Image Segmentation

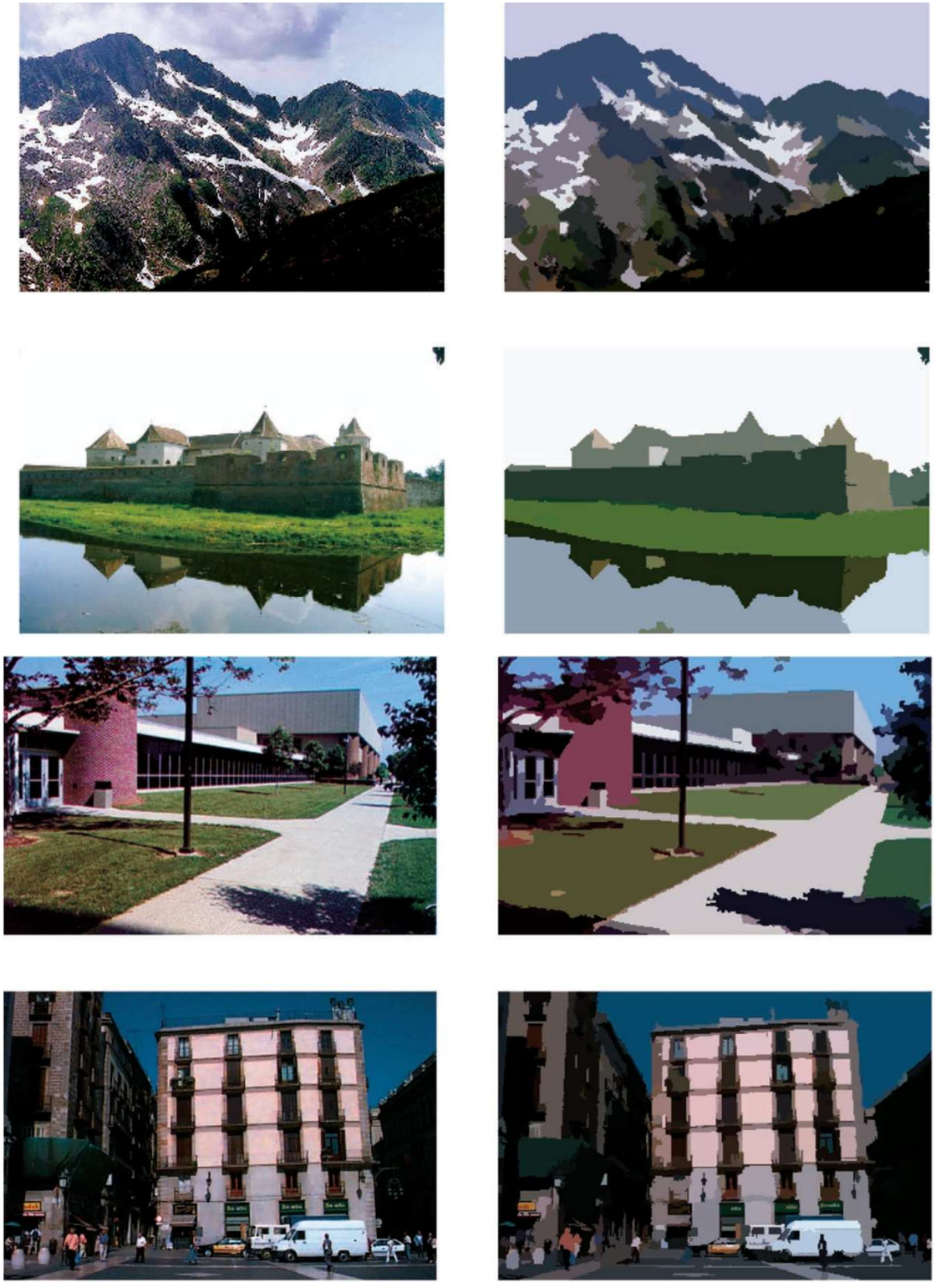
CS 663, Ajit Rajwade

What is image segmentation?

- The process of dividing the image into disjoint regions or segments, each of which contains pixels that (in some sense) belong "together", or are part of the same "**entity**".

- Example, pixels with **nearby** spatial locations and **similar** color (or grayscale) values could be considered part of one segment.

Images

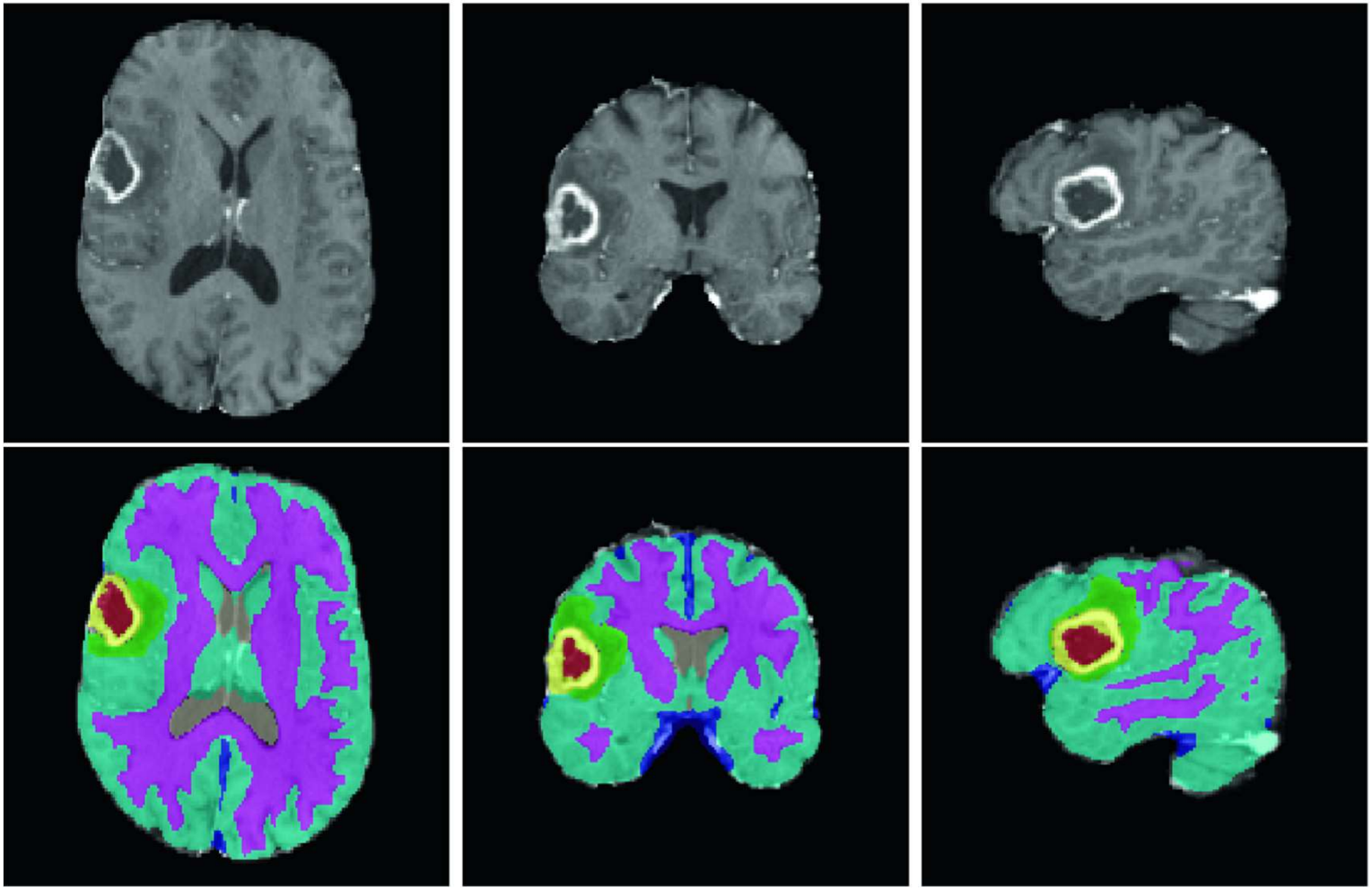


Segmented Images: each segment is assigned a single color. In this example, that color happens to be the average color of all pixels that belong to that segment. Such a segmented image looks like a cartoon!

https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf

Applications of Image Segmentation

- **In medical imaging:** Location of special types of tissues such as tumours and measuring their geometric properties such as perimeter/area/volume.
- **Image editing:** Replacing a segment representing a particular entity by some other pattern or entity.
- **Object detection:** examples include people or vehicles in traffic videos, or roads/water-bodies/vegetation in satellite images, or fruits/vegetables in images of orchards/farms.

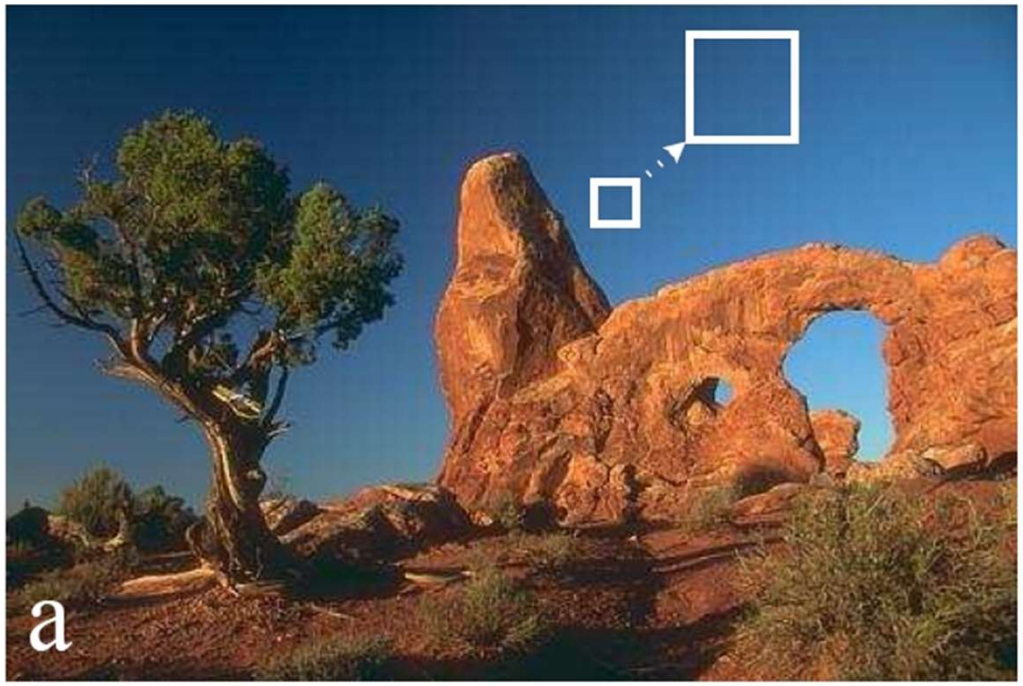


https://link.springer.com/chapter/10.1007/978-3-030-11726-9_26

Questions concerning segmentation

- How to define **similarity** between pixels? If based on color, which color space? If not color, do we pick gradients or second derivatives or something else?
- How **many** segments will the image consist of?

* Extreme case (1): The entire image is one segment (called under-segmentation).
* Extreme case (2): Every pixel is a separate segment (called over-segmentation)
* Both these extremes are generally undesirable!



Example of over-segmentation!

Example of under-segmentation!

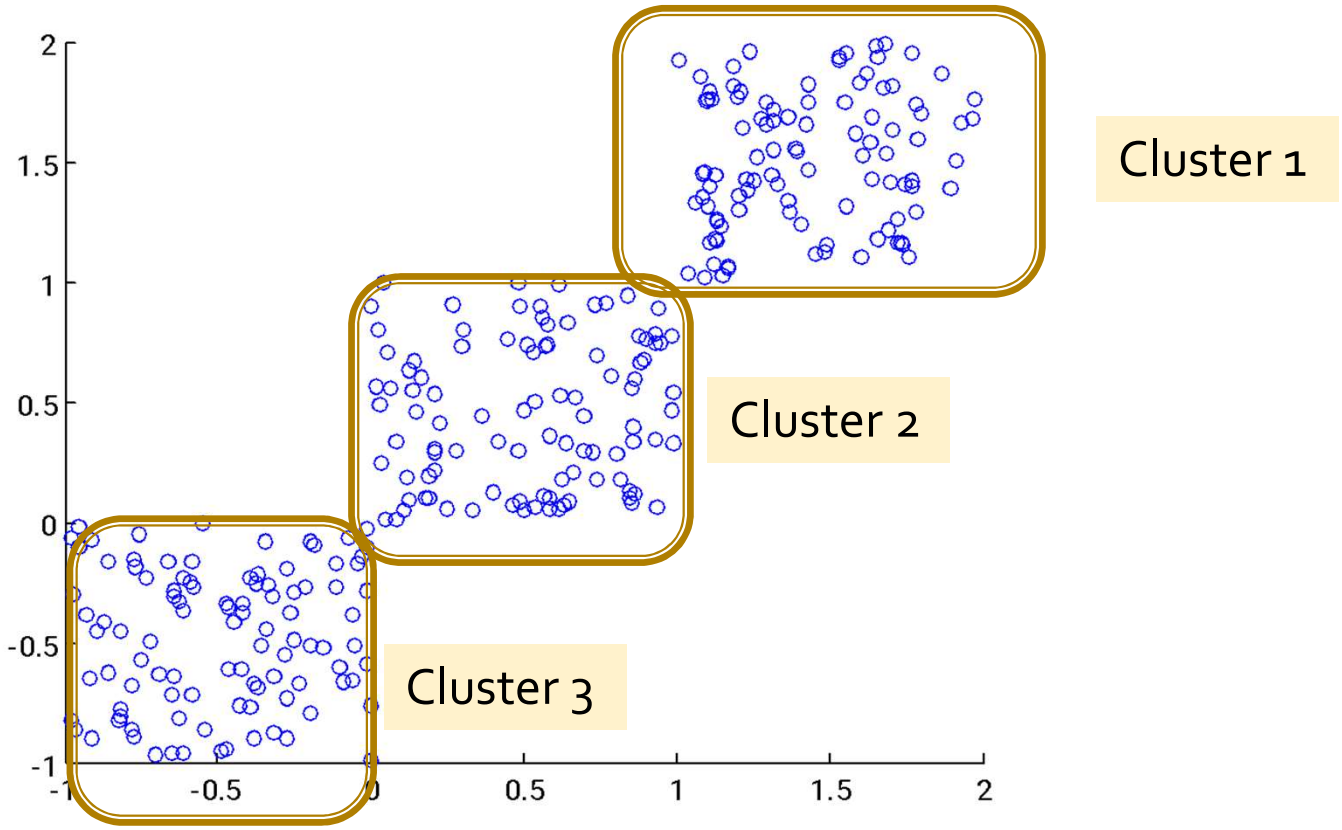
Segmentation approaches

- There exist two major approaches to image segmentation: one of them is **statistical**, the other is **geometric**.

- We will look at two statistical approaches in this course.

Segmentation as clustering

- Clustering is a concept in (unsupervised) machine learning.
- Given a set of N data-points $\{q_1, q_2, \ldots, q_N\}$ each in some *d* dimensions, it aims to group them into disjoint sets called as **"clusters"**.
- There are a gazillion techniques for clustering (eg: k-means, fuzzy k-means, mixture models, mean shift, etc.).
- Out of these, we will mainly study two techniques: **k-means** and **mean shift**.


Cluster 1
Cluster 2
Cluster 3
2
1.5
1
0.5
0
-0.5
-1
-0.5
0.5
1
1.5
2

K-means

Animation for k-means: http://shabal.in/visuals/kmeans/3.html, http://www.youtube.com/watch?v=5FmnJVv73fU

- **Input**: *N* data-points: $\{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_N\}$.
- **Output**: A set of *K* clusters, such that each point $\mathbf{q}_i \in R^d$ ($1 \le i \le N$) belongs to exactly one cluster C_j ($1 \le j \le K$, $K << N$).
- **Method**:

1. Pick *K* = the number of clusters (guessing the "right" K is a difficult problem).
2. Define K **cluster centers** (called **centroids**) $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K\}$ where each center is a point in R^d. Initialize them to random values.
3. For every $\mathbf{q}_i$, find its closest (in terms of sum squared distance) $\mathbf{c}_j$ and assign it the label 'j'. Do this for every point $\mathbf{q}_i$.
4. Re-compute all the *K* cluster centers as follows:

$$\boldsymbol{c}_j = \sum_{q_i : label(q_i)=j} \boldsymbol{q}_i \; / \sum_{q_i : label(q_i)=j} 1$$

Repeat steps 3 and 4 above, for a fixed number of iterations, or until the cluster center values don't change much (called as a **convergence criterion**).

Variants of k-means

- Use some other **distance measure** instead of Euclidean (i.e. sum of squared) distances.
- Instead of finding centroids, find **medoids**.
- **Fuzzy** k-means, where a point has a fuzzy degree (between 0 and 1) of belonging to a cluster.
- Use **fast data structures** to speed up the search for the closest centroid.
- **Gaussian mixture models:** each cluster is represented by a Gaussian probability density function (pdf), where the mean of the Gaussian pdf coincides with the cluster centroid. Assignments of points to clusters are not deterministic, but probabilistic.

K-means and Image segmentation

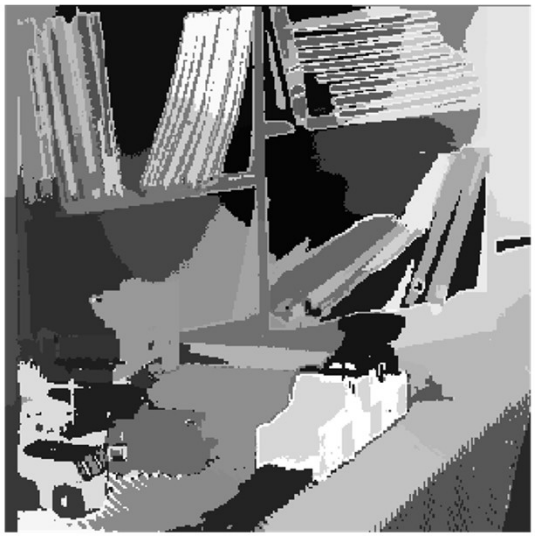
- For each pixel location (x_i, y_i), create a feature vector $\mathbf{q_i} = (x_i, y_i, R(x_i, y_i), G(x_i, y_i), B(x_i, y_i))$ in case of color images, or $(x_i, y_i, I(x_i, y_i))$ in case of gray-scale images.
- Perform k-means on these feature vectors. For an image with N pixels, there will be N feature-vectors to be clustered. Each feature vector is 5-dimensional for a color image, or 3-dimensional for a grayscale image.
- What will happen if we use $(R(x_i, y_i), G(x_i, y_i), B(x_i, y_i))$ as a feature vector?

But…K-means is problematic!

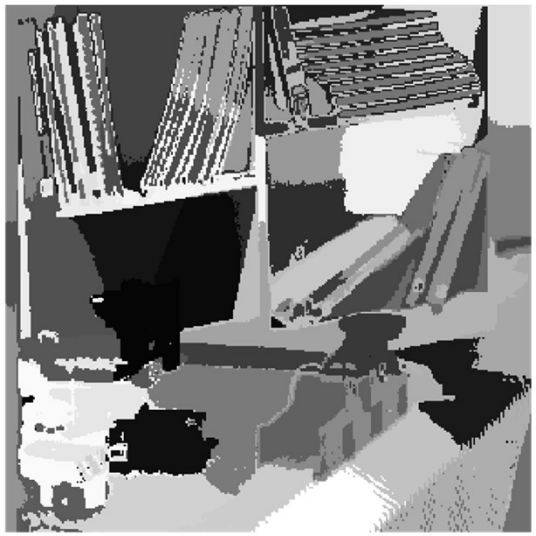
- **How to find a good K?** Too small a K will lead to under-clustering, too large a K is very expensive and will lead to over-clustering.
- Highly **sensitive to the initial assignments** of the centroid values.
- Can be slow to converge, and in fact may converge to a "bad" assignment of clusters.

K = 30

K = 30



K = 50



K = 50

The segmentation results using K-means will vary if you chose different values of K. Even for the same K, you will get different results if you initialized the cluster centers differently.

Mean-shift

- It is a technique that does not require you to pre-specify the value of *K*.
- It gives you a mathematically clean and convenient to use convergence criterion.
- It is heavily parallelizable.
- It has shown success in so many image processing and computer vision applications!

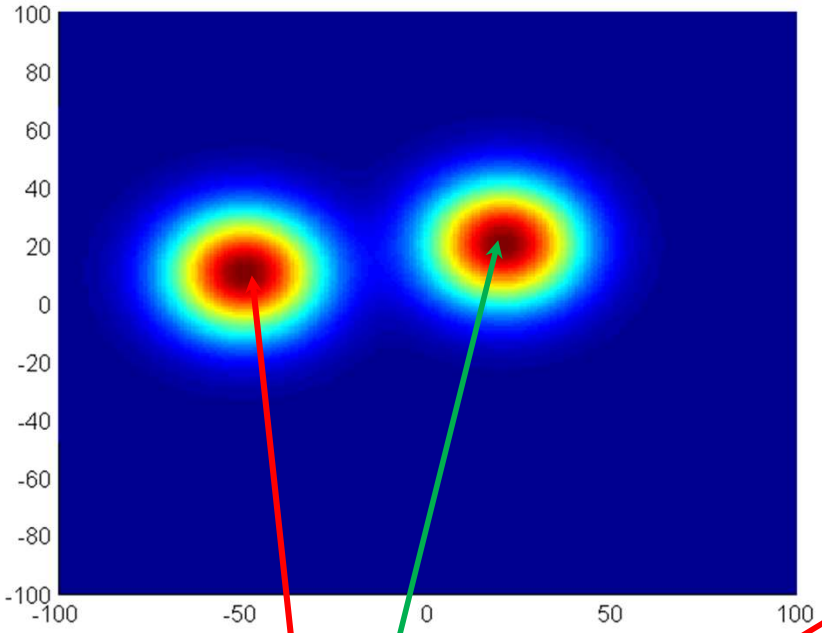
To understand mean-shift, we need to understand an important machine learning concept – called **probability density estimation**, and in particular **kernel density estimation**.

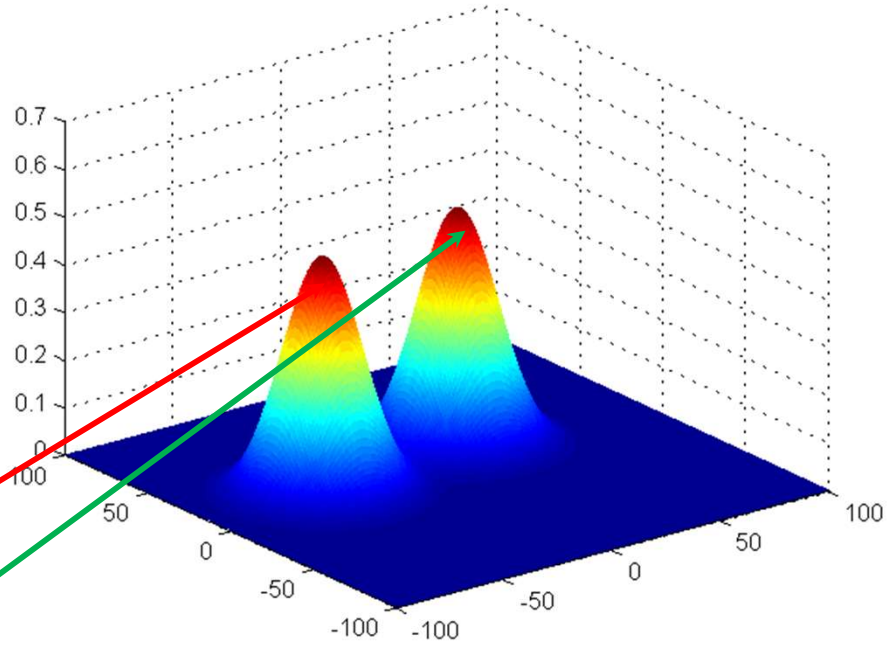
Clustering and probability density functions

- We can consider that the data points we want to cluster are **samples** from some (unknown) probability density function (pdf).
- The underlying pdf is unknown, but given the samples, we can **estimate** it.
- If we have more and more samples, our estimates will be more and more accurate.
- It turns out (see next slide) that the cluster centers are **modes** *(i.e.* **local maxima***)* of the pdf, i.e. they are points where the pdf has **maximum value**!

$$p(x,y) = \frac{1}{2}\frac{1}{2\pi\sigma_1}\exp(-\frac{(x-x_1)^2+(y-y_1)^2}{2\sigma_1^2}) + \frac{1}{2}\frac{1}{2\pi\sigma_2}\exp(-\frac{(x-x_2)^2+(y-y_2)^2}{2\sigma_2^2});$$

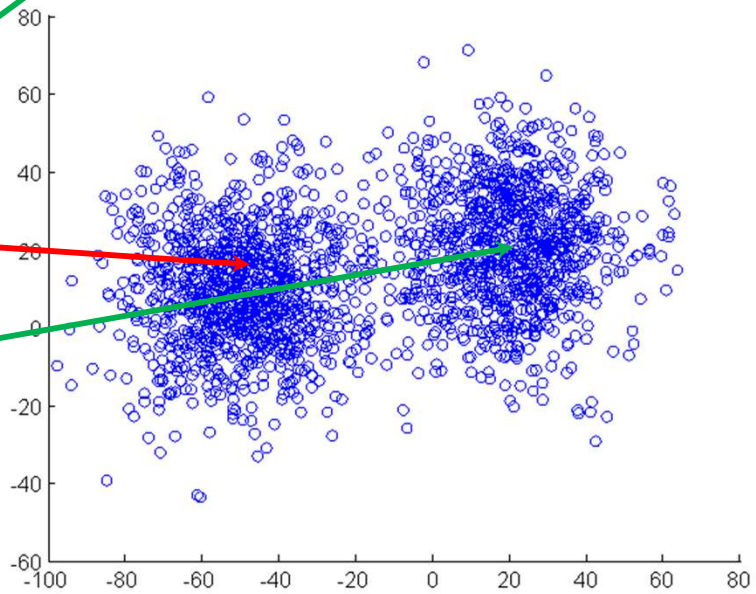
$$x_1 = y_1 = 20; x_2 = -50, y_2 = 10; \sigma_1 = \sigma_2 = 15$$





Cluster center 1

Cluster center 2



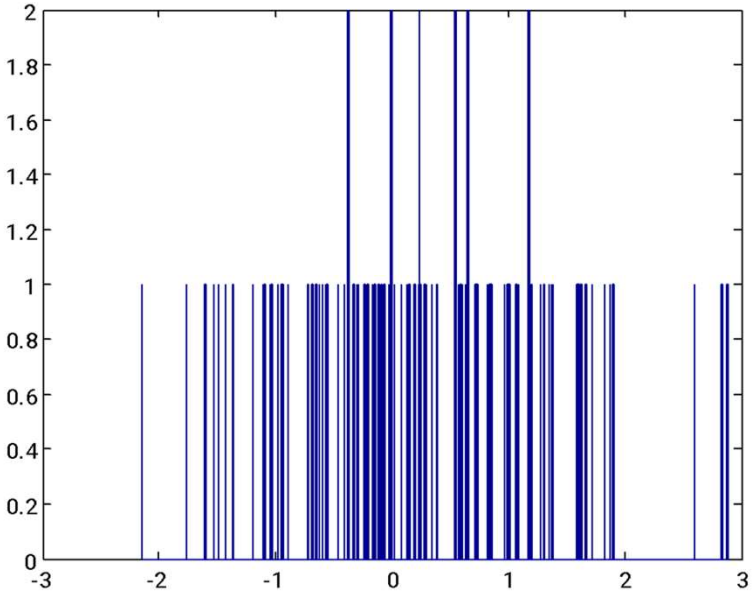
Samples from the above pdf plotted as a scatterplot in MATLAB. Notice the modes of the pdf and the cluster centers are the same.

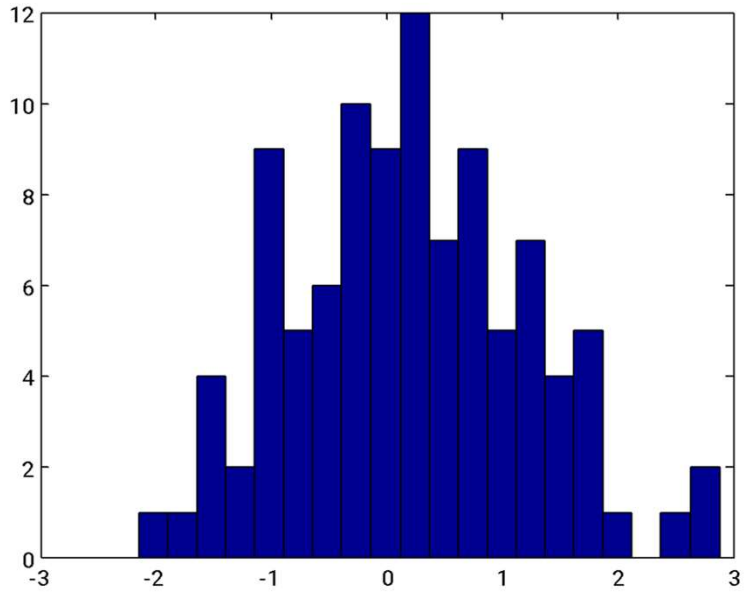
Clustering and pdf's

- If we knew the pdf underlying the samples, we could identify the modes, and then our clustering is done.
- But we DO NOT know the pdf.
- So we need to estimate it using a method shown on the next few slides.
- The great thing about the mean-shift algorithm is that it can **identify the modes of the probability density efficiently**. This will help us in image segmentation.

Histograms

- Consider a set of N samples, $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, with $\mathbf{x}_i \in R^d$ $(1 \le i \le N)$. We want to estimate the underlying pdf.
- The most well-known method to estimate the pdf is the **histogram**.





Histograms

- Computing a histogram is equivalent to putting individual samples into **boxes** (see figure on the previous slide).
- The optimal binwidth is $O(N^{-1/3})$, i.e. the optimal number of bins is $O(N^{1/3})$.
- The error between the true pdf and its estimate via the histogram converges at the rate $O(N^{-2/3})$.
- Problems with the histogram:
  - (1) picking the bin width (i.e. width of the box kernel) – equivalently the number of bins,
  - (2) discontinuity of the density estimate, i.e. sudden change in estimate if the location of the bin is slightly perturbed.

Kernel density estimates

- This is equivalent to putting a smoothing kernel around each sample point (no need for binning).
- The overall equation for a kernel density estimate (KDE) in 1-D is:

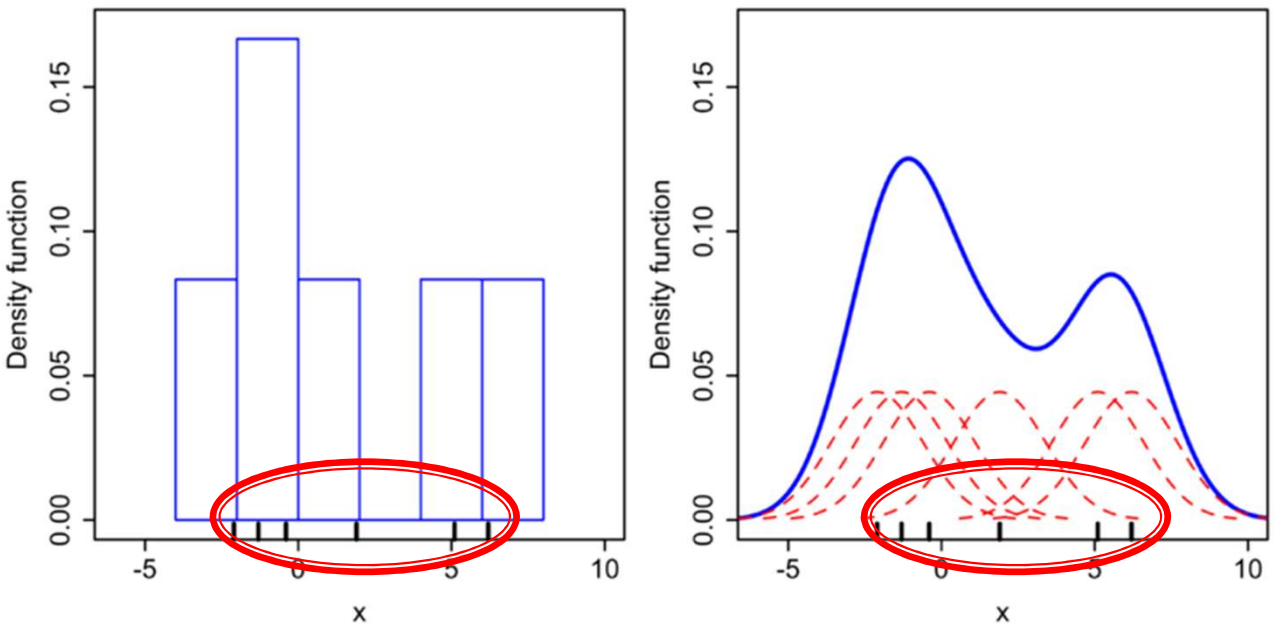
$$\hat{p}_N(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i; \sigma)$$

This stands for the i-th sample. There are totally N samples.

Kernel density estimates

- If we choose a Gaussian kernel in 1D, the equation looks like:

$$\hat{p}_N(x) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi)^{1/2}\sigma}\exp\left(-\frac{(x-x_i)^2}{2\sigma^2}\right)$$



Comparing a histogram with a Gaussian kernel density estimate. See how we are putting a Gaussian function around each sample (marked with a small vertical line in the figure on the right)

Kernel density estimates

- The overall equation for a kernel density estimate (KDE) in higher (i.e. $d > 1$) dimensions is:

$$\hat{p}_N(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} K(\boldsymbol{x} - \boldsymbol{x}_i; \boldsymbol{\Sigma})$$

Gaussian Kernel in higher dimensions

- The Gaussian kernel in higher dimensions is given as:

$$K(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}\sqrt{|\boldsymbol{\Sigma}|}}\exp\left(-\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^t\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

$\boldsymbol{x} \in R^d, \boldsymbol{\mu} \in R^d = E(\boldsymbol{x})$

$\boldsymbol{\Sigma}$ is a $d\ x\ d$ covariance matrix : it is symmetric and positive semi - definite

$\Sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)]$

- Sometimes a diagonal covariance matrix is used:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & 0 & . & . & 0 \\ 0 & \sigma_2^2 & . & . & 0 \\ & & . & & . \\ & . & . & . & 0 \\ 0 & . & . & . & \sigma_d^2 \end{pmatrix}; K(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}\prod_{i=1}^{d}\sigma_i}\exp\left(-\sum_{i=1}^{d}\frac{(x_i-\mu_i)^2}{2\sigma_i^2}\right)$$

Here x_i stands for the i-th element of vector **x**

Gaussian Kernel in higher dimensions

- Going one step further, a fully isotropic covariance matrix is sometimes used for simplicity:

$$\Sigma = \begin{pmatrix} \sigma^2 & 0 & . & . & 0 \\ 0 & \sigma^2 & . & . & 0 \\ & & . & & . \\ & . & . & . & 0 \\ 0 & . & . & . & \sigma^2 \end{pmatrix}; K(x;\boldsymbol{\mu},\Sigma) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left(-\sum_{i=1}^{d} \frac{(x_i - \mu_i)^2}{2\sigma^2} \right)$$

Gaussian Kernel in KDE

- With the Gaussian kernel, the equation for a kernel density estimate (KDE) is:

$$\hat{p}_N(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi)^{d/2}\sqrt{|\boldsymbol{\Sigma}|}}\exp(-(\boldsymbol{x}-\boldsymbol{x}_i)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{x}_i)/2)$$

$$\boldsymbol{x} \in R^d, \boldsymbol{x}_i \in R^d$$

Gaussian Kernel in KDE

- If we choose an isotropic Gaussian kernel, the equation for a kernel density estimate (KDE) is:

$$\hat{p}_N(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi)^{d/2}\sigma^d}\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right) = \frac{1}{N}\sum_{i=1}^{N}K(\boldsymbol{x}-\boldsymbol{x}_i;\sigma)$$

Other kernels in KDE

- Apart from Gaussian kernels, other types of kernels can also be used.

- The kernel must satisfy:

$$\int K(\boldsymbol{x})d\boldsymbol{x} = 1;$$

K must be symmetric

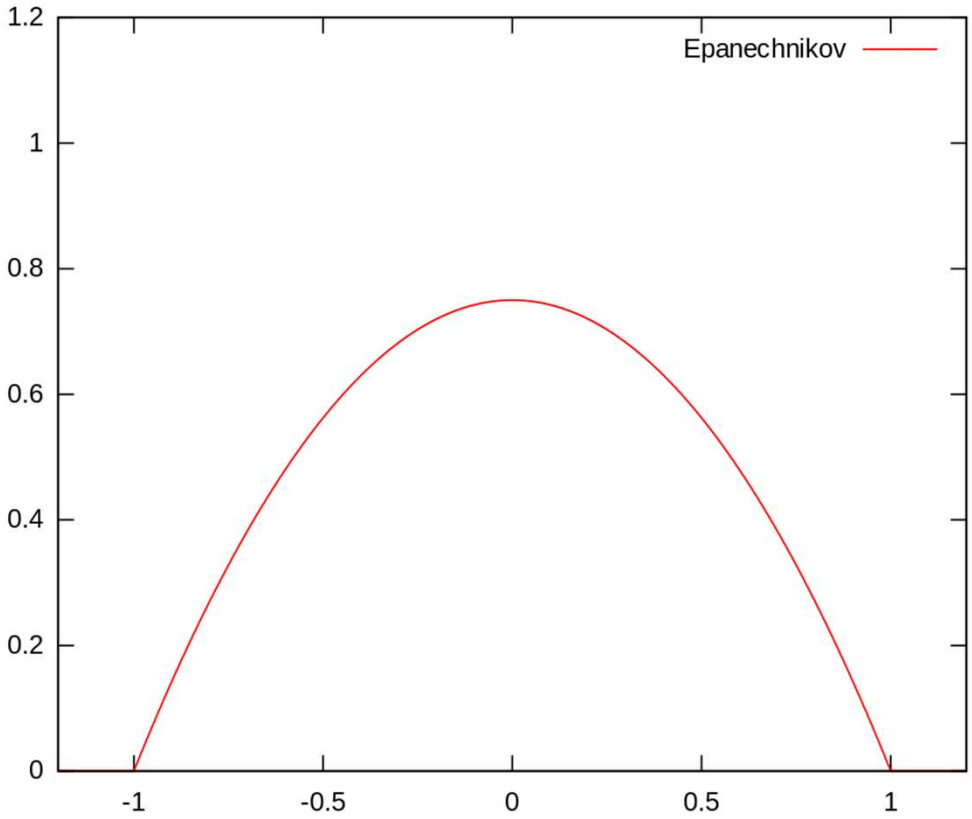
$$\int \boldsymbol{x}K(\boldsymbol{x})d\boldsymbol{x} = 0;$$

$$\lim_{\|\boldsymbol{x}\|\to\infty} K(\boldsymbol{x}) = 0$$

$$\text{Remember}: \boldsymbol{x} \in R^d$$

Other kernels in KDE

- Epanechnikov (parabolic):

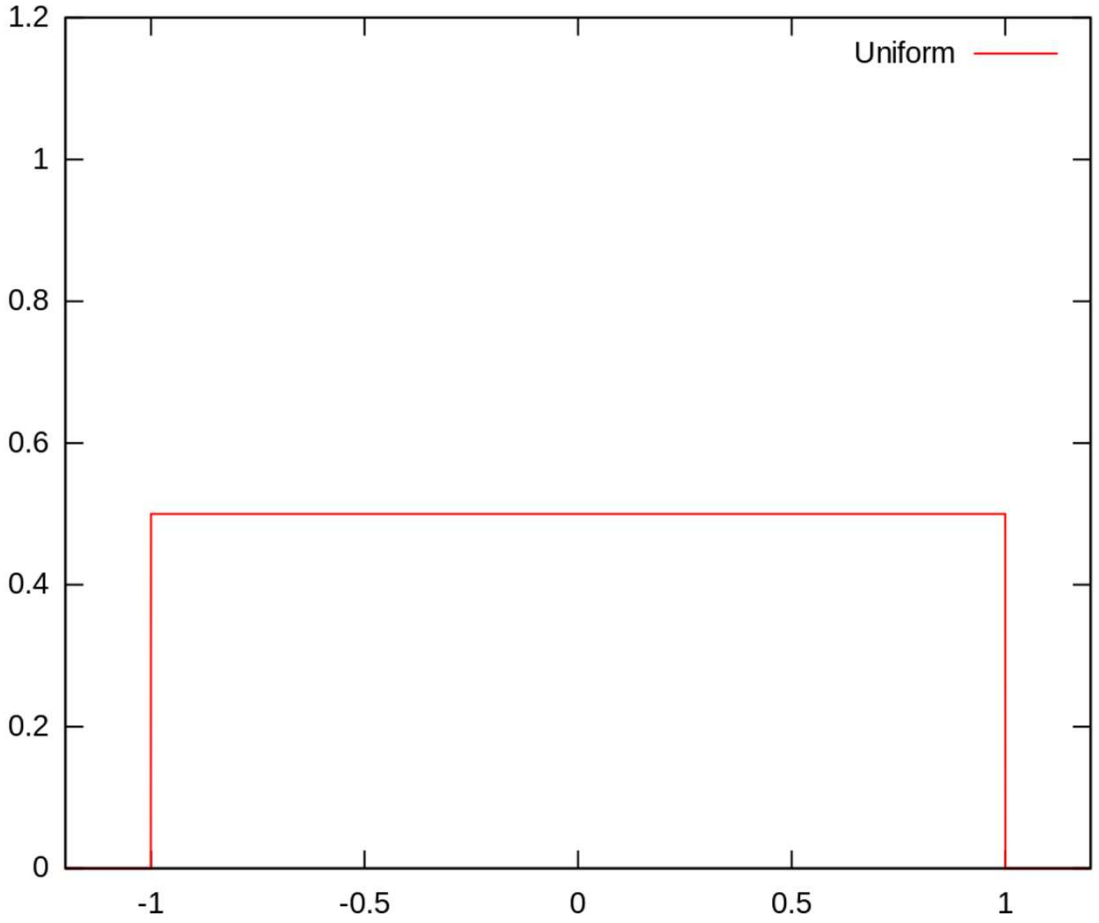


$$K(u) = 0.75(1-u^2)$$

https://en.wikipedia.org/wiki/Kernel_(statistics)

Other kernels in KDE

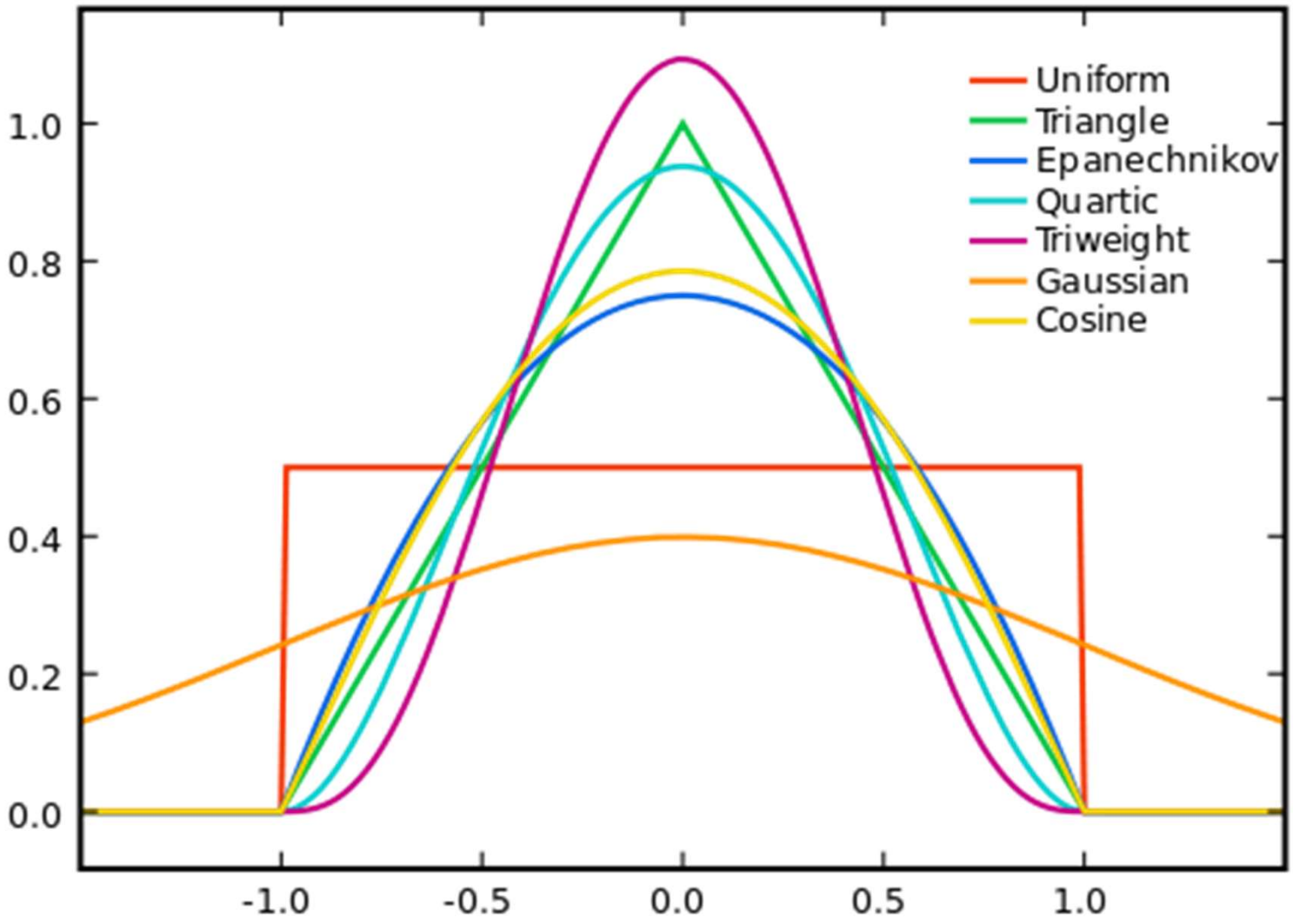
- Uniform:



$$K(u) = 0.5(cons\tan t)$$

https://en.wikipedia.org/wiki/Kernel_(statistics)

Other kernels in KDE



https://en.wikipedia.org/wiki/Kernel_(statistics)

Kernel density estimates

- Whether we are using histograms or KDEs, these are all (non-parametric) estimates of the true underlying pdf.

- We have the following properties:

$$\lim_{N\to\infty} E[\hat{p}_N(x)] = p(x) = \text{true PDF}$$
$$\lim_{N\to\infty} Var[\hat{p}_N(x)] = 0$$

http://faculty.washington.edu/yenchic/18W_425/Lec6_hist_KDE.pdf

Kernel density estimates

- For Gaussian and some other kernels, it has been proved that as the number of samples (i.e. N) tends to infinity, the kernel density estimate becomes more accurate and the error rate is $O(N^{-0.8})$ in 1D (in general $O(N^{-4/(4+d)})$ for *d* dimensions), provided that the kernel widths σ tend to 0 at a rate slower than N^{-1}.
- This was proved by Emmanuel Parzen in a classic paper in 1952. Hence KDE is sometimes also called **Parzen windows**.
- These estimates **have better error rate** than simple histograms which converge at the rate of $O(N^{-0.667})$ in 1D (in general $O(N^{-2/(2+d)})$ for *d* dimensions).
- KDEs are also smoother than histograms. Unlike histograms they are also differentiable everywhere.
- For these reasons, KDEs are preferred over histograms in many applications.

http://faculty.washington.edu/yenchic/18W_425/Lec6_hist_KDE.pdf

Mean shift algorithm

INPUT : N data - points to be clustered $\mathrm{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$

For every data - point $\boldsymbol{x} \in X$, do

{

$$\boldsymbol{x} = \frac{\sum_{i=1:\boldsymbol{x}_i \in N(\boldsymbol{x})}^{N} \boldsymbol{x}_i \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right)}{\sum_{i=1:\boldsymbol{x}_i \in N(\boldsymbol{x})}^{N} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right)}$$

We will see the motivation for this algorithm on the next several slides.

Small neighborhood around the point

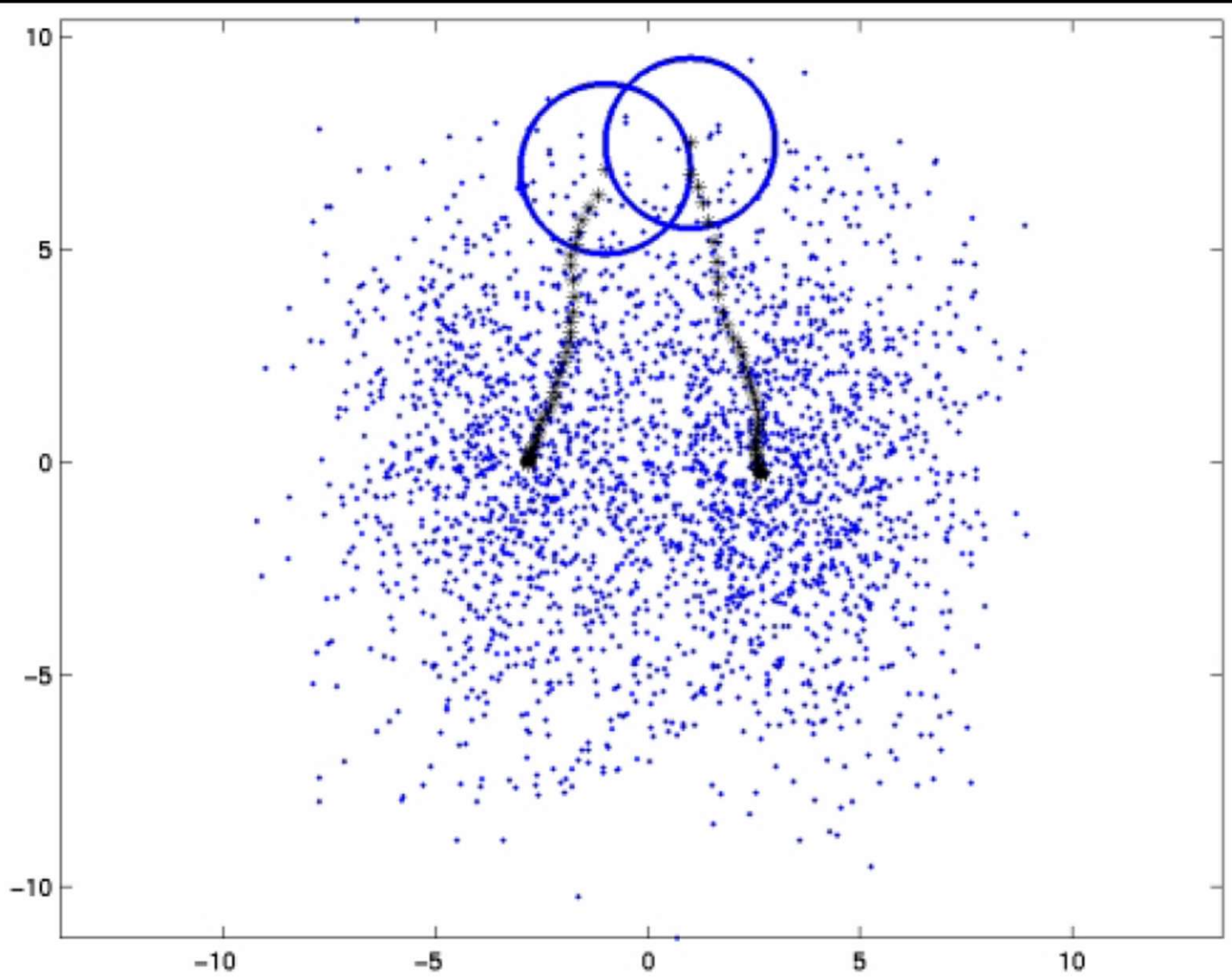
} until a convergenc e criterion is met, i.e. until

$\boldsymbol{x}$ does not change by more than some (small - valued) ε

OUTPUT : The set X divided into clusters, as many datapoints will converge to the same point

Mean shift

Mean-shift mode finding



Source: web tutorial on mean-shift-
http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUZEL1/MeanShift.pdf

Mean Shift Demos

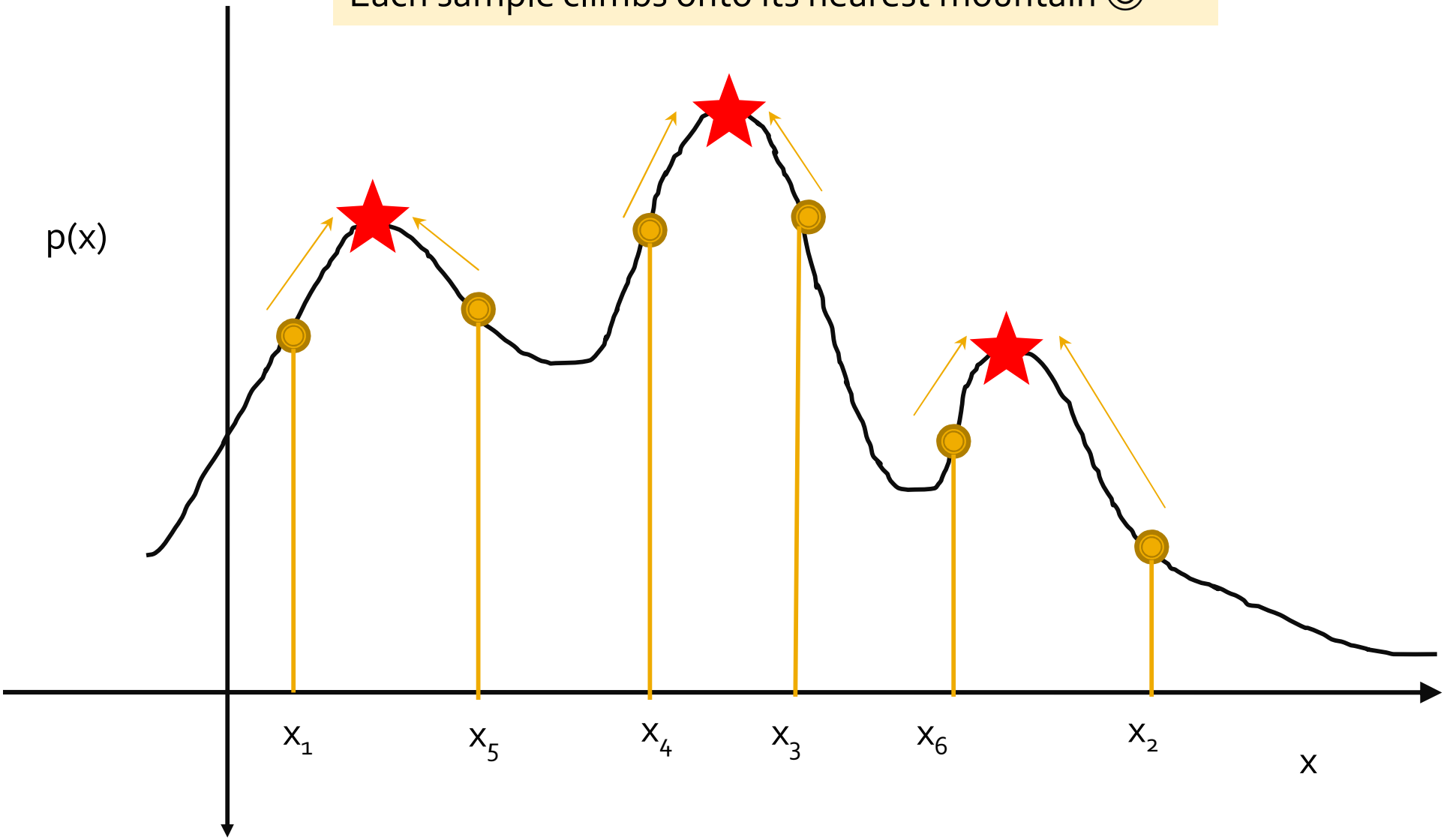
http://www.youtube.com/watch?v=kmaQAsotT9s

http://www.youtube.com/watch?feature=player_detailpage&v=M8B3RZVqgOo#t=2275

Mean shift algorithm

- In the mean-shift method, for each sample x_i we want find the closest mode (of the pdf estimated from all the N samples $\{x_i\}$).
- For this, we perform a **gradient ascent**, i.e. we move in the direction of **grad(p(x_i))** [which is the direction of greatest increase in the density], **starting from x_i**.

Mean shift is like climbing in a range of mountains.
Each sample climbs onto its nearest mountain ☺



Digression: Gradient ascent

- Suppose we want to maximize a function f(x) w.r.t. x.
- The procedure for the same is given below:

$x = \text{random value}$

Repeat :

$x = x + \varepsilon \nabla f(x)$

until some convergenc e criterion is met, i.e. until the value of x does not change significan tly during updates.

The ε is a small step - size : if it is too large, the gradient ascent will not converge

If it is too small, the procedure will take too long

Digression: Gradient ascent

- In practice, we use an adaptive step-size.
- That is, we start with a large value of ε.
- If that value leads to an updated x for which f(x) does not increase, we reduce the value of ε by some factor (maybe 0.9 times the earlier value) – until you get an increase in the value of f(x).

Mean-shift

$$\hat{p}_N(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{(2\pi)^{d/2}\sigma^d}\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right) = \frac{1}{N}\sum_{i=1}^{N}K(\boldsymbol{x}-\boldsymbol{x}_i;\sigma)$$

$$\nabla\hat{p}_N(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}\sigma^{d+2}N}\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{x})\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right)$$

$$= \frac{1}{(2\pi)^{d/2}\sigma^{d+2}N}\left[\sum_{i=1}^{N}\exp\left(-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}\|^2}{2\sigma^2}\right)\right]\left[\frac{\sum_{i=1}^{N}\boldsymbol{x}_i\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right)}{\sum_{i=1}^{N}\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right)}-\boldsymbol{x}\right]$$

$$= \frac{1}{\sigma^2}\hat{p}_N(\boldsymbol{x})m(\boldsymbol{x};\sigma)$$

$$\therefore m(\boldsymbol{x};\sigma) \propto \frac{\nabla\hat{p}_N(\boldsymbol{x})}{\hat{p}_N(\boldsymbol{x})}$$

Divide and multiply by the following

$$\sum_{i=1}^{n}\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right)$$

Mean shift vector – the first term is the weighted mean of all the samples

Mean shift

- Notice the mean-shift vector points in the same direction as grad($p_N(\mathbf{x}_i)$).
- So we can move in the direction of the mean shift vector!
- Mean-shift is a form of adaptive gradient ascent. We don't need to pick the step-size, it comes in automatically.
- In low-density regions, i.e. where p($\mathbf{x}$) is low, mean-shift moves very fast (why?). In higher density regions, it moves slowly.
- These mean-shift based gradient updates have been shown to converge to the local mode, with monotonic increase in the PDF all along. (Theorem 1 of this paper.)
- Although, we have shown derivations for the Gaussian kernel, these properties of mean shift and the mean shift vector are true for any kernel which satisfies the aforementioned properties.

Mean shift for clustering

- Here is how the mean shift algorithm works:

(1) Given a datapoint **x**, compute the mean shift vector **m**(**x**;σ).

(2) **Translate the point** using the mean shift vector, i.e. **x** = **x** + **m**(**x**;σ).

$$\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{m}(\boldsymbol{x};\sigma) = \boldsymbol{x} + \frac{\sum_{i=1}^{N} \boldsymbol{x}_i \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right)}{\sum_{i=1}^{N} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right)} - \boldsymbol{x} = \frac{\sum_{i=1}^{N} \boldsymbol{x}_i \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right)}{\sum_{i=1}^{N} \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}_i\|^2}{2\sigma^2}\right)}$$

(3) Repeat steps 1 and 2 until convergence, i.e. until the value of **x** does not change very much. When that happens, we have reached a local mode of the pdf.

(4) Repeat steps 1, 2 and 3 for every sample point x.

Mean shift for clustering

- Equation for mean shift iteration in step (3) on the previous slide:

For $t = 0$, until convergence

$$x^{(t+1)} = \frac{\sum_{i=1}^{N} x_i \exp\left(-\frac{\left\|x^{(t)} - x_i\right\|^2}{2\sigma^2}\right)}{\sum_{i=1}^{N} \exp\left(-\frac{\left\|x^{(t)} - x_i\right\|^2}{2\sigma^2}\right)}, t = t+1$$

Note: this procedure is guaranteed to converge in a finite number of iterations (in practice, as few as 10-20 iterations). The proof of convergence can be found in the paper by Comaniciu and Meer.

- At the end of the procedure (i.e. after step (4)), we are left behind with all **the maxima (modes) of the underlying pdf.**
- If you carefully go through the algorithm, you don't even need to estimate the full pdf. You only estimate **mean shift vectors**!

Mean shift for clustering: speed-up

- In practice, the summations are computed only over those samples that in a pre-defined neighborhood around the point being updated.

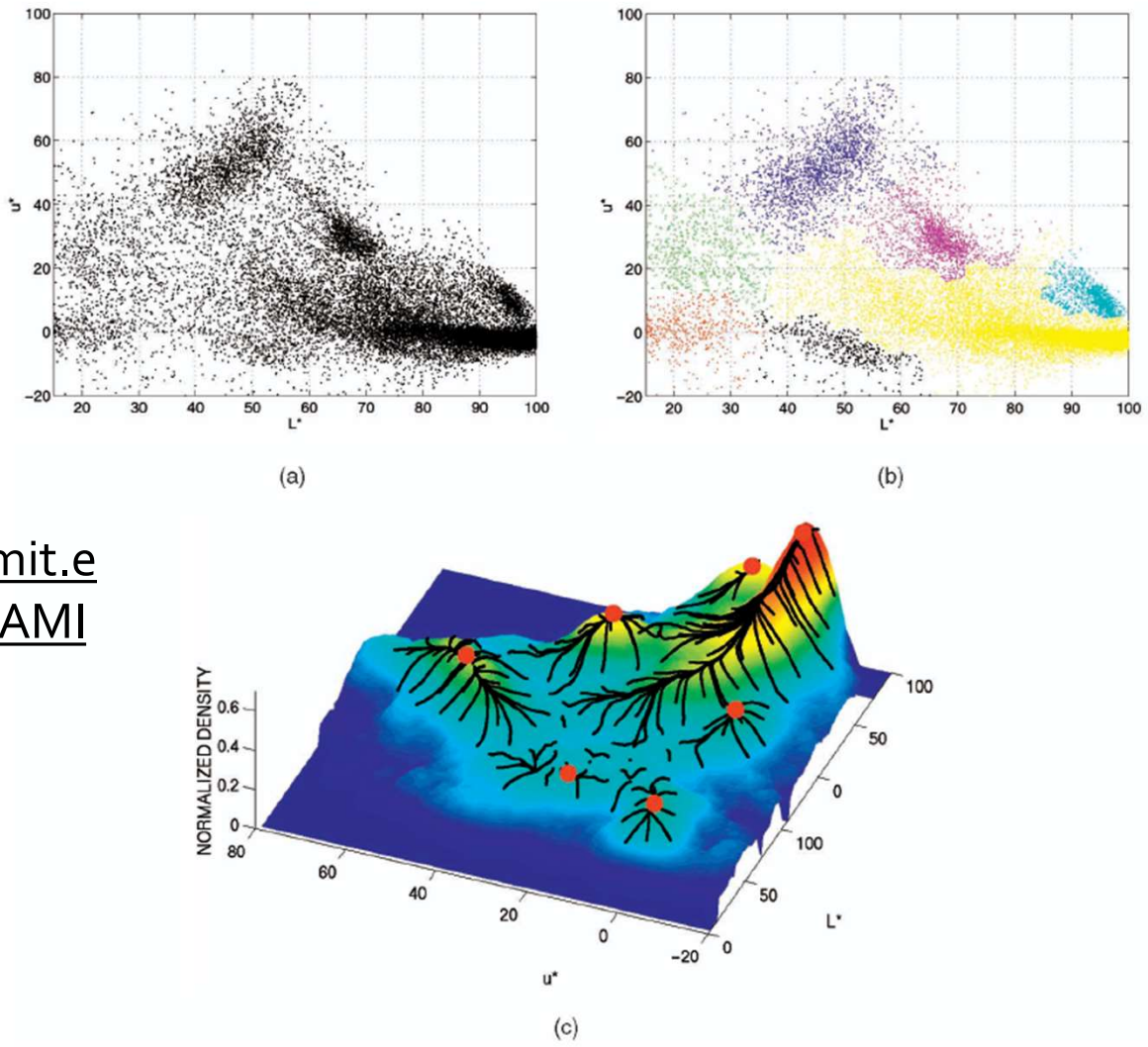
For $t = 0$, until convergence

$$\boldsymbol{x}^{(t+1)} = \frac{\sum_{i=1:\boldsymbol{x}_i \in N(\boldsymbol{x}^{(t)})}^{N} \boldsymbol{x}_i \exp\left(-\frac{\left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}_i\right\|^2}{2\sigma^2}\right)}{\sum_{i=1:\boldsymbol{x}_i \in N(\boldsymbol{x}^{(t)})}^{N} \exp\left(-\frac{\left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}_i\right\|^2}{2\sigma^2}\right)}, t = t+1$$

- This does not affect the results, because:

$$\text{if } \boldsymbol{x}^{(t)} \text{ and } \boldsymbol{x}_i \text{ are far apart, then } \exp\left(-\frac{\left\|\boldsymbol{x}^{(t)} - \boldsymbol{x}_i\right\|^2}{2\sigma^2}\right) \text{ is almost 0.}$$

Source: The paper "Mean shift: a robust approach towards feature space analysis", By Dorin Comaniciu and Peter Meer, IEEE Trans. PAMI, 2002



https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf

Fig. 2. Example of a 2D feature space analysis. (a) Two-dimensional data set of $110,400$ points representing the first two components of the L*u*v* space shown in Fig. 1b. (b) Decomposition obtained by running 159 mean shift procedures with different initializations. (c) Trajectories of the mean shift procedures drawn over the Epanechnikov density estimate computed for the same data set. The peaks retained for the final classification are marked with red dots.

Mean shift for image smoothing

- Consider a color image with N pixels. For each pixel (x_i, y_i), define the feature vector $\mathbf{v}_i = (x_i, y_i, R(x_i, y_i), G(x_i, y_i), B(x_i, y_i))$.

(1) Run the mean-shift procedure until convergence, starting from $\mathbf{v}_i$.

(2) Let the point to which the procedure converged, i.e. the mode of the pdf, be denoted by (x', y', R', G', B').

(3) Replace the intensity values at (x_i, y_i) by (R', G', B').

- Repeat steps 1,2,3 for all (x_i, y_i).
- **Note: many different pixels (x_i, y_i) will converge to the same point, i.e. the same value of (x', y', R', G', B').**

Mean shift for image smoothing

- The kernel densities for image smoothing are chosen to be Gaussian (other kernels are also possible, but their analysis turns out to be slightly more complicated).
- Usually, we use separate smoothing parameters for the spatial coordinates $\mathbf{x} = (x,y)$ and the intensity values $\mathbf{I} = (R,G,B)$.
- The smoothing parameters for $\mathbf{x}$ will be denoted as h_s and the smoothing parameter for $\mathbf{I}$ is denoted as h_r.

Mean shift for image smoothing

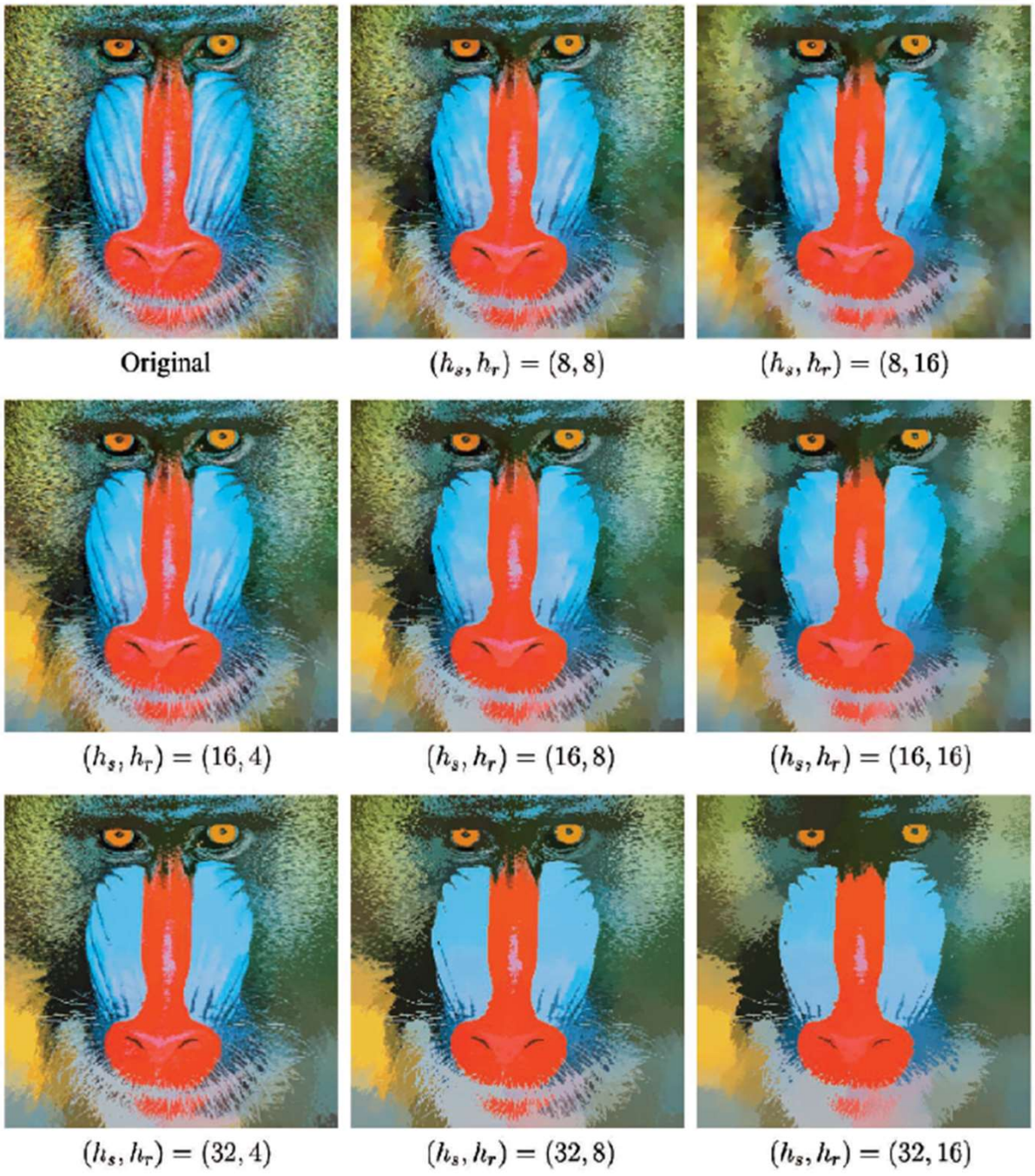
- The mean shift vector under these assumptions looks as follows:

$$\nabla p([\vec{x} \mid \vec{I}]) = \frac{1}{(2\pi)^{3/2} h_s^2 h_r^3 N} \left[\sum_{i=1}^{N} \exp\left(-\frac{\|\vec{x} - \vec{x}_i\|^2}{2h_s^2} \right) \exp\left(-\frac{\|\vec{I} - \vec{I}_i\|^2}{2h_r^2} \right) \right]$$

$$\left[\frac{\sum_{i=1}^{N} [\vec{x}_i \mid I_i] \exp\left(-\frac{\|\vec{x} - \vec{x}_i\|^2}{2h_s^2} \right) \exp\left(-\frac{\|\vec{I} - \vec{I}_i\|^2}{2h_r^2} \right)}{\sum_{i=1}^{N} \exp\left(-\frac{\|\vec{x} - \vec{x}_i\|^2}{2h_s^2} \right) \exp\left(-\frac{\|\vec{I} - \vec{I}_i\|^2}{2h_r^2} \right)} - \vec{x} \right]$$

$$= \frac{1}{(2\pi)^{3/2} h_s^2 h_r^3} p(\vec{x}) m(\vec{x}; h_s, h_r)$$

$$\therefore m(\vec{x}; h_s, h_r) \propto \frac{\nabla p(\vec{x})}{p(\vec{x})}$$



https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf

Mean shift for smoothing

- For each pixel k, do the following until convergence:

$$(\vec{x}_k \mid I_k) \leftarrow \left[\frac{\sum_{i \in N(k)} [\vec{x}_i \mid I_i] \exp\left(-\frac{\|\vec{x}_k - \vec{x}_i\|^2}{2h_s^2} \right) \exp\left(-\frac{\|I_k - \vec{I}_i\|^2}{2h_r^2} \right)}{\sum_{i \in N(k)} \exp\left(-\frac{\|\vec{x}_k - \vec{x}_i\|^2}{2h_s^2} \right) \exp\left(-\frac{\|I_k - \vec{I}_i\|^2}{2h_r^2} \right)} \right]$$

- Use these values of I_k in the smoothed image.
- Let the value to which (x_k, I_k) converged be called z_k.

Mean shift and bilateral filtering

- You would have (should have) noticed the similarity between mean shift and bilateral filtering.
- Mean shift is an **iterated version of bilateral filtering** where the **number of iterations of smoothing varies from pixel to pixel**, and this number is always chosen automatically by the mean shift procedure itself!

Mean shift segmentation

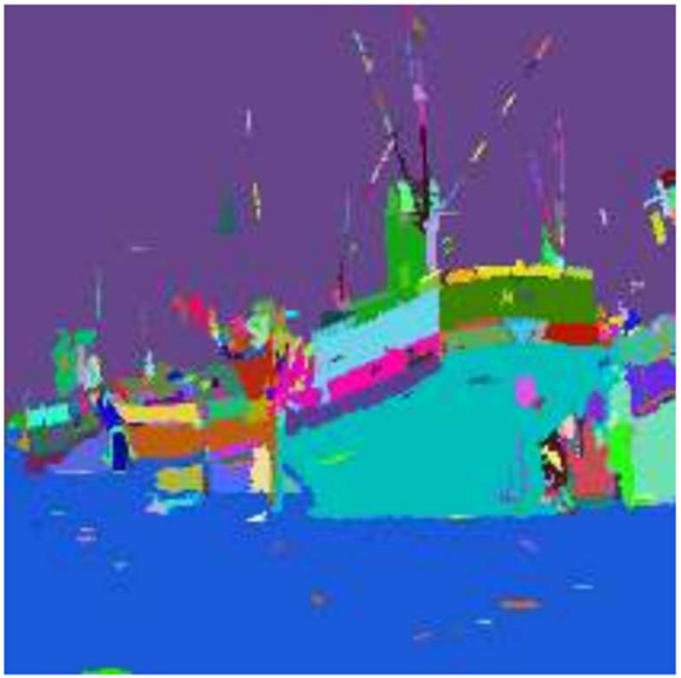
- Perform mean-shift filtering using parameters h_s and h_r, on the given image (say f(x,y)) to give you a filtered image g(x,y).
- Assign all those pixels that converged to the same point (say $\mathbf{z_k}$), to one and the same segment!
- Merge together all $\mathbf{z_k}$ which are closer than h_s in spatial domain and closer than h_r in intensity domain into a single cluster.

Fig. 6. *MIT* image. (a) Original. (b) Segmented $(h_s, h_r, M) = (8, 7, 20)$. (c) Region boundaries.

https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf

Fig. 7. *Room* image. (a) Original. (b) Region boundaries delineated with $(h_s, h_r, M) = (8, 5, 20)$, drawn over the input.

Boat $(h_s, h_r) = (4, 6)$



Boat $(h_s, h_r) = (4, 6)$

https://www.ipol.im/pub/art/2019/255/article_lr.pdf

Cameraman $(h_s, h_r) = (8, 8)$

Cameraman $(h_s, h_r) = (8, 8)$

House $(h_s, h_r) = (4, 6)$

House $(h_s, h_r) = (4, 6)$

https://www.ipol.im/pub/art/2019/255/article_lr.pdf

References

- This topic is not present in the book. You can refer to the following online tutorials:

http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUZEL1/MeanShift.pdf
http://en.wikipedia.org/wiki/Mean-shift

OR one of the following papers (which are not meant for the faint-hearted ☺)

- "Mean shift: a robust approach towards feature space analysis", by Dorin Comaniciu and Peter Meer, IEEE Trans. PAMI, 2002.
- Fukunaga, Keinosuke; Larry D. Hostetler (January 1975). "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition". *IEEE Transactions on Information Theory* (IEEE) **21** (1): 32–40.
- Cheng, Yizong (August 1995). "Mean Shift, Mode Seeking, and Clustering". *IEEE Transactions on Pattern Analysis and Machine Intelligence* (IEEE) **17** (8): 790–799