Advanced Clustering Methods

MeanShift, MedoidShift and QuickShift

References:


MeanShift Clustering

nonparametric mode seeking
Background: Parzen Estimation
(Aka Kernel Density Estimation)

Consider a mathematical model of how histograms are formed. Assume continuous data points.
Parzen Estimation
(Aka Kernel Density Estimation)

Consider a mathematical model of how histograms are formed. Assume continuous data points.

Convolve with box filter of width \( w \) (e.g. [1 1 1])

Subsample the results, with spacing \( w \)

Resulting value at point \( u \) represents count of data points falling in range \( u-w/2 \) to \( u+w/2 \)
Example Histograms

Box filter

\[ [1 \ 1 \ 1] \]

\[ [1 \ 1 \ 1 \ 1 \ 1] \]

\[ [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \] \]

Increased smoothing

Why Formulate it This Way?

• Generalize from box filter to other filters (for example Gaussian)
• Gaussian acts as a smoothing filter.

• Key idea: produces a nonparametric estimate of a continuous distribution from a discrete set of points.
Kernel Density Estimation

- **Parzen windows**: Approximate probability density by estimating local density of points (same idea as a histogram)
  - Convolve points with window/kernel function (e.g., Gaussian) using scale parameter (e.g., sigma)

from Hastie *et al.*
Density Estimation at Different Scales

• Example: Density estimates for 5 data points with differently-scaled kernels
• Scale influences accuracy vs. generality (overfitting)
Smoothing Scale Selection

• Unresolved issue: how to pick the scale (sigma value) of the smoothing filter
• Answer for now: a user-settable parameter

from Duda et al.
Mean-Shift

The mean-shift algorithm is a hill-climbing algorithm that seeks modes of a density without explicitly computing that density.

The density is implicitly represented by raw samples and a kernel function. The density is the one that would be computed if Parzen estimation was applied to the data with the given kernel.
What is Mean Shift?

A tool for:
Finding modes in a set of data samples, manifesting an underlying probability density function (PDF) in $\mathbb{R}^N$

PDF in feature space
- Color space
- Scale space
- Actually any feature space you can conceive
- …

Ukrainitz & Sarel, Weizmann
Intuitive Description

Region of interest
Center of mass
Mean Shift vector

Objective: Find the densest region

Ukrainitz & Sarel, Weizmann
Intuitive Description

Objective: Find the densest region

Ukrainitz & Sarel, Weizmann
Intuitive Description

**Objective**: Find the densest region

Ukrainitz & Sarel, Weizmann
Intuitive Description

Objective: Find the densest region

Ukrainitz & Sarel, Weizmann
Intuitive Description

Objective: Find the densest region

Ukrainitz & Sarel, Weizmann
Intuitive Description

Objective: Find the densest region

Ukrainitz & Sare, Weizmann
Intuitive Description

Objective: Find the densest region

Ukrainitz&Sarel, Weizmann
MeanShift as Mode Seeking

Parzen density estimation
MeanShift as Mode Seeking

Seeking the mode from an initial point $y_1$
Construct a tight convex lower bound $b$ at $y_1$  [$b(y_1)=P(y_1)$]
Find $y_2$ to maximize $b$
MeanShift as Mode Seeking

Note, $P(y_2) \geq b(y_2) > b(y_1) = P(y_1)$
Therefore $P(y_2) > P(y_1)$
MeanShift as Mode Seeking

$P(x)$

Move to $y_2$ and repeat until you reach a mode
MeanShift in Equations

Parzen estimate using \( \{x_1, x_2, ..., x_N\} \)

\[
\rho(y) = \frac{1}{N} \sum_{j=1}^{N} \phi(||y-x_j||^2)
\]

where \( \Phi(r) \) is a convex, symmetric windowing (weighting) kernel; e.g. \( \Phi(r) = \exp(-r) \)

Form lower bound of each \( \Phi_j \) by expanding about \( r_0 = ||y_{\text{old}}-x_j||^2 \)

\[
\phi(r) \geq \phi(r_0) + (r-r_0)f'(r_0) + \text{h.o.t.}
\]

note: first-order taylor series expansion
MeanShift in Equations

Lower bound of $\Phi$ at $r_0$

$$\phi(r) \geq \phi(r_0) + (r - r_0) \dot{\phi}(r_0)$$

now let

$$r_0 = \|y_{0k} - x_j\|^2$$
$$r = \|y - x_j\|^2$$

and substitute into Parzen equation

to get a quadratic lower bound (wrt $y$) on the Parzen density function

$$p(y) \geq p(y_{0k}) + \frac{1}{n} \sum_{j=1}^{n} (\|y - x_j\|^2 - \alpha_j) \omega_j$$

where

$$\omega_j = \dot{\phi}(\|y_{0k} - x_j\|^2)$$
$$\alpha_j = \|y_{0k} - x_j\|^2$$
MeanShift in Equations

Quadratic lower bound on the Parzen density function

\[
P(y) \geq P(y_{old}) + \frac{1}{n} \sum_{j=1}^{n} (||y-x_j||^2 - \alpha_j) w_j
\]

Want to find \(y\) to maximize this. We therefore need to maximize

\[
\arg\max_y \quad \frac{1}{n} \sum_{j=1}^{n} (y-x_j)^T(y-x_j) w_j
\]

\[= \frac{1}{n} \sum_j ((y^T y - 2y^T x_j + x_j^T x_j) w_j)
\]

Take derivative wrt \(y\), set to 0, and solve

\[
\frac{\partial}{\partial y} = \frac{1}{n} \sum_j (2y - 2x_j) w_j = 0
\]

\[\Rightarrow \quad \sum_j w_j y = \sum w_j x_j
\]

\[y = \frac{\sum w_j x_j}{\sum w_j}
\]

(mean-shift update eqn)
MeanShift Clustering

For each data point $x_i$

Let $y_i(0) = x_i$ ; $t = 0$

Repeat

Compute weights $w_1, w_2, \ldots, w_N$ for all data points, using $y_i(t)$

Compute updated location $y_i(t+1)$ using mean-shift update eqn

If ( $y_i(t)$ and $y_i(t+1)$ are “close enough” )
   declare convergence to a mode and exit
else
   $t = t + 1$;
end (repeat)

number distinct modes found = number clusters
points that converge to “same” mode are labeled as belonging to one cluster
What is MedoidShift?

Instead of solving for continuous-valued $y$ to maximize the quadratic lower bound on the Parzen density estimate,

consider only a discrete set of $y$ locations, taken from the initial set of data points $\{x_1, x_2, \ldots, x_N\}$

That is, for the first step towards the mode

$$
\gamma_i(1) = \arg\max_{\gamma \in \{x_1, x_2, \ldots, x_N\}} \frac{1}{N} \sum_{j=1}^{N} \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{1}{2} \frac{(\gamma - x_j)^2}{\hat{\sigma}^2} \right)
$$

$y_i(0) = x_i$

$y$ is now chosen from a discrete set of data points
MedoidShift

Implications of this strategy for clustering:

Since we are choosing the best \( y \) from among \( \{x_1, x_2, \ldots, x_N\} \), \( y_i(1) \) will be one of the original data points.

Therefore, we only need to compute a single step starting from each data point, because \( y_i(2) \) will then be the result of \( y_{yi(1)}(1) \).
MedoidShift

Different clusters are found as separate connected components (trees, actually... so can do efficient tree traversal to locate all points in each cluster.

Also note: no need for heuristic distance thresholds to test for convergence to a mode, or to determine membership in a cluster.
MedoidShift behavior asymptotically approaches MeanShift behavior as number of data samples approaches infinity.
Advantages of MedoidShift

• Automatically determines the number of clusters (as does mean-shift)

• Previous computations can be reused when new samples are added or old samples are deleted (good for incremental clustering applications)

• Can work in domains where only distances between samples are defined (e.g. EMD or ISOMAP); there is no need to compute a mean.

• No need for heuristic terminating conditions (don’t need to use a distance threshold to determine convergence)
MeanShift vs MedoidShift

Figure 8. Clustering results obtained by using mean-shift (left) in Euclidean space and medoid shift (right) using the manifold distance function described in [18].
Drawbacks of MedoidShift

• $O(N^3)$ [although not really – Vedaldi paper]

• Can fail to find all the modes of a density, thereby causing over-fragmentation
Iterated MedoidShift

- keep a count of how many points move to $x_j$ and apply a new iteration of medoidshift with points weighted by that count.

Figure 2. Left to right: Clustering using medoidshift after 1, 2, and 3 iterations, and the final labels for each point.

- Each iteration is essentially using a smoother Parzen estimate
In future work, we intend to investigate a further speed up. The requirement that for the current location $y_k$, $y_{k+1}$ be the data point that minimizes Equation 2 is not strictly required for convergence. The sufficient condition in Equation 7 in Theorem 2.1 is simply that the new point $y_{k+1}$ have a score better than $y_k$. If this condition is used instead of the exact condition the computation can be terminated early. An implementation showed that the computational saving obtained from this was roughly 80% greater than that of the exact algorithm. Further investigation is needed to determine the degree of approximation error. The extension
Contributions:

• If using Euclidean distance, MedoidShift can actually be implemented in $O(N^2)$

• generalize to non-Euclidean distances using kernel trick

• show that MedoidShift does not always find all modes

• propose QuickShift, an alternative approach that is simple, fast, and yields a one-parameter family of clustering results that explicitly represent the tradeoff between over- and under-fragmentation.
Complexity of MedoidShift

\[ y_i(1) = \underset{j \in \{x_1, x_2, \ldots, x_n\}}{\text{argmax}} \quad \frac{1}{N} \sum_{j=1}^{N} (\| y - x_j \|^{2} \phi(\| x_i - x_j \|^{2})) \]

\[ = \frac{1}{N} \sum_{j=1}^{N} d^2(y, x_j) \phi(d^2(x_i, x_j)) \]

Define matrices (N x N matrices)

\[ D_{kj} = d^2(x_k, x_j) \]
\[ F_{ji} = \phi(d^2(x_j, x_i)) \]

Then

\[ y_i(1) = \underset{k}{\text{argmax}} \sum_{j=1}^{N} D_{kj} F_{ji} \]

note: define N x N matrices \( D = [D_{kj}] \), \( F = [F_{ji}] \)
then matrix multiply does the summation over \( j \)
and argmax is looking for max element in col \( i \)
Complexity of MedoidShift

\[
\text{naive matrix multiply}
\]

\[
\begin{align*}
A & \quad B & \quad \mathcal{O}(nmp) \\
\end{align*}
\]

so \( DF = \mathcal{O}(N^3) \) Then another \( \mathcal{O}(N^2) \) to find maximum of each column

However, if using Euclidean distance (which is typical), the computation time is better than this.
Euclidean MedoidShift

Let \( \mathbf{x} = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix} \) e.g. \( \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \)

Let \( \mathbf{n} = (\mathbf{x}^T \odot \mathbf{x}^T) \mathbf{1} = \begin{bmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ x_3^2 + y_3^2 \end{bmatrix} = \begin{bmatrix} x_1^T \mathbf{x} \\ x_2^T \mathbf{x} \\ x_3^T \mathbf{x} \end{bmatrix} \)

then we can write the squared Euclidean distance matrix \( \mathbf{D} \) as

\[
\mathbf{D} = \mathbf{1} \mathbf{n}^T + \mathbf{n} \mathbf{1}^T - 2 \mathbf{x}^T \mathbf{x}
\]

\[
= \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & x_1^T x_3 \\ x_2^T x_1 & x_2^T x_2 & x_2^T x_3 \\ x_3^T x_1 & x_3^T x_2 & x_3^T x_3 \end{bmatrix} + \begin{bmatrix} x_1^T x_1 & x_1^T x_1 & x_1^T x_1 \\ x_2^T x_2 & x_2^T x_2 & x_2^T x_2 \\ x_3^T x_3 & x_3^T x_3 & x_3^T x_3 \end{bmatrix} - 2 \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & x_1^T x_3 \\ x_2^T x_2 & x_2^T x_2 & x_2^T x_3 \\ x_3^T x_3 & x_3^T x_2 & x_3^T x_3 \end{bmatrix}
\]

\[\| \mathbf{x}_i - \mathbf{x}_j \|^2 = x_i^T x_i + x_j^T x_j - 2 x_i^T x_j\]
Euclidean MedoidShift

Therefore

\[ DF = (1n^T + n1^T - 2x^T x)^F \]

\[ = 1(n^T x^F) + n(4^T F) - 2x^T(x F) \]

This has constant columns and can be ignored

We have to compute this

But these quantities can all be computed in \( O(N^2) \) time

\[ n(1^T F) \quad n^T(x F) \]

\[ n = (n x^1 x^2 \ldots x^m) n \]

All take \( \Theta(d N^2) \) operations \( d \) is dimension of each data point
Kernel MedoidShift

Insight: medoidshift is defined in terms of pairwise distances between data points.

Therefore, using the "kernel trick", we can generalize it to handle non-Euclidean distances defined by a kernel matrix $K$ (similar to how many machine learning algorithms are "kernelized")
Kernel MedoidShift

Assume a positive definite kernel matrix \( K \) such that

\[
d^2(x, y) = K(x, x) + K(y, y) - 2 K(x, y)
\]

Then we still have

\[
\mathbf{D} = \mathbf{1} \mathbf{n}^T + \mathbf{n} \mathbf{1}^T - 2 \mathbf{K} \quad \text{where } \mathbf{n} = \text{diag}(\mathbf{K})
\]

\[
\mathbf{DF} \propto \mathbf{n}(\mathbf{1}^T \mathbf{F}) - 2 \mathbf{K} \mathbf{F}
\]

and we can apply medoidshift using this DF matrix instead
Kernel MedoidShift

note:

$$DF \propto n(1^TF) - 2KF$$

here $n \times n$ $n \times n$
so we are back to $O(n^3)$

But

if we can find a low-rank decomposition

$$K \approx G^T G$$

of rank 1

Then $KF \approx G^T (G F)$

and we are able to compute in $O(dN^2)$

[cost of decomposing $K$ is typically $O(d^2N^2)$]
What is QuickShift?

A combination of the above kernel trick and...

instead of choosing the data point that optimizes the lower bound function $b(x)$ of the Parzen estimate $P(x)$

choose the closest data point that yields a higher value of $P(x)$ directly

\[
\gamma_i(1) = \arg\min_{j} D(x_i, x_j) \quad \text{s.t.} \quad P(x_j) > P(x_i)
\]

\[
P(x_i) = \frac{1}{N} \sum_{j=1}^{N} \phi(D(x_i, x_j))
\]
QuickShift vs MedoidShift

• Recall our failure case for medoidshift
Comparisons

**meanshift**  
**medoidshift**  
**quickshift**

Fig. 1. Mode seeking algorithms. Comparison of different mode seeking algorithms (Sect. 2) on a toy problem. The black dots represent (some of) the data points $x_i \in X \subset \mathbb{R}^2$ and the intensity of the image is proportional to the Parzen density estimate $P(x)$. **Left.** Mean shift moves the points uphill towards the mode approximately following the gradient. **Middle.** Medoid shift approximates mean shift trajectories by connecting data points. For reasons explained in the text and in Fig. 2, medoid shifts are constrained to connect points comprised in the red circles. This disconnects portions of the space where the data is sparse, and can be alleviated (but not solved) by iterating the procedure (Fig. 2). **Right.** Quick shift (Sect. 3) seeks the energy modes by connecting nearest neighbors at higher energy levels, trading-off mode over- and under-fragmentation.
Quickshift Benefits

• Similar benefits to medoidshift, plus additional ones:

• All points are in one connected tree. You can therefore explore various clusterings by choosing a threshold $T$ and pruning edges such that $d(x_i, x_j) < T$

• Runs much faster in practice than either meanshift or medoidshift
Applications

image segmentation by clustering \((r,g,b)\), or \((r,g,b,x,y)\), or whatever your favorite color space is
Applications

manifold clustering using ISOMAP distance

Fig. 4. Clustering on a manifold. By using kernel ISOMAP we can apply kernel mean and medoid shift to cluster points on a manifold. For the sake of illustration, we reproduce an example from [20]. From left to right: Kernel mean shift (7.8s), non-iterated kernel medoid shift (0.18s), iterated kernel medoid shift (0.48s), quick shift (0.12s). We project the kernel space to three dimensions $d = 3$ as the residual dimensions are irrelevant. All algorithms but non-iterated medoid shift segment the modes successfully. Compared to [20], medoid shift has complexity $O(dN^2)$, (with a small constant and $d = 3 \ll N$) instead of $O(N^3)$ (small constant) or $O(N^{2.38})$ (large constant)
Applications

clustering bag-of-features signatures using Chi-squared kernel (distances between histograms) to discover visual "categories" in Caltech-4

clusters in 400D space (visualized by projection to rank 2)

5 discovered categories. Ground-truth category "airplanes" is split into two clusters

Fig. 6. Automatic visual categorization. We use kernel mean shift to cluster bag-of-features image descriptors of 1600 images from Caltech-4 (four visual categories: airplanes, motorbikes, faces, cars). Top. From left to right, iterations of kernel mean shift on the bag-of-features signatures. We plot the first two dimensions of the rank-reduced kernel space (z vectors) and color the points based on the ground truth labels. In the rightmost panel the data converged to five points, but we artificially added random jitter to visualize the composition of the clusters. Bottom. Samples from the five clusters found (notice that airplane are divided in two categories). We also report the clustering quality, as the percentage of correct labels compared to the ground truth (we merge the two airplanes categories into one), and the execution time. We use basic implementations of the algorithms, although several optimizations are possible.
Applications

Clustering images using a bhattacharya coefficient on 5D histograms composed of \((L,a,b,x,y)\) values

Figure 10. Clustering a collection of images. Each row shows images selected from a single cluster found by medoidshift

Figure 11. Key-frames selected from a video of 1500 frames. Each key-frame corresponds to a mode as estimated by our algorithm.