More on Mean-shift

R.Collins, CSE, PSU
CSE598G Spring 2006

Recall: Kernel Density Estimation

For kernel H with bandwidth h, estimated function value \( f(x) \) at location \( x \) is

\[
\hat{f}(x) = \sum \frac{1}{n} H(x - x_i)
\]

some sample kernels:

Radially Symmetric Kernels

Height at point is function only of distance from center

Can be written in terms of a 1D Profile function that is
is a function of the radius (we will use squared radius below)

\[
\hat{f}(r) = \frac{1}{h^2} \int H(r - r') r' dr'
\]

Examples

- Epanechnikov
- Biweight
- Gaussian

- Flat

\[
\text{repl} \times \frac{1}{r} \rightarrow \frac{1}{r^2}
\]

Kernel-Shadow Pairs

Given a convolution kernel H, what is the corresponding mean-shift kernel K?
Perform change of variables \( r = ||x_i - x|| \)
Rewrite \( H(x) = H(||x||^2) = h(r) \).
Then kernel K must satisfy \( h'(r) = -c k(r) \)

Examples

- Epanechnikov
- Biweight
- Gaussian

Kernel-Shadow Pairs

Given a convolution kernel H, what is the corresponding mean-shift kernel K?
Perform change of variables \( r = ||x|| \)
Rewrite \( H(x) = h(||x||^2) = h(r) \).
Then kernel K must satisfy \( h'(r) = -c k(r) \)

Self-replicating:
Mean-Shift and Gradient Ascent

We will derive an explicit equation relating the mean-shift procedure using kernel \( K \) with gradient ascent on the KDE surface formed by using the shadow kernel \( H \).

Gradient of KDE

\[
\nabla \hat{f}(x) = \sum \nabla \hat{h}(x_i - x) \cdot (x_i - x)
\]

Sidebar derivation:

change of variables

\[
\nabla \hat{h}(x) = \frac{\partial h}{\partial x} = \frac{\partial (\hat{h}(x) \cdot (x_i - x))}{\partial x}
\]

chain rule

\[
- \hat{h}(x) \frac{\partial (x_i - x)}{\partial x} + \frac{\partial \hat{h}(x)}{\partial x} \cdot (x_i - x)
\]

change variables back

This is the relationship we wanted to derive.

Cont.

weighted center of mass

\[
\frac{\sum \hat{h}(x_i - x) \cdot x_i}{\sum \hat{h}(x_i - x)} - x
\]

mean shift vector

• same direction as gradient of KDE

• step size is inversely proportional to KDE

• mean-shift performs gradient ascent with adaptive step size

Generalizing to Weighted Points

Let each point sample \( x_i \) have an associated nonnegative weight \( w(x_i) \). Can rederive the equations with \( w(x_i) \) factors in them:

KDE using shadow kernel \( H \)

\[
\hat{f}(x) = \sum w(x_i) \cdot \hat{h}(x_i - x)
\]

KDE using kernel \( K \)

\[
\hat{f}(x) = \sum w(x_i) \cdot k(x_i - x)
\]

mean shift vector is still a gradient ascent process

This is important for running on images. Since pixels form a lattice, spatial density of samples is fixed, so need a weight to denote sample density at each point.
Mean-Shift Tracking

Let pixels form a uniform grid of data points, each with a weight (pixel value) proportional to the “likelihood” that the pixel is on the object we want to track. Perform standard mean-shift algorithm using this weighted set of points.

\[
\Delta x = \frac{\sum_i K(x_i - x) \, w(x_i) \, (x_i - x)}{\sum_i K(x_i - x) \, w(x_i)}
\]

Nice Property

Running mean-shift with kernel \( K \) on weight image \( w \) is equivalent to performing gradient ascent in a (virtual) image formed by convolving \( w \) with some “shadow” kernel \( H \).

Computational savings: only have to compute convolution values at the relatively small number of points you visit on the path to the mode.

Mean-shift on Weight Images

Ideally, we want an indicator function that returns 1 for pixels on the object we are tracking, and 0 for all other pixels.

Instead, we compute likelihood maps where the value at a pixel is proportional to the likelihood that the pixel comes from the object we are tracking.

Computation of likelihood can be based on:
- color
- texture
- shape (boundary)
- predicted location

Claim: these weight images are all the mean-shift algorithm “sees”, whether they be explicitly computed (e.g. Bradski) or implicitly computed (e.g. Comaniciu, Ramesh and Meer).

Explicit Weight Images

Histogram backprojection

Histogram is an empirical estimate of \( p(\text{color} \mid \text{object}) = p(\text{c} \mid o) \)

Bayes rule says \( p(o \mid c) = \frac{p(c \mid o) \, p(o)}{p(c)} \)

Simplistic approx: assume \( (p(o)/p(c)) \) is constant. Then \( p(o|c) = p(c|o) \).

Use histogram \( h \) as a lookup table to set pixel values in the weight image. (If pixel maps to bucket \( i \) in histogram \( h \), set weight for that pixel to \( h(i) \)).

Sidebar: Swain and Ballard, 1991

Using color histograms for recognition.

- Works surprisingly well
- In the first paper (1991), 66 objects could be recognized almost without errors

[Swain & Ballard, 1991]

Swain and Ballard

- Histogram backprojection:
  - “Where in the image are the colors being looked for?”
  - Query: object with histogram \( M \)
  - Given: image with histogram \( I \)
  - Compute the “ratio histogram”: \( R = \min \left( \frac{M}{I}, 1 \right) \)

  \( R \) reveals how important an object color is, relative to the current image. This value is projected back into the image (i.e., the image values are replaced by the values of \( R \) that they index).

  The result image is convolved with a circular mask the size of the target object. Peaks in the convolved image indicate detected objects.

  does this sound familiar?
Swain and Ballard

Object Localization

- Example result after backprojection
  - Looking for blue pullover...

Note: relationship between recognition and tracking. This will come up again later.

Sidebar: Jones and Rehg, 2002


General Idea:
- Learn skin color distribution from examples
- Learn distributions of skin and nonskin color
- Nonparametric distributions: color histograms
- Bayesian classification of skin pixels

Learned Distributions

Skin color

Non-Skin color

Likelihood Ratio

Label a pixel skin if
\[
\frac{P(\text{rgb} \mid \text{skin})}{P(\text{rgb} \mid \text{not skin})} > \Theta
\]

\[\Theta = \frac{\text{(cost of false positive)} \cdot P(\text{seeing not skin})}{\text{(cost of false negative)} \cdot P(\text{seeing skin})}\]

\[0 \leq \Theta \leq 1\]
Sample Application: HCI
Haiying Guan, Matthew Turk, UCSB

relevance to blob tracking is clear

Implicit Weight Images

Sometimes the weight image is not explicitly created. An example is Comaniciu, Ramesh and Meer. Here, the “weight image” is embedded in the procedure (taking derivatives of bhattacharyya coeff with respect to image location of the window of samples).

Interesting point: their weight image changes between iterations of mean-shift, as compared to iterating to convergence on an explicit weight image!

Comaniciu et.al.

Color Histogram Representation:

Distance between histograms measured by:

\[
d(y) = \sqrt{1 - \rho(p(y), q)}
\]

where \(\rho(p(y), q) = \sum_{i=1}^{m} \sqrt{p_i(y)q_i(y)}\), Bhattacharyya coefficient

Sometimes the weight image is not explicitly created. An example is Comaniciu, Ramesh and Meer. Here, the “weight image” is embedded in the procedure (taking derivatives of bhattacharyya coeff with respect to image location of the window of samples).

Interesting point: their weight image changes between iterations of mean-shift, as compared to iterating to convergence on an explicit weight image!

Comaniciu et.al.

the histograms are computed via Parzen estimation:

\[
q_k = C \sum_{i=1}^{m} \delta\left(\frac{[x_i] - y}{h}\right),
\]

where \(k\) is some radially symmetric smoothing kernel (profile)

This is Important!

• interpolates histograms in off-lattice locations
• makes histogram \(p_y\) differentiable wrt to \(y\)

Comaniciu et.al.

Let’s look at the weight terms more closely. For each pixel \(x_i\):

\[
w_i = \sum_{i=1}^{m} \delta\left(\frac{[x_i] - y}{h}\right)
\]

which is just standard mean-shift on (implicit) weight image \(w_i\)

Let’s look at the weight terms more closely. For each pixel \(x_i\):

\[
w_i = \sum_{i=1}^{m} \delta\left(\frac{[x_i] - y}{h}\right)
\]

This term is only 1 once in the summation

If pixel \(x_i\)’s value maps to histogram bucket B, then \(w_i = \sqrt{q_k / p_k(y_i)}\)
So if model histogram is $q_1, q_2, ..., q_m$
and current data histogram is $p_1, p_2, ..., p_m$
form weights $q_1/p_1, q_2/p_2, ..., q_m/p_m$
and then do “histogram backprojection” of these values into
the image to get the weight image $w_i$
also note, $p_1, p_2, ..., p_m$ changes at each iteration, and
therefore so does the weight image $w_i$

Qualitative Intuition

Assume some object that
is 60% red and 40% green
$q_1 = .6, q_2 = .4, q_i = 0$ for all other $i$
If we just did histogram backprojection of these
likelihood values (ala Bradski), we would get a

Mean shift does a weighted center of mass
computation at each interation

Mean shift window will be biased towards the
region of red pixels, since they have higher weight

Qualitative Intuition

Now use Comaniciu et.al.’s weights
Let’s say the data histogram is perfectly located
$q_1 = .6, q_2 = .4, q_i = 0$ for all other $i$
$p_1 = .6, p_2 = .4, p_i = 0$ for all other $i$

Resulting weight image:

This is the indicator
function image we
always hope for!

Qualitative Intuition

Say we have too little percentage of red in data hist
$q_1 = .6, q_2 = .4, q_i = 0$ for all other $i$
$p_1 = .5, p_2 = .5, p_i = 0$ for all other $i$

$w_1 = \sqrt{.6/.5}, w_2 = \sqrt{.4/.5}, w_i = 0$ for all other $i$

So red pixels will be favored in center of mass
computation, hopefully causing more of them
to be included at the next iteration.

Qualitative Intuition

Say we have very little percentage of red in data hist
$q_1 = .6, q_2 = .4, q_i = 0$ for all other $i$
$p_1 = .2, p_2 = .8, p_i = 0$ for all other $i$

$w_1 = \sqrt{.6/.2}, w_2 = \sqrt{.4/.8}, w_i = 0$ for all other $i$

So red pixels will be even more highly weighted
relative to green pixels.
Qualitative Intuition

Say we have too much percentage of red in data hist

\[ q_1 = .6, \quad q_2 = .4, \quad q_i = 0 \text{ for all other } i \]
\[ p_1 = .7, \quad p_2 = .3, \quad p_i = 0 \text{ for all other } i \]
\[ w_1 = \sqrt{.6/.7}, \quad w_2 = \sqrt{.4/.3}, \quad w_i = 0 \text{ for all other } i \]

<1 \quad \text{if } i = 1

>1 \quad \text{if } i > 1

So green pixels will now be favored.

Other Features

We’ve only talked about color, but of course we could use histograms of other features like edge orientations (or any filter-bank response)

However, one point I’ve been trying to make today is that we don’t need histograms at all. We just need a way to label pixels with the likelihood that they belong to the object (rather than the background).

That is, we just need to be able to specify a weight image, either explicitly or implicitly.

Discuss using mean-shift to find modes in correlation surfaces

Explicit is easy. Implicit??

Brief overview of our two “bleeding edge” topics

Choose discussion leaders.