it. I will check if I find any more details of our discussion.

**eric** *on April 5, 2011 at 1:05 am* | *Reply*

Hi,where is the code ? I can't find it! Could you be so kind to send it to me ? Thank you!



> **Saravanan Thirumuruganathan** *on April 7, 2011 at 11:20 am* | *Reply*
>
> Eric,
>
> Code is linked in the post. The url is https://code.google.com/p/saravanant/source/browse/trunk/datamining/meanshift/meanshift.m .

**jifeng.shen@163.com** *on April 9, 2011 at 4:18 am* | *Reply*

Hi, the link of code at

https://code.google.com/p/saravanant/source/browse/trunk/datamining/meanshift/meanshift.m is not avaliable now!

so could you send me a copy of code ?

> **Saravanan Thirumuruganathan** *on April 9, 2011 at 11:59 am* | *Reply*
>
> Jifeng,
>
> Please check at http://saravanant.googlecode.com/svn/trunk/datamining/meanshift/meanshift.m. If not give your email id and i will mail it to you !

**jifeng.shen@163.com** *on April 9, 2011 at 7:01 pm* | *Reply*

Yes, It OK now! Thank you very much!

**Mean Shift Algorithm « Pacific NavigaTor's Blog** *on June 4, 2011 at 7:09 am* | *Reply*

[…] 3. Introduction To Mean Shift Algorithm […]

**indah** *on June 16, 2011 at 11:16 am* | *Reply*

hi…i've a final project to transform color image to grayscale by VCP algorithm…
i need value of a variabel, that is from mean shift algorithm…



so, how i can apply this algorithm to an image, where i just want to get the mean value from that…

help me…
thnx…

> **Saravanan Thirumuruganathan** *on June 20, 2011 at 11:31 am* | *Reply*
>
> indah,

Unfortunately, I do not much about image processing or computer vision and may not be able to help you. I think may be the matlab central has some image processing code ? Or there might be some opencv mean shift code in net ….

**Heba** *on June 25, 2011 at 9:16 am* | *Reply*
Good work ,Thanks a million:)

**kodylau** *on June 26, 2011 at 1:36 am* | *Reply*
Thank you very much! This is a really helpful introduction to mean shift.

**Sagitari** *on July 1, 2011 at 11:20 am* | *Reply*
Thank you so much. It helps much for my Seminar !! Thought i don't really into this but your article helped me how can i write mine . Thanks again for helping article ^^ have fun !!

**Valentino Traversa** *on July 26, 2011 at 4:57 am* | *Reply*
Thank you for this beautiful introduction, Saravanan!
If someone want to perform image segmentation, there is an open source software that does it [I discovered mean-shift trough it, and it work very well!].
It is named "Mirone", based on matlab:

http://w3.ualg.pt/~jluis/mirone/

**Saravanan Thirumuruganathan** *on August 3, 2011 at 1:37 am* | *Reply*
@Valentiono,

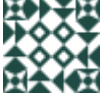
Thanks for the information ! I will take a look at the software ….

**Ahmet** *on August 23, 2011 at 4:01 am* | *Reply*
Hi,
I'm new in this subject and I don't understand. For example, i think y_i^0 = x_i means every x is equal to 1. am i right?

I have to use this algorithm for image processing. please help me and thanks for everything.

**Saravanan Thirumuruganathan** *on August 28, 2011 at 12:34 am* | *Reply*
Ahmet,
The line y_i^0 = x_i means set value of y_i at time 0 as the same as x_i. During time interval 1 , y_i^1 , it will use an update formula to compute this value.

**sana** *on September 2, 2011 at 6:09 pm* | *Reply*
Hi Saravanan,



I would like to use your code. but don't know how to use it as I don't have good experience in program especially matlab. I copied your codes into m.file but how is it run? I have only a 3D point set so need to cluster using mean shift. can you explain how to use these codes?do I need to get neibours(KNN) for each point? so how can I give input data? I hope, this is a stupid question. sorry..Plese explain each steps to run this code.

**Saravanan Thirumuruganathan** *on September 4, 2011 at 2:19 am* | *Reply*

Sana,

IIRC, the code already uses a variable called numDimensions and can handle arbitrary dimensions (although it cannot plot them). The easiest way to run the code is to load the file in Matlab and press the execute button. I used Octave to test my code. Currently it generates some random points and tries to cluster them. The code has two modes. In one, the bandwidth (H) is hard coded. In other mode (which occurs when useKNNToGetH is true), the nearest k neighbours decide the bandwith. I have hard coded K as 100. If you want you can change it in line 6. Hope this helps !

**Sushrut** *on September 29, 2011 at 12:16 pm* | *Reply*

Hi,
Great article!
could anyone please provide me with some C code in OpenCV which implements mean-shift ?
A small sample program will work🙂

**Jürgen Lorenz Simon** *on October 17, 2011 at 2:28 pm* | *Reply*

@Sushrut: cvMeanShift and cvCamShift. Check the OpenCV API and example code.

**Sushrut** *on November 28, 2011 at 6:06 am* | *Reply*

ok, thanks a lot🙂

**Hasan** *on October 18, 2011 at 5:29 am* | *Reply*

Thanks for your nice intuition about mean-shift

**vasanthi** *on November 16, 2011 at 11:10 pm* | *Reply*



Hello sir,
i have decided to use this mean-shift algorithm for non-convex environment for avoiding obstacle.
can i implement in Ns-2 is there possible or not

**Saravanan Thirumuruganathan** *on November 19, 2011 at 2:34 am* | *Reply*

Vasanthi,

My knowledge of networks is very limited. Hence I may not be the best person to comment on this.

**Deming** *on December 8, 2011 at 7:23 pm* | *Reply*



Dear Saravanan,

Great article!

Just a question about your Eq below "Using a Gaussian kernel as an example" and before Proof of Convergence

The summation in EQ 2 should be j=1 to n instead of i=1..n? or I miss somehting here?

Thanks

DW

**Saravanan Thirumuruganathan** *on December 14, 2011 at 6:47 pm* | *Reply*

Deming,

You are right.It must be iterating over j. I will fix it sometime this week.

**Punit** *on December 20, 2011 at 12:37 am* | *Reply*

Hey Saravanan, Excellent Job ..Can u GIve me the code in Matlab considering Spherical Kernal.

**Punit** *on December 20, 2011 at 12:38 am* | *Reply*

Hey Saravanan, Excellent Job ..Can u GIve me the code in Matlab considering Spherical Kernal.Thanks in Advance

**Saravanan Thirumuruganathan** *on December 25, 2011 at 2:31 am* | *Reply*

Punit,

The code for Spherical kernel seems very straightforward. I found some resources online too. Google is your friend🙂

**Anonymous** *on January 27, 2012 at 11:20 pm* | *Reply*

Million Thanks

**Pratik** *on March 22, 2012 at 4:41 pm* | *Reply*

Shouldn't the rectangular kernel be 1/(b-a) for the integral to be 1?

**Saravanan Thirumuruganathan** *on March 22, 2012 at 10:13 pm* | *Reply*

Pratik,

I think you are right. I will fix the post.

**Ahmed** *on April 18, 2012 at 10:16 pm | Reply*

hi saravanan!
thanx very much for your post, i ve got one question though:
when we apply the gradient ascent to the Kernel density estimator, why do you set the differential of the KDE to zero? is it to find a local maxima/minima?
Thank you🙂

**AB** *on April 20, 2012 at 10:21 am | Reply*

Hello Saravanan,
Thanks for the great explanation, the gradient ascent thing really made my life, but the derivation step has been bothering me, i am not quite getting the idea, if you could provide more details, and when you saying set to 0, do you mean you're setting the gradient to 0 so that you find local maxima ?
Thanks
AB

**elitap** *on April 26, 2012 at 9:45 am | Reply*

Hey,
First of all thanks for the realy great explenation of the mean shift algorithm.
But I am working with openCv, and I would like to use the mean shift algorithm to cluster some datapoints. Yea I know there exists a meanshift Funktion in openCv, but as fas as i know this funktion can only be used as a histogram based tracker. Does anyone konw how to use the opencv funktion for clutstering, or even better did someone implement this matlab function in c++, or is there a good c++ implementaion of a meanshift clustering around? I am thankfull for every clue?

**sean** *on June 26, 2012 at 4:03 am | Reply*

hello,
thank you for this wonderful introduction and for the souce code.
I would like to use the mean shift with the Epanechnikov kernel and I'm newbie I was wondering if someone could help me

thank you

**Anonymous** *on July 25, 2012 at 10:45 pm | Reply*

we wan the code for it….jaisu

**sean** *on July 31, 2012 at 5:28 pm | Reply*

I would use it but using the Epanechnikov kernel.
If you already have a code running I would have ravished the

thank you

**Anonymous** *on September 15, 2012 at 5:33 am* | *Reply*

Hi,
Thanks for this wonderful write up.
My aim is to classify an image into 2 clusters.These clusters may not be physically together.i.e. few pixels may belong to cluster1 then few to cluster 2 ,again to cluster1 and all. which technique will work best in this case?
Regards
Richa

**best Uk web agency** *on September 15, 2012 at 8:38 pm* | *Reply*

I'm curious to find out what blog platform you're using?
I'm experiencing some minor security issues with my latest website and I would like to find something more safeguarded. Do you have any recommendations?

**hamed** *on October 7, 2012 at 4:07 am* | *Reply*

hi dear saravanan,
please rectify the cod, i use it in my project but it has error , i do not understand what is the problem!
please help me , can you rectify your code and send me the correct one?
thank you very much

**Wildthing** *on October 24, 2012 at 3:21 pm* | *Reply*

Thank you for this great topic, but i have a question:How to define h (needed for the density estimation) when i use knn to find neighbors?

**Wildthing** *on October 24, 2012 at 4:53 pm* | *Reply*

I have another one; in the formulation of the meanshift there is -x at last. what does that mean, and why did you not include it in the code?

**Mahesh** *on November 23, 2012 at 1:48 am* | *Reply*

Thank you thanks a lot.

**Deepak** *on December 17, 2012 at 2:04 pm* | *Reply*

nice article saravanan. i have implemented my own version which uses a kd-tree and it works fairly well and i am applying it for some clustering problems where a model-based approach seems infeasible due to the massive number of clusters i have to deal with.

However, i have been using a fixed bandwidth so far and on some of my data i find it very hard (literally impossible) to set the bandwidth such that it works on all my data given its diversity. I am now searching for ways to make the bandwidth data-driven or locally adaptive. I like the knn idea (gives large bandwidth at sparse areas) and smaller bandwidth (at dense areas) but then i but setting k doesnt seem easier too.

What does your intuition say on the bandwidth selection issue? It would be great if you could point to some important papers in this regard with ideas that work robustly. I found lots of papers on google but i am not sure which works.

The knn idea would surely make the code run faster but in terms of achieving good clustering consistently of a different datasets (sparse to dense) is it any better than using a fixed bandwidth in terms of sensitivity of parameters?

Looking forward to hear from you.

Regards,

Deepak

**Sohail Shafii** *on February 13, 2013 at 9:34 pm* | *Reply*



Shouldn't there be a normalization constant in front of the kernel(s)? Comaniciu et al used defined a kernel profile, then a kernel based on the kernel profile that satisfies the "integrates to 1" criterion, which requires a constant I think.

**Yazhou Ren** *on April 9, 2013 at 8:14 am* | *Reply*

Hi Saravanan,
Thanks for giving this wonderful article.
I have a question for the code.
In function domeanshift(), you wrote:

```
curDataPoint = dataPoints(i,:);
euclideanDist = sqdist(curDataPoint',origDataPoints');
bandwidth = getBandWith(origDataPoints(i,:),origDataPoints,euclideanDist,useKNNToGetH);
kernelDist = exp(-euclideanDist ./ (bandwidth^2));
```

But I think the following codes may be more reasonable

```
curDataPoint = dataPoints(i,:);
euclideanDist = sqrt( sqdist(curDataPoint',origDataPoints') );
bandwidth = getBandWith(origDataPoints(i,:),origDataPoints,euclideanDist,useKNNToGetH);
kernelDist = exp(- (euclideanDist.^2) ./ (bandwidth^2));
```

The reason is that in getBandWith(), we set the bandwidth to the k-th smallest euclidean distance, not the k-th smallest square euclidean distance.

**Raj kumar sah** *on May 10, 2013 at 12:01 am* | *Reply*

problems of mean shift ???

**Anonymous** *on June 21, 2013 at 2:03 am* | *Reply*

Hi Saravanan. u have explained it very well. I am looking for C# code of mean shift. Any one having mean shift code in C# plz let me know, coz i dont want to write it all from scratch. i am doing a research project and mean shift a bit part of it, so if any one having the code plz mail me, I will appreciate it.

**Anonymous** *on November 6, 2013 at 3:31 pm* | *Reply*

thank you…

**Gaurav** *on December 26, 2013 at 2:44 am* | *Reply*

Hi Saravanan, Thanks for the post on Mean Shift. Can you please send me the link for the Chris Ding's lecture notes/material, I am not able to find it on the internet.

**kakal** *on June 10, 2014 at 12:53 pm* | *Reply*

Hey Saravan! Thank you for your nice and analytical post! Could you please tell me, what change should be done in your code in order to not insert the number of classes manually? I need to find the proper modes in an automated manner. Thank you in advance

Comments RSS