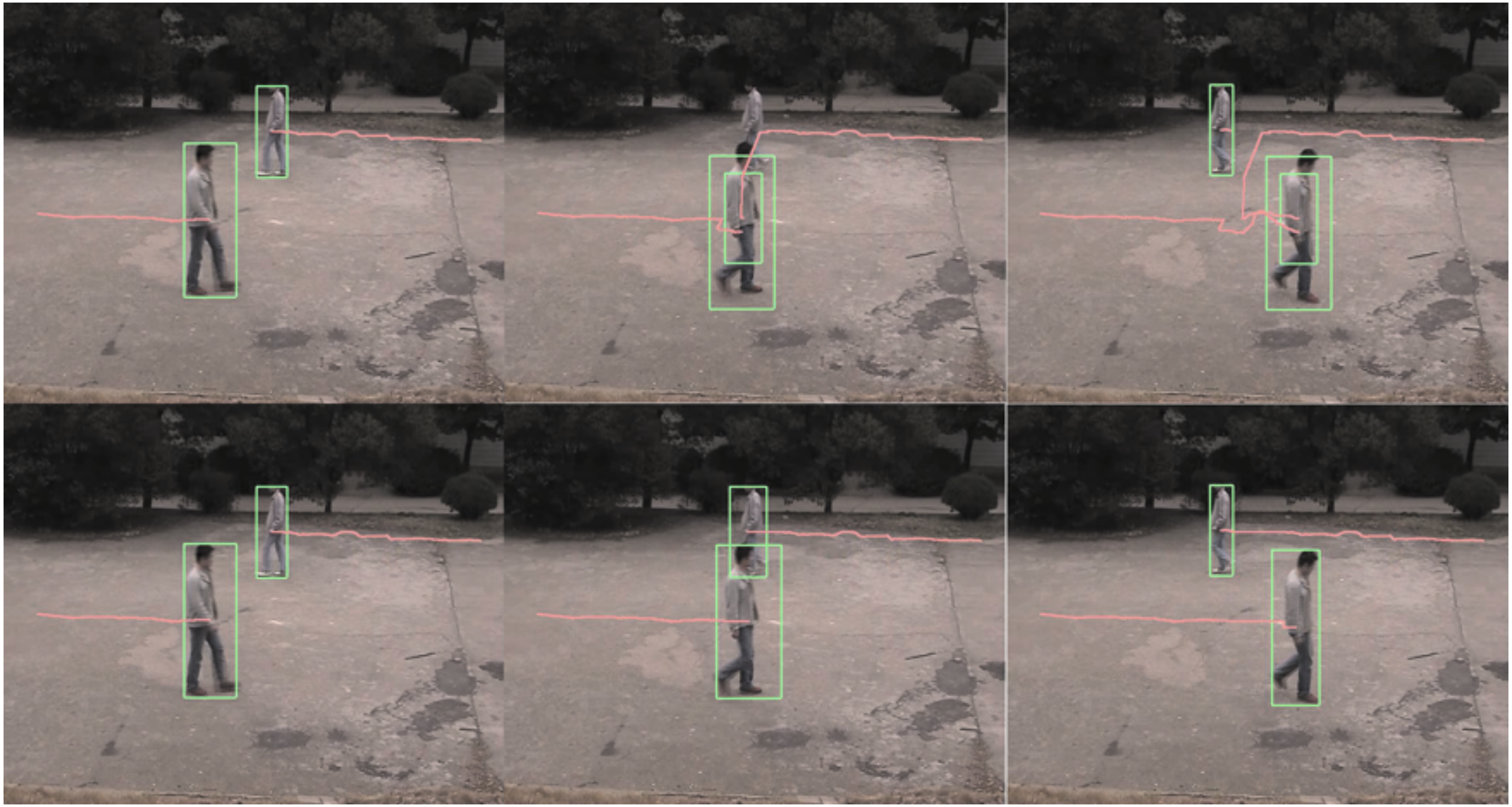


Mean-Shift Tracker

16-385 Computer Vision (Kris Kitani)

Carnegie Mellon University


Nationale-Nederlanden
jackpot €5 milj.
ING

Mean Shift Algorithm

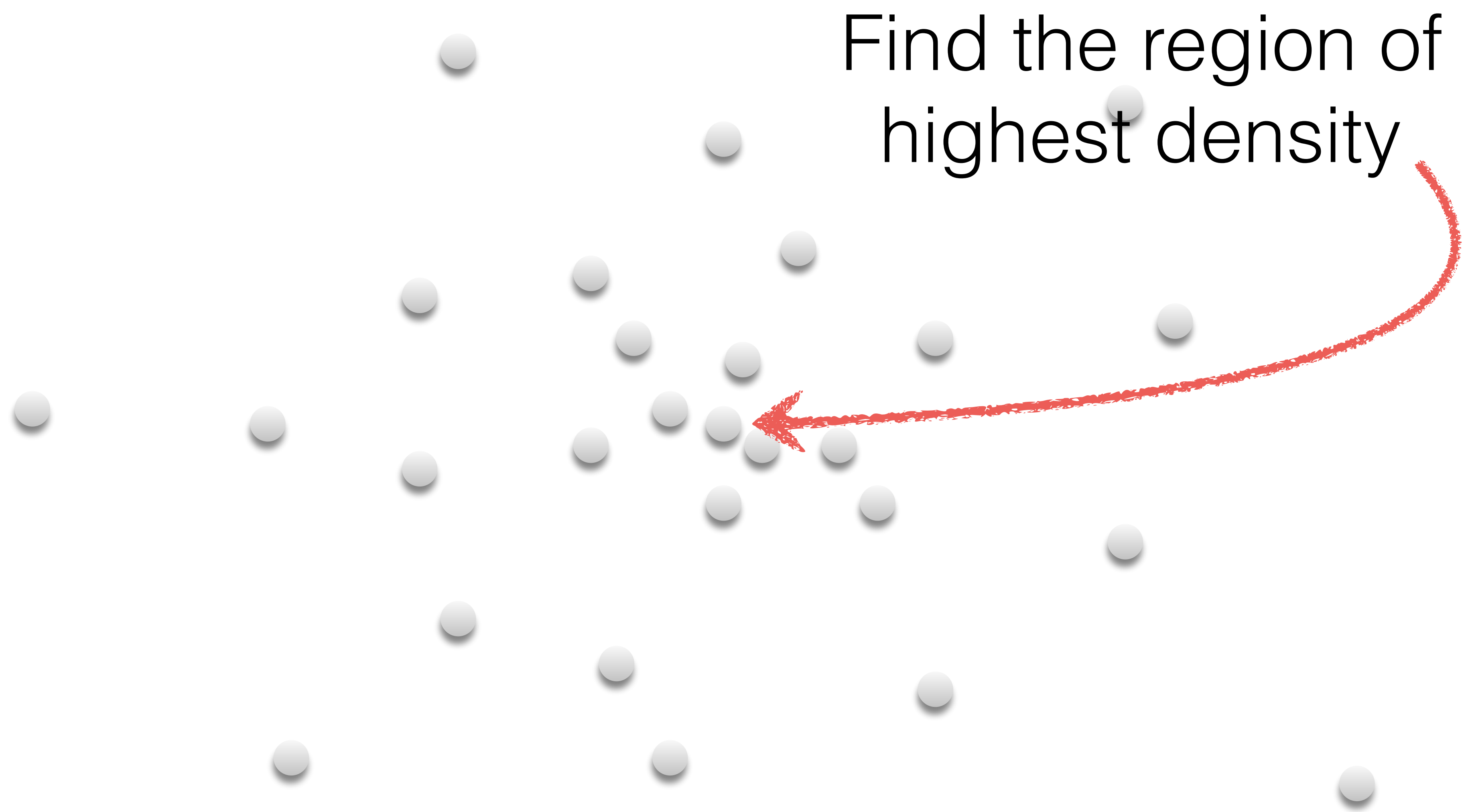
A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

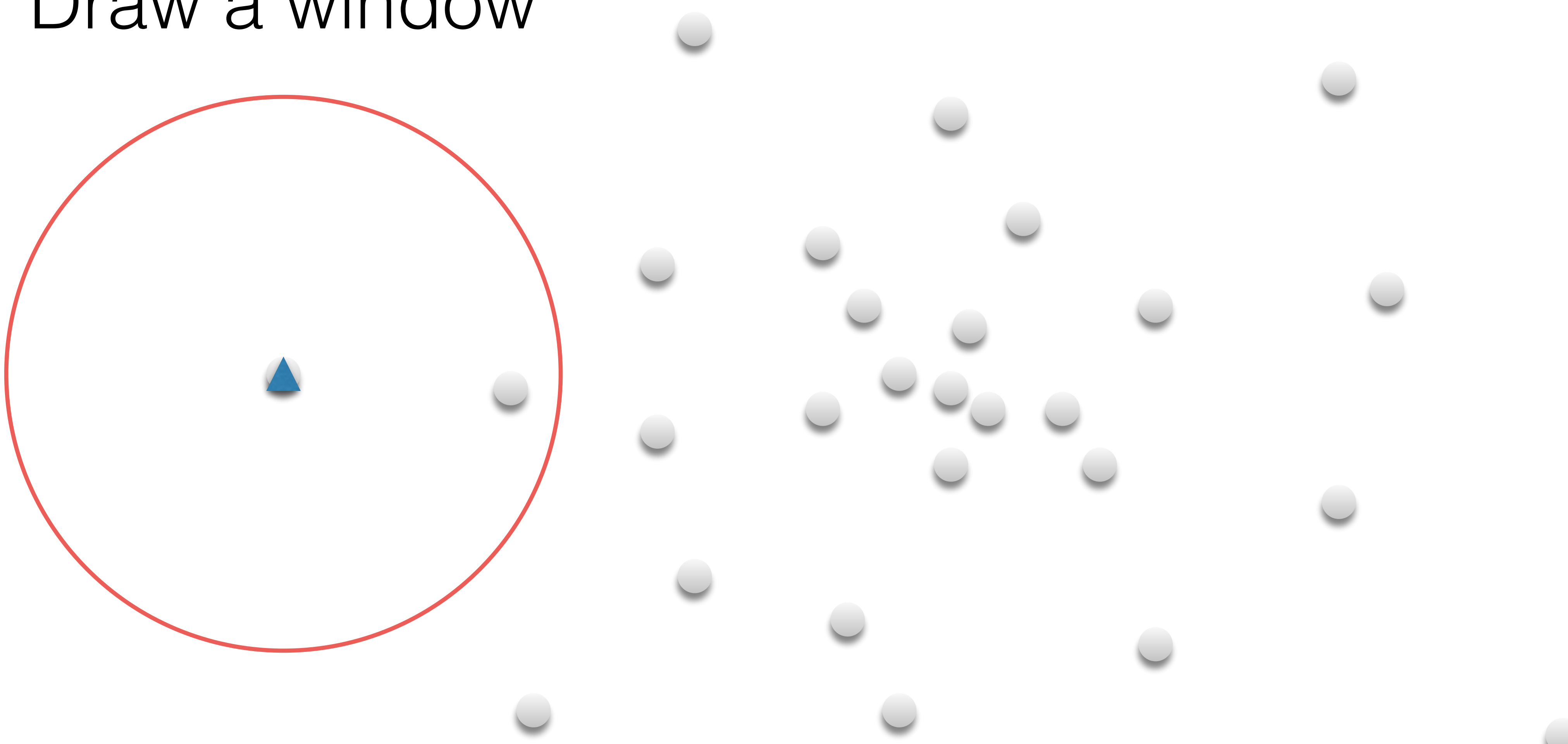
Pick a point

Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Draw a window

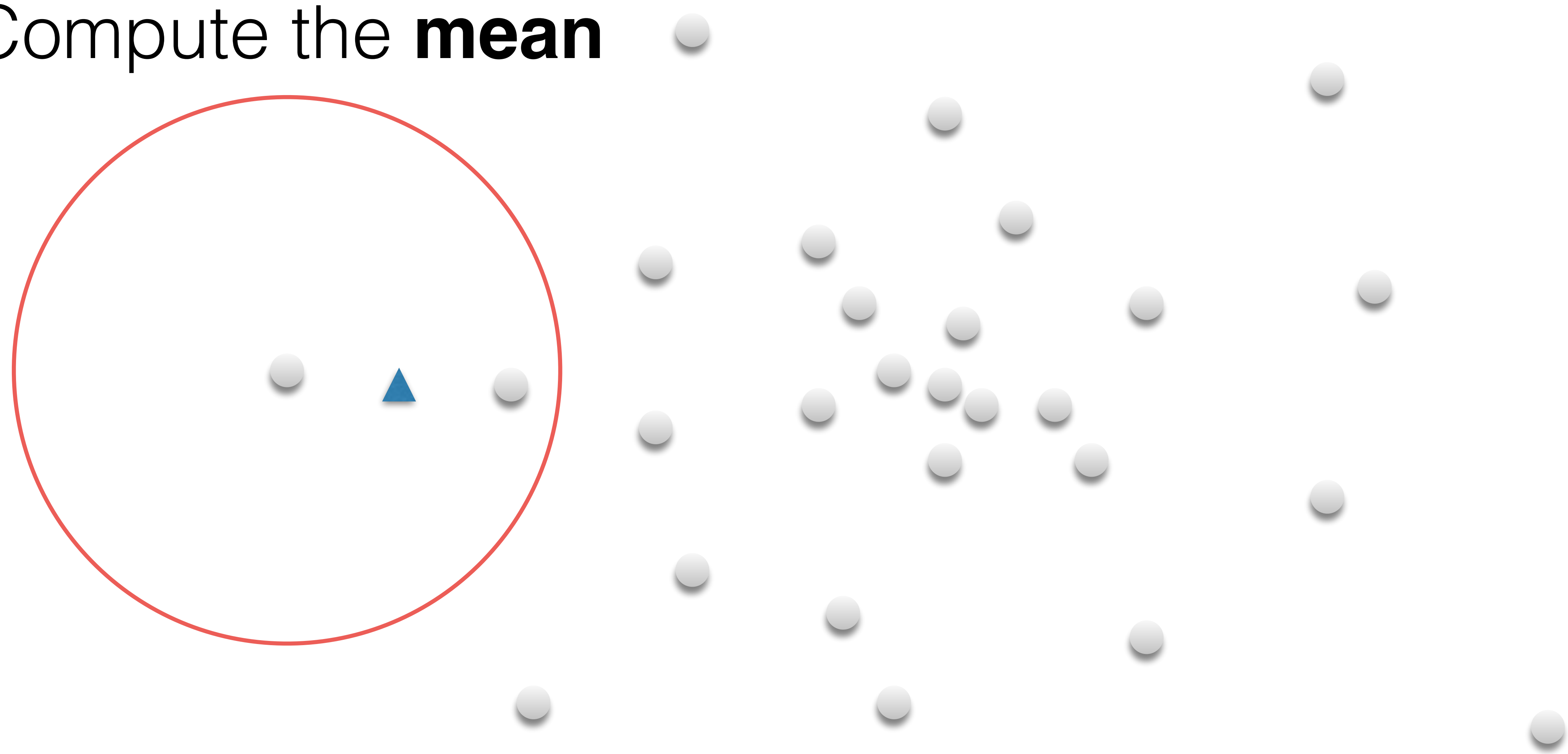


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Compute the **mean**

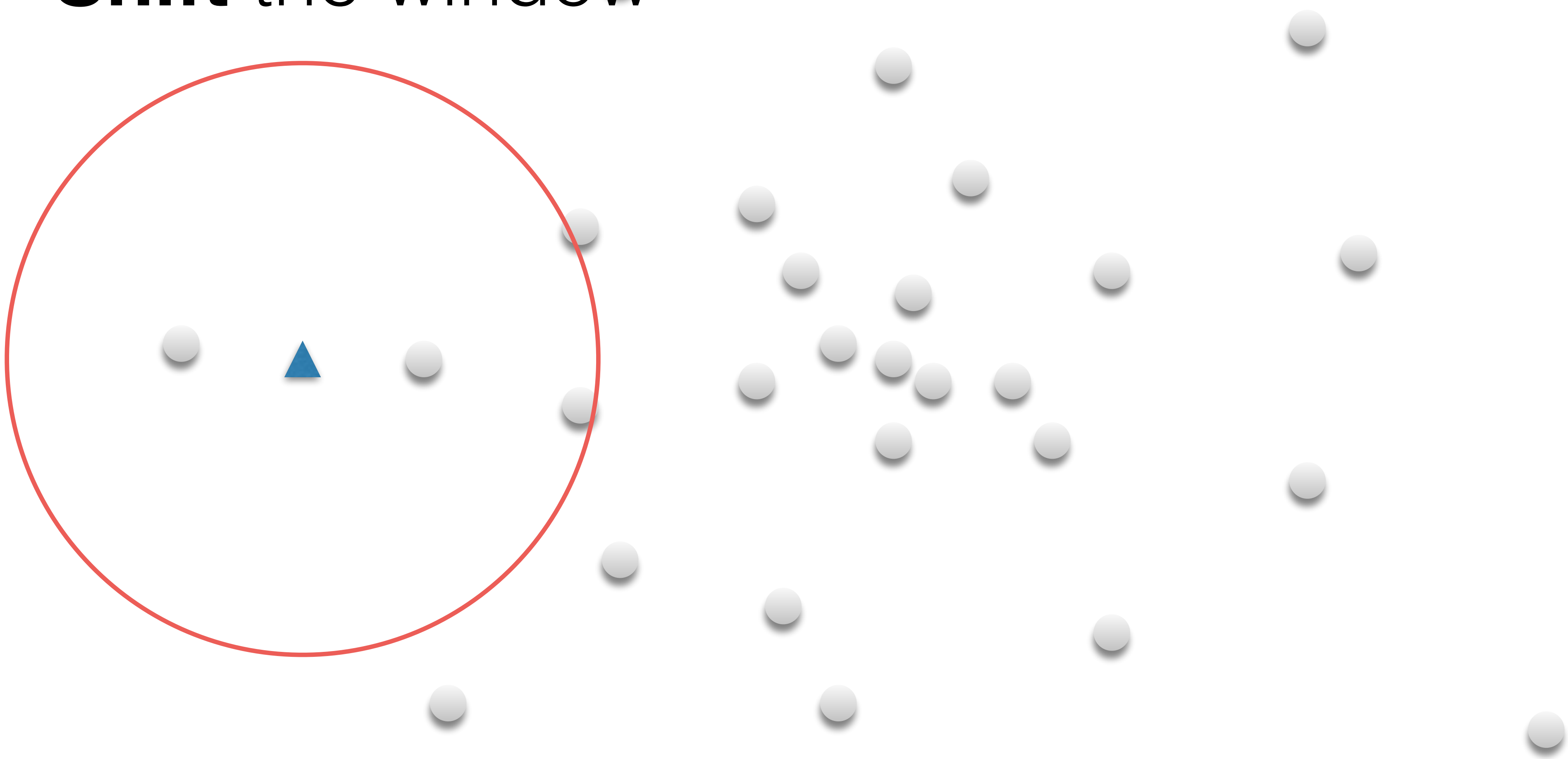


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window

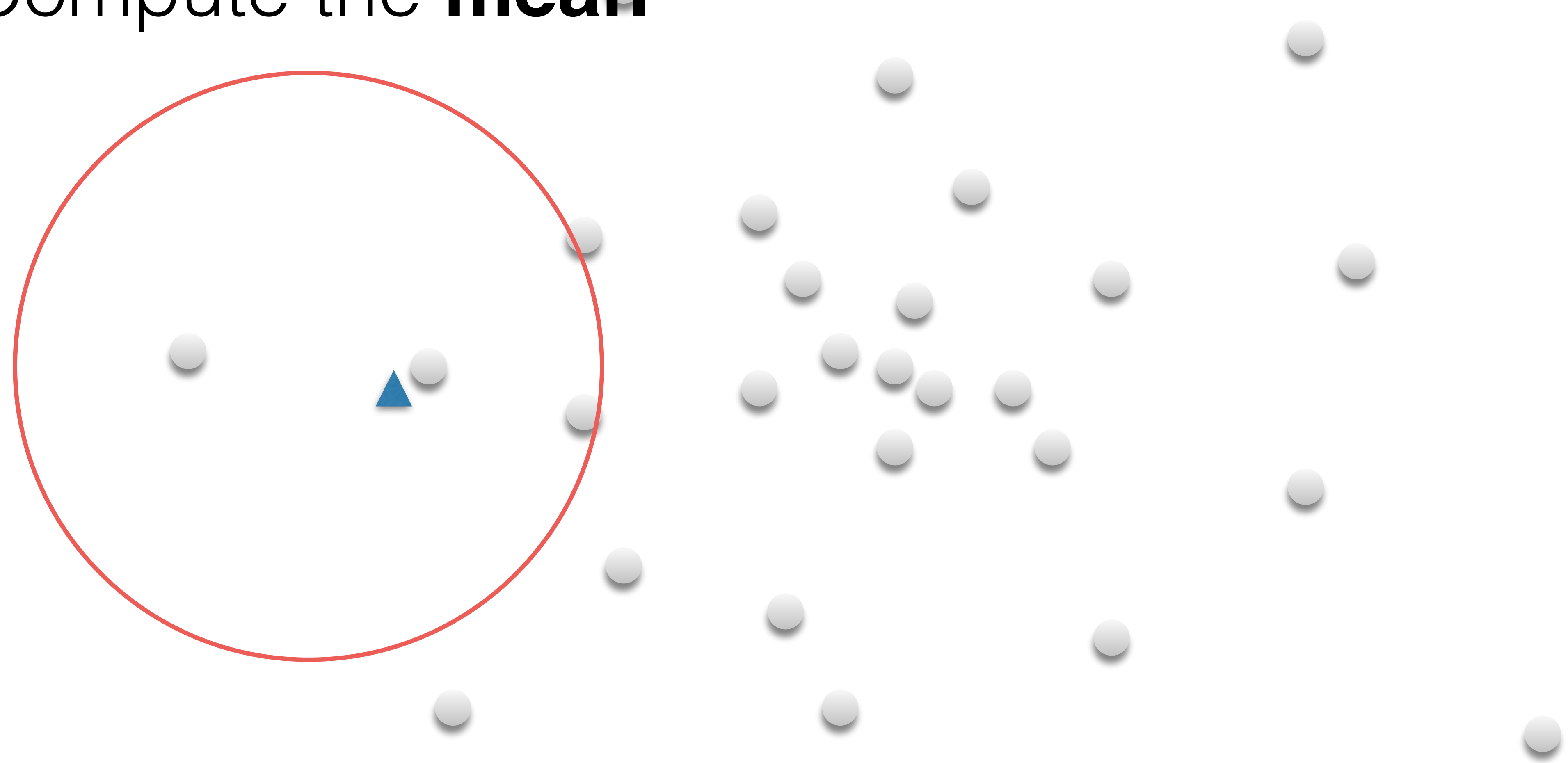


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

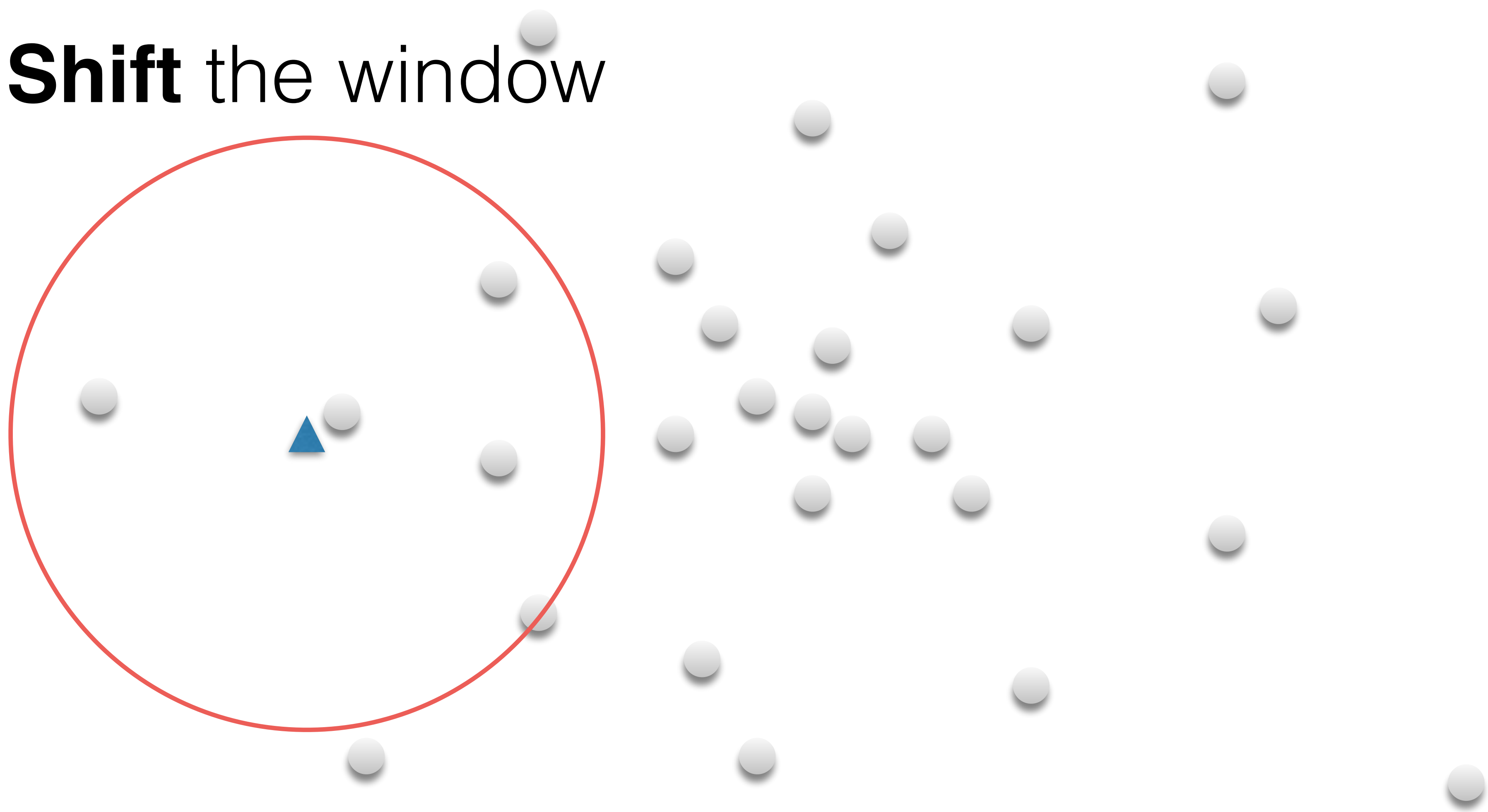
Compute the **mean**



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

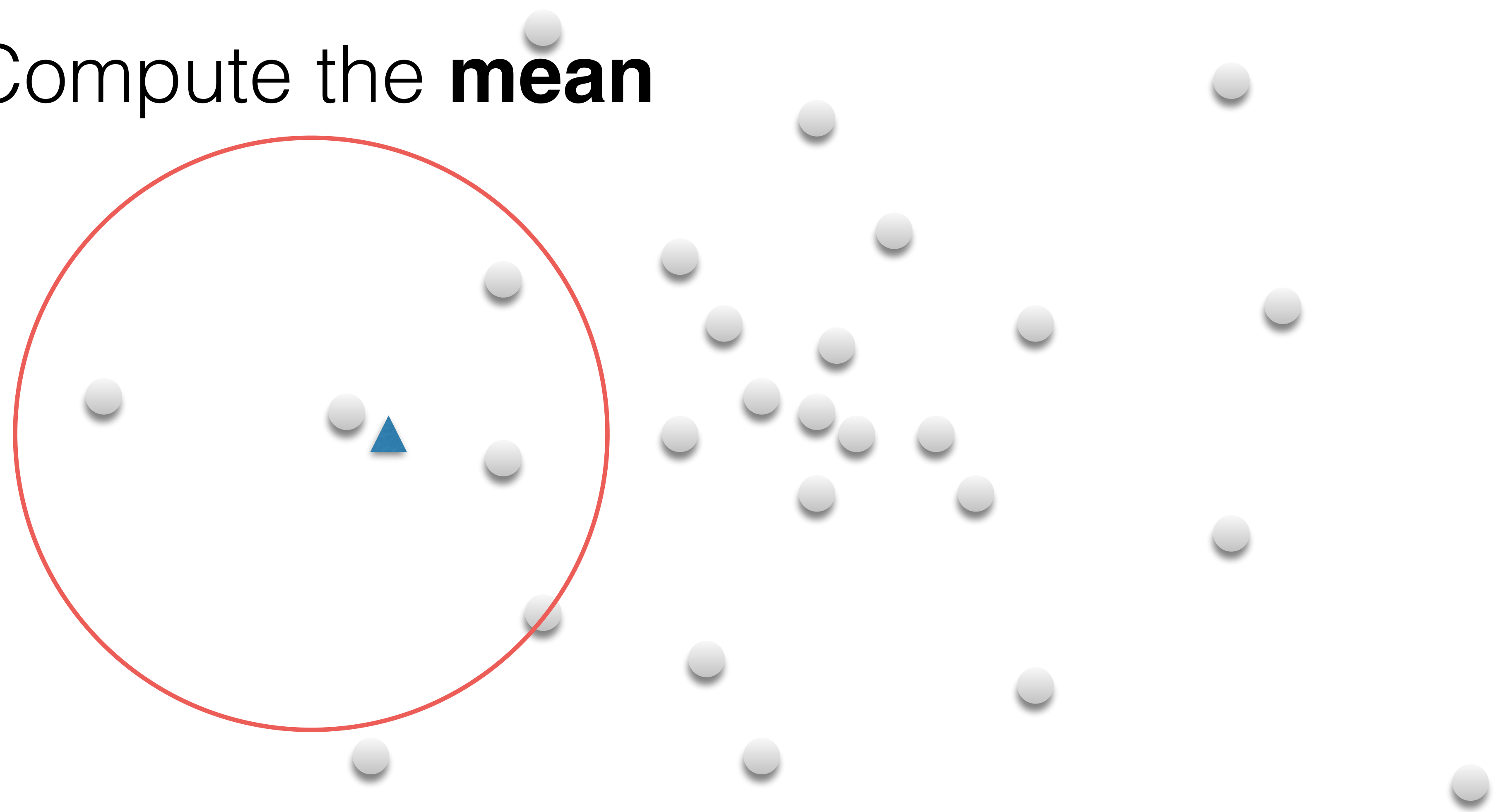


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

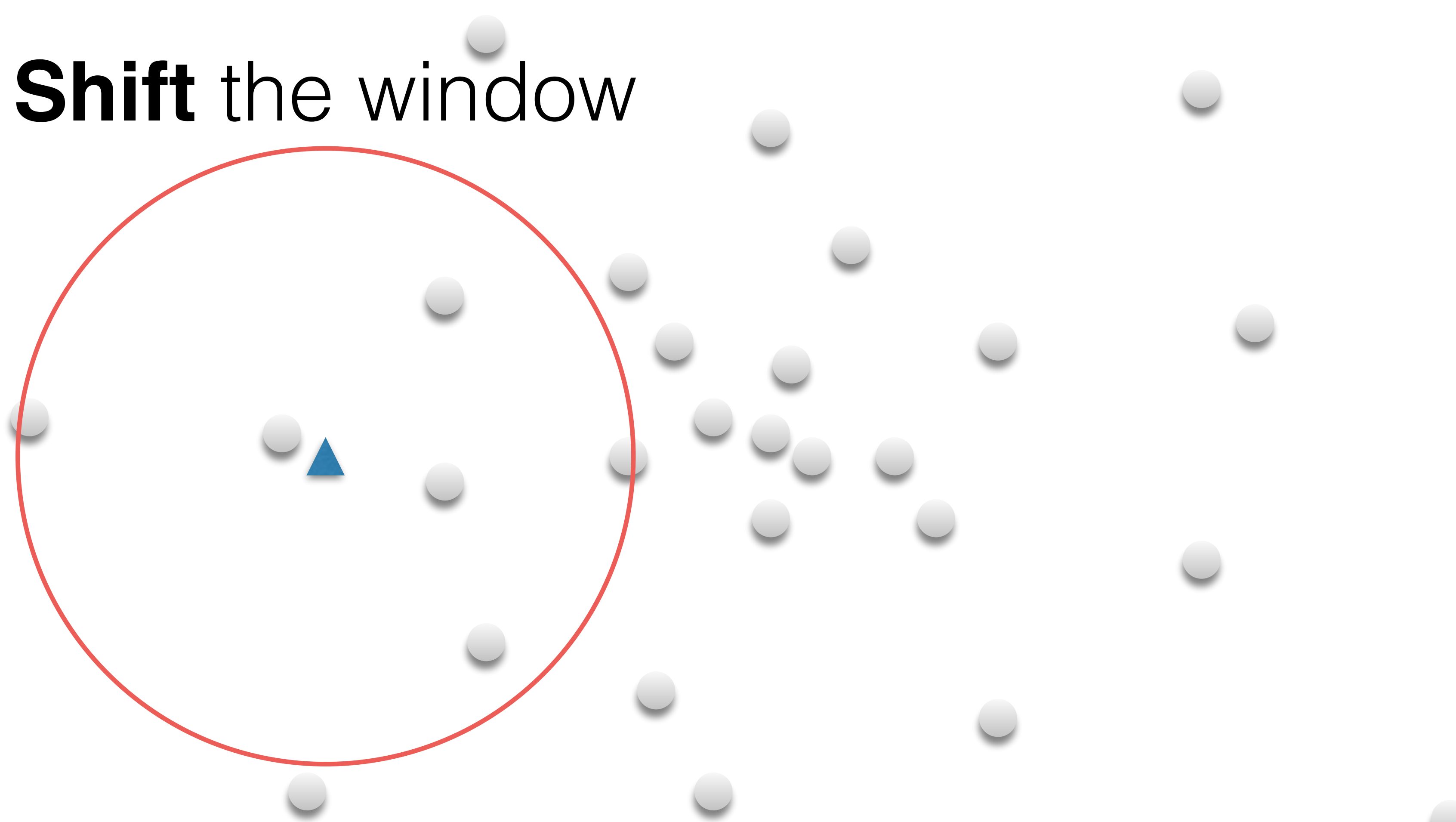
Compute the **mean**



Mean Shift Algorithm

A ‘mode seeking’ algorithm

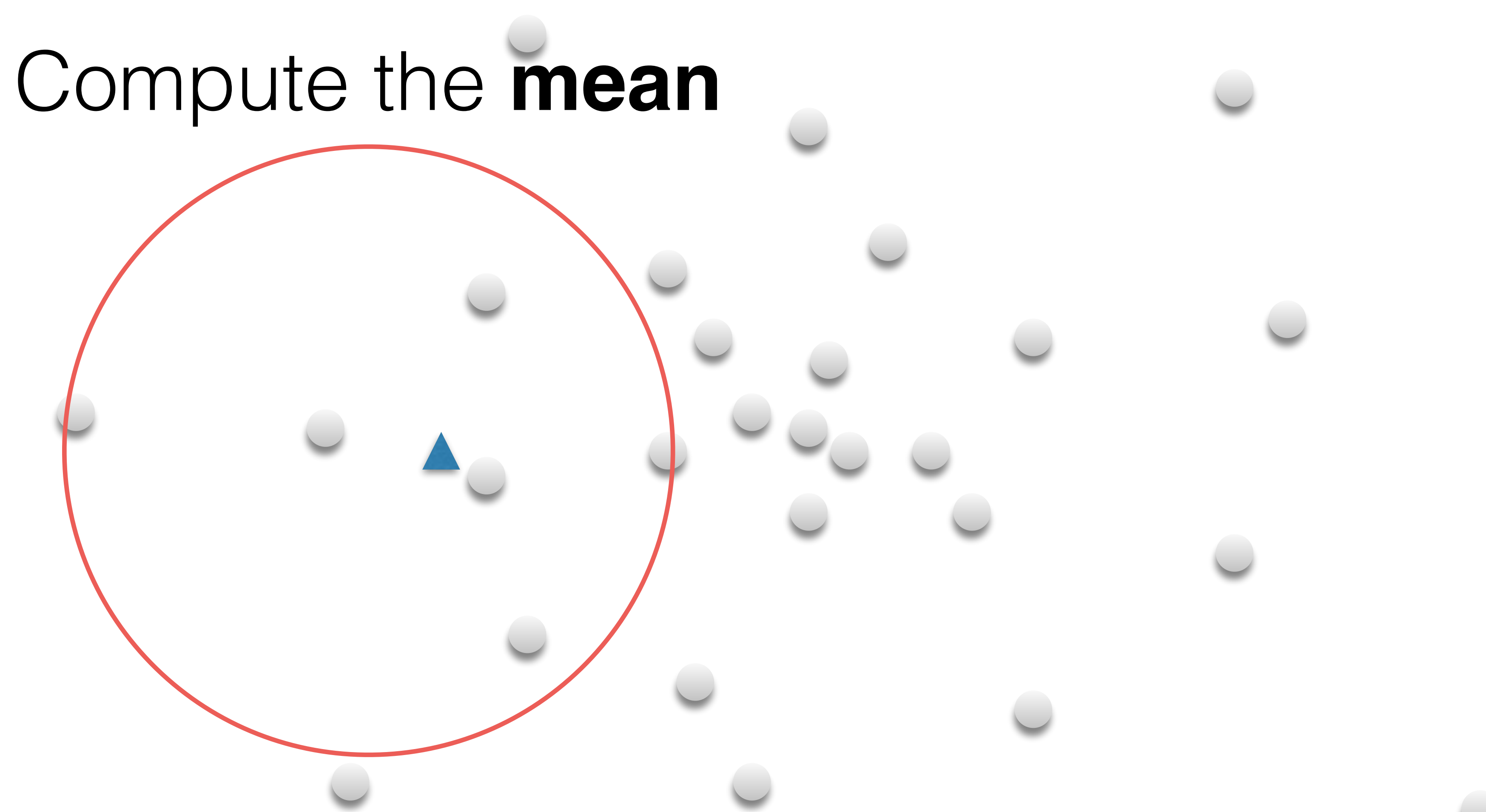
Fukunaga & Hostetler (1975)



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

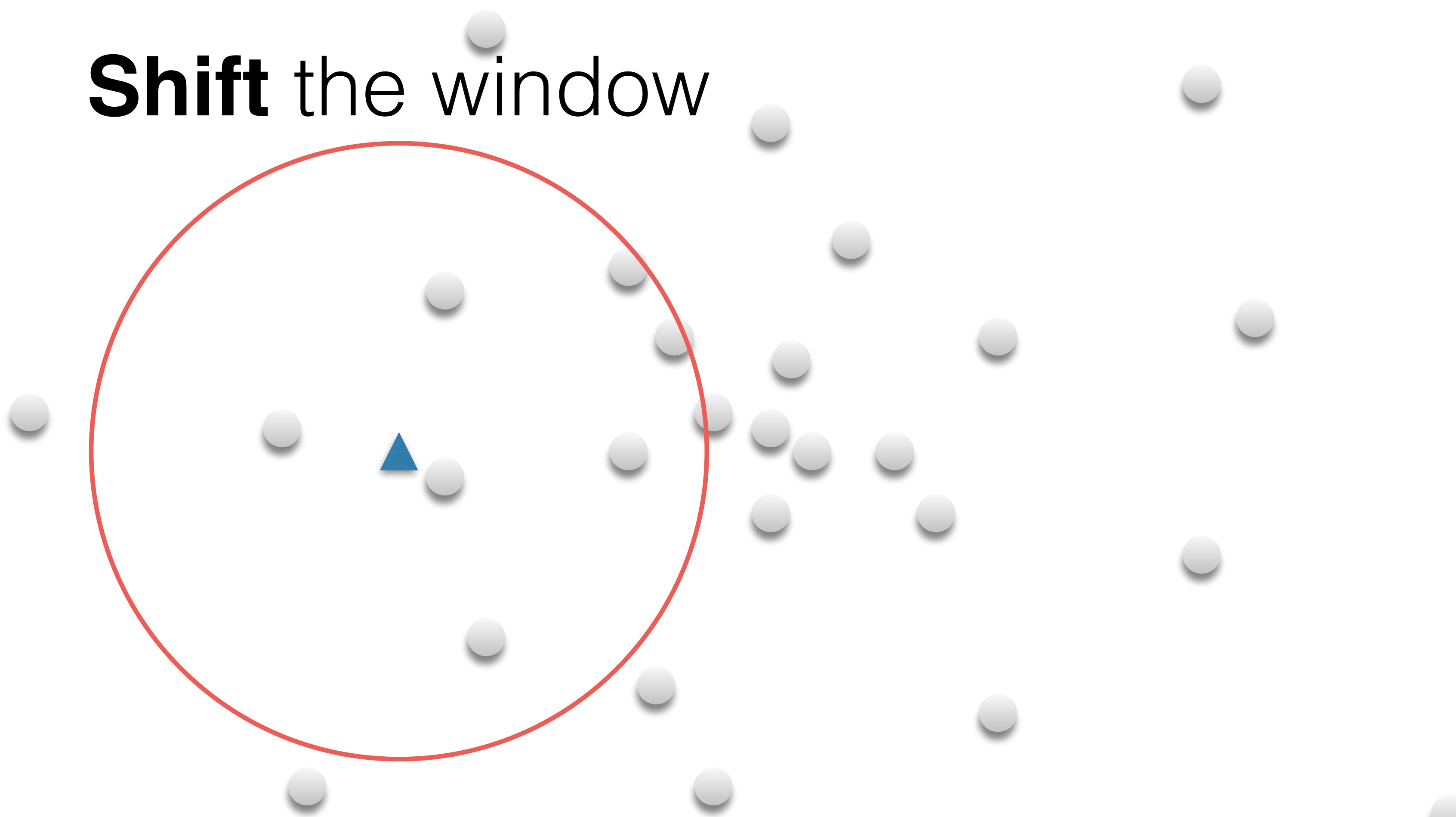


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

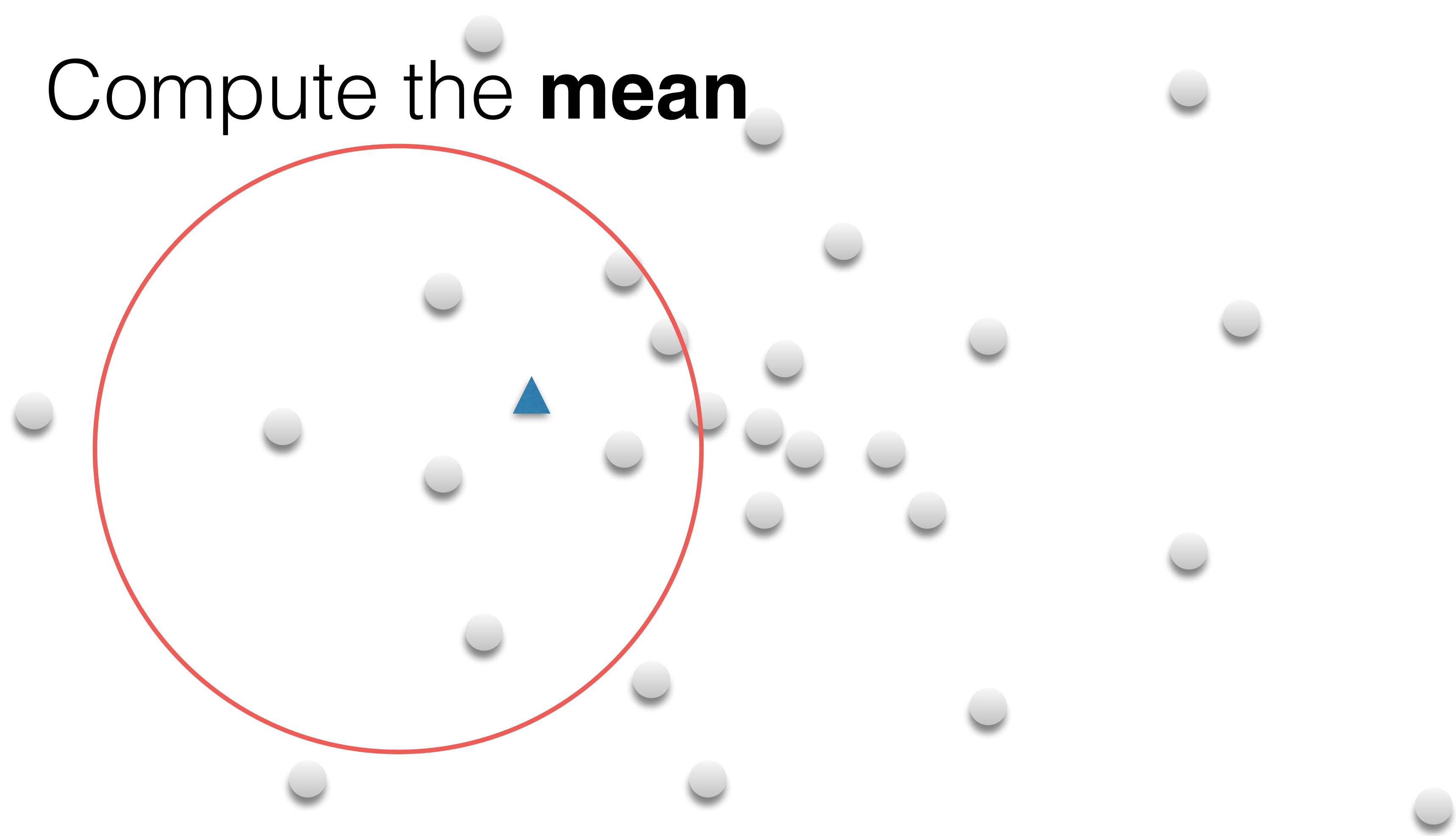
Shift the window



Mean Shift Algorithm

A 'mode seeking' algorithm

Fukunaga & Hostetler (1975)

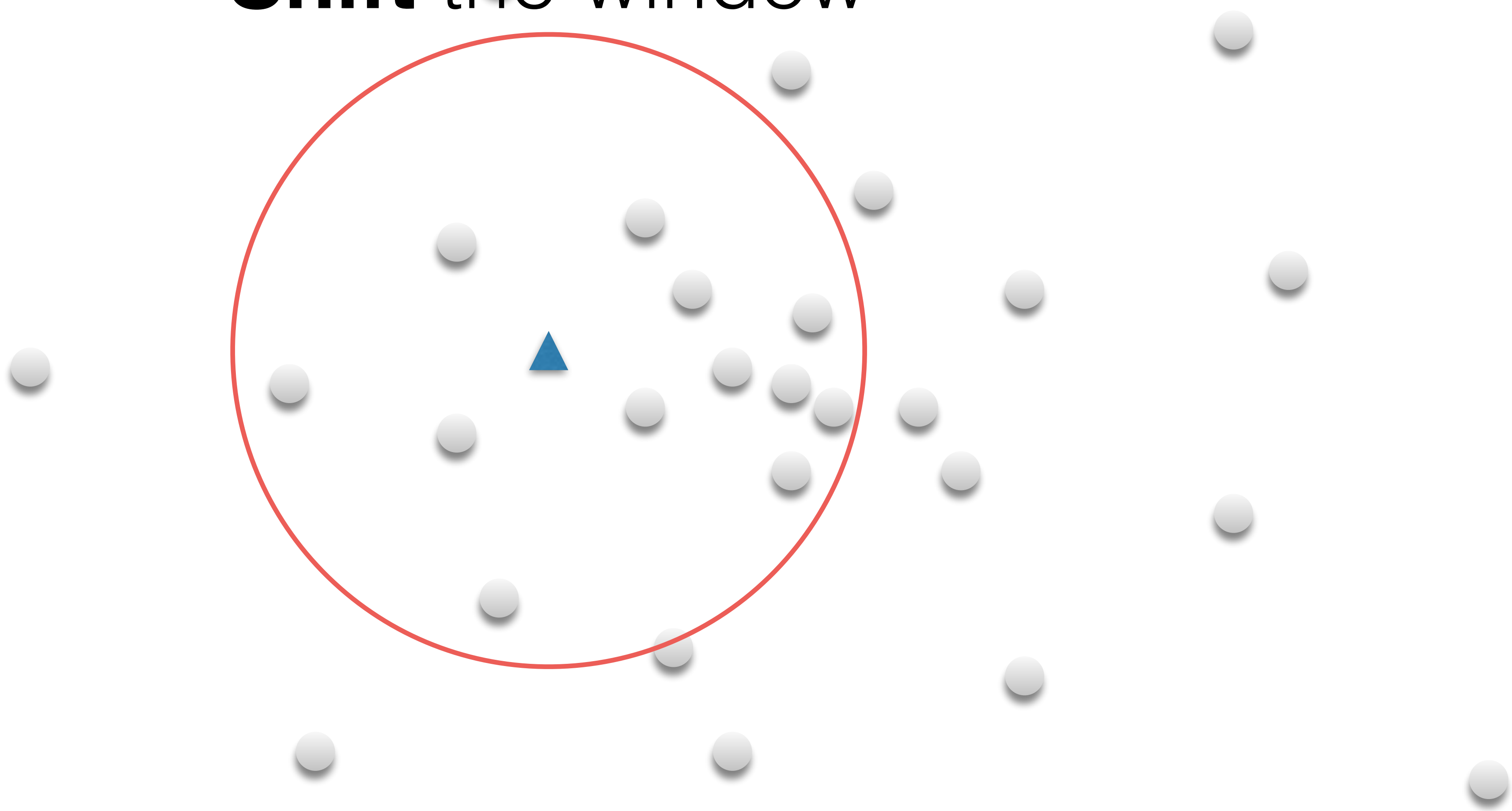


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window

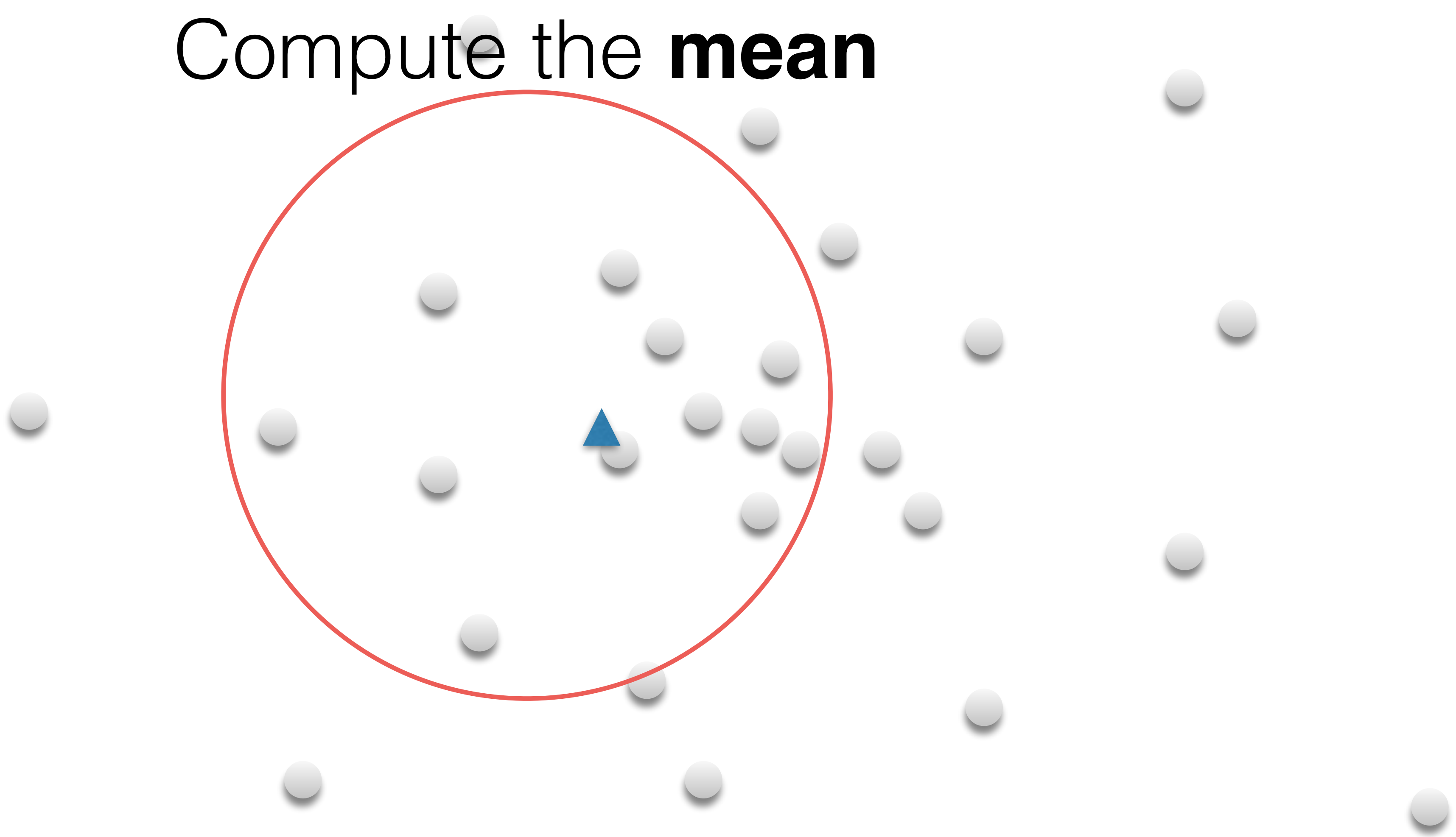


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

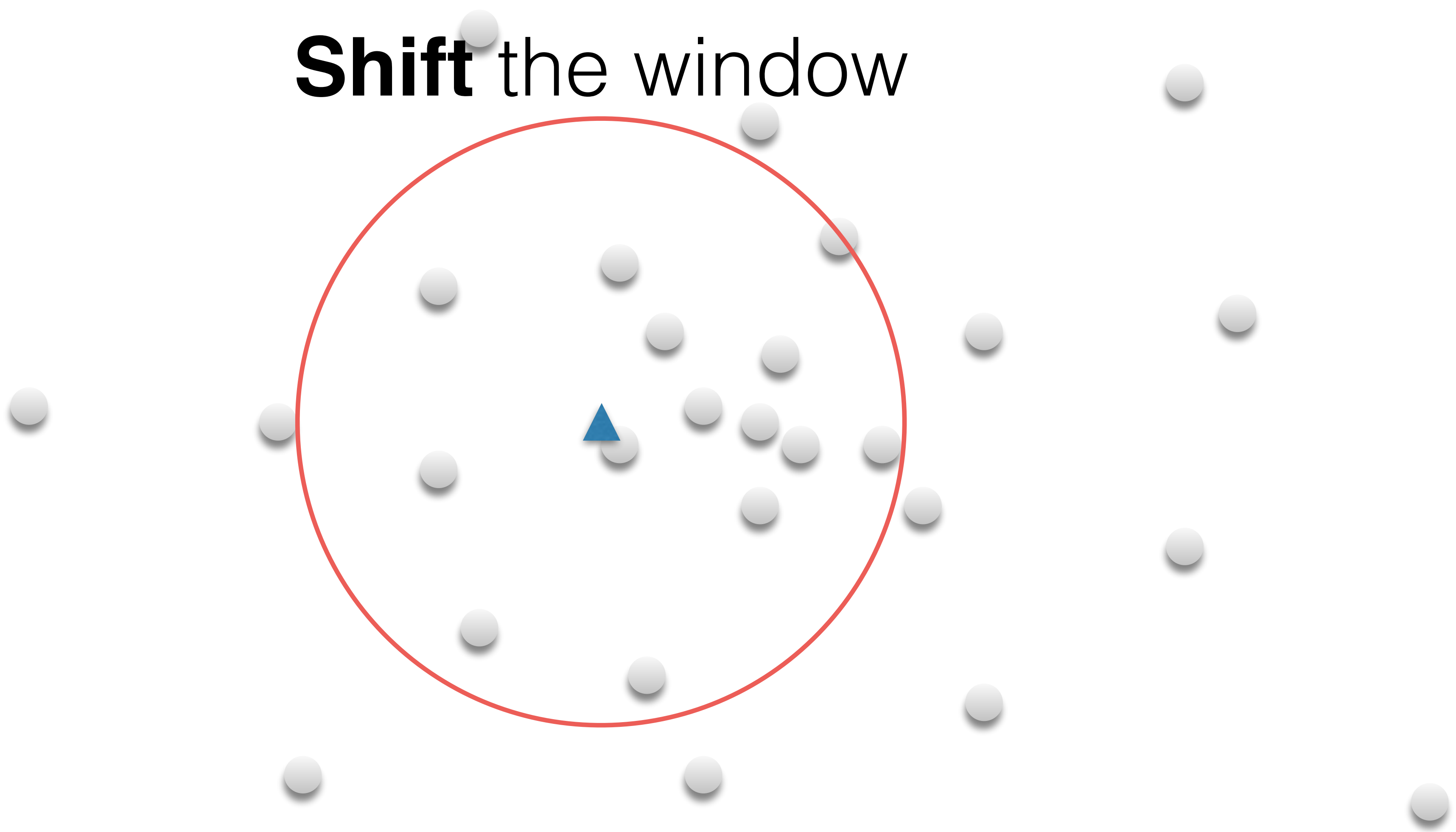
Compute the **mean**



Mean Shift Algorithm

A ‘mode seeking’ algorithm

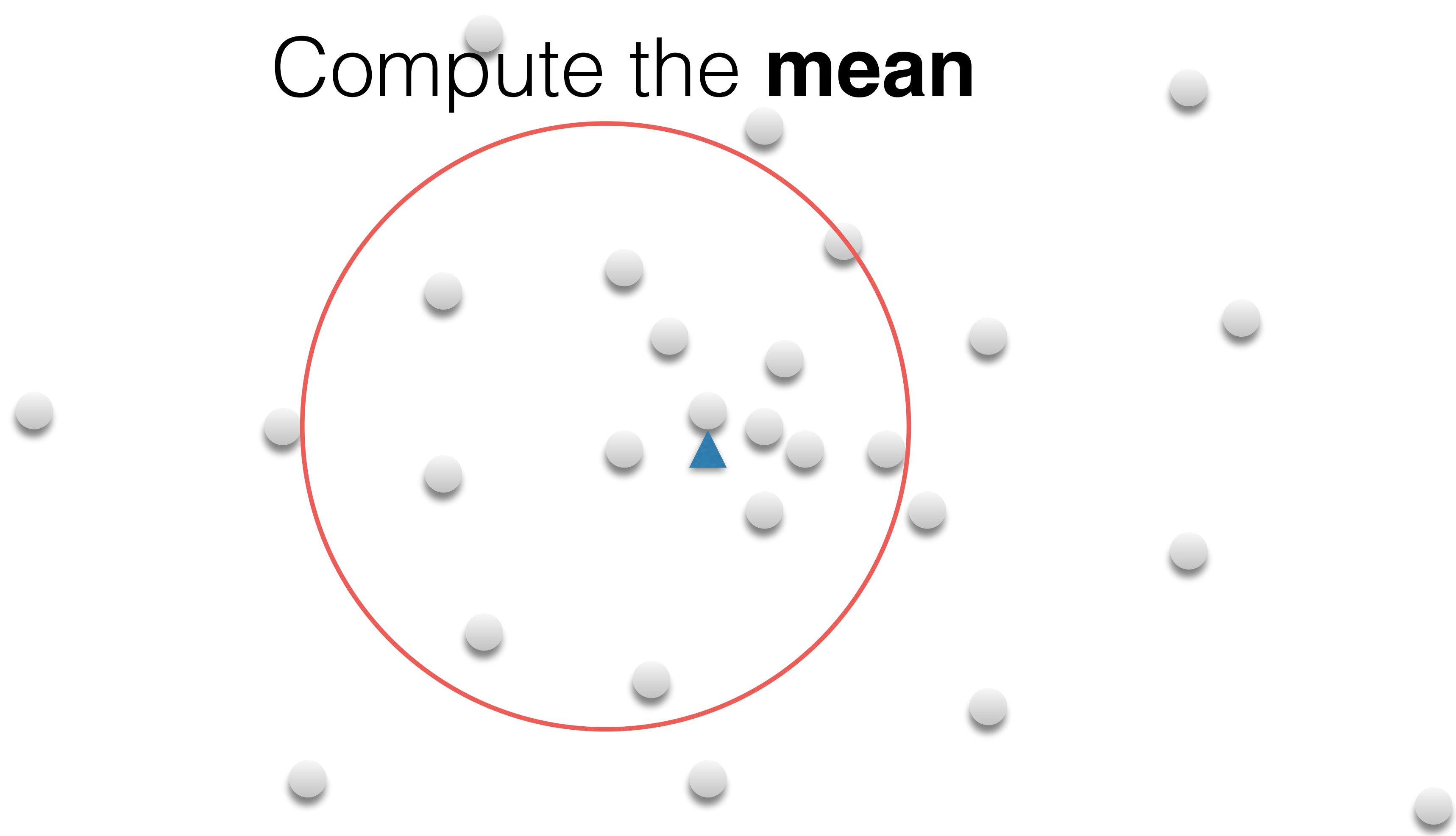
Fukunaga & Hostetler (1975)



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

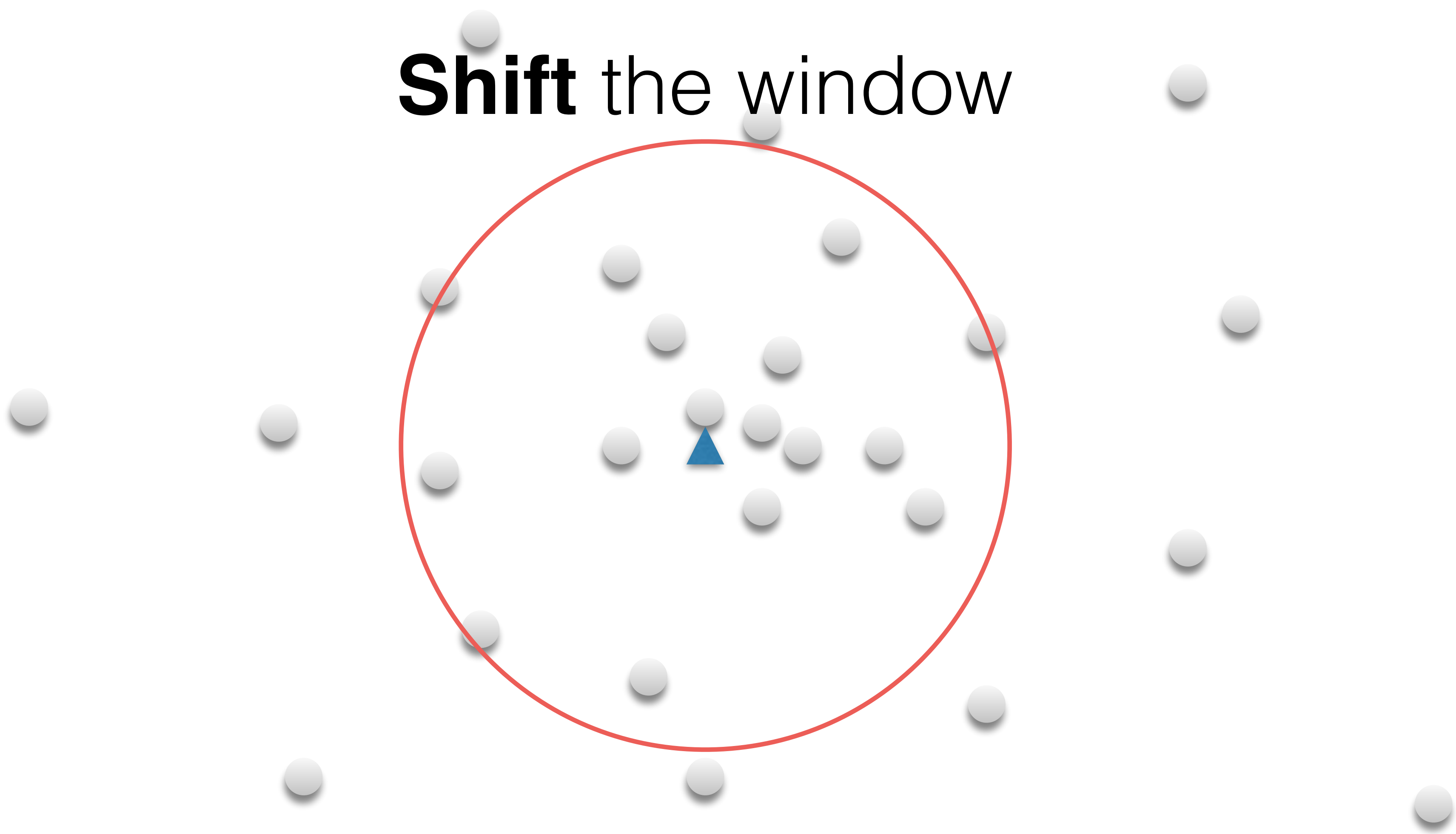


Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

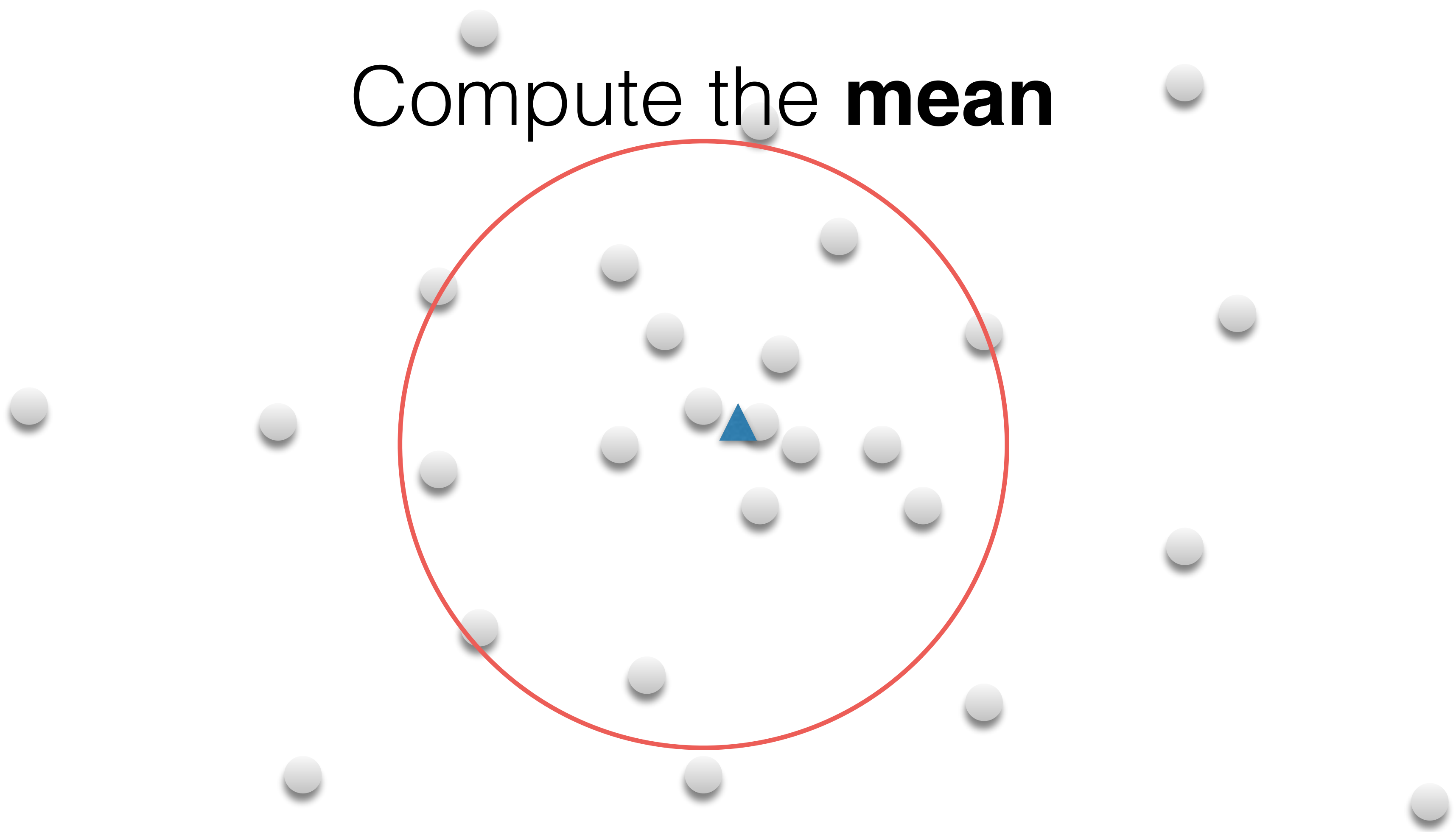
Shift the window



Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

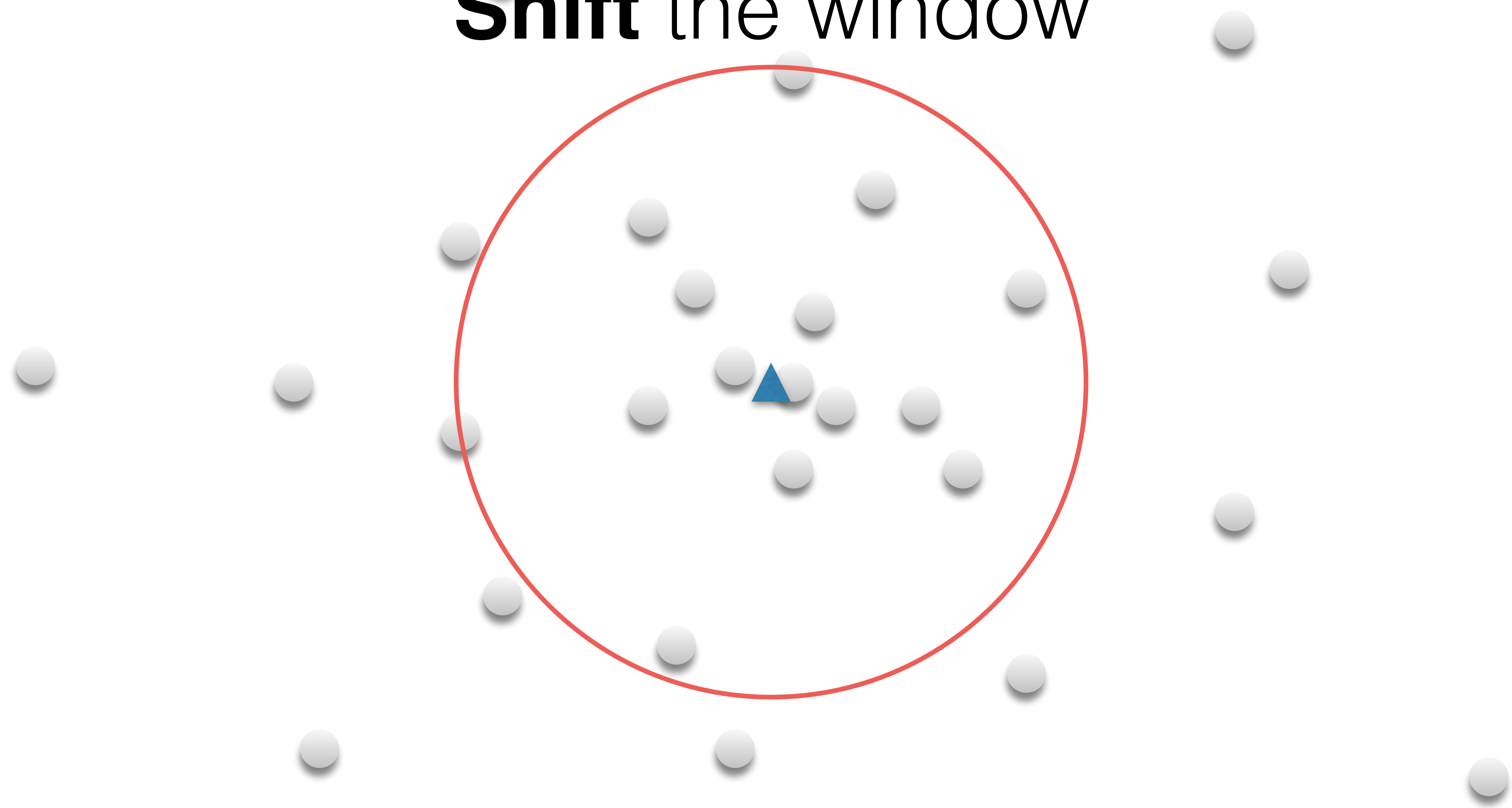


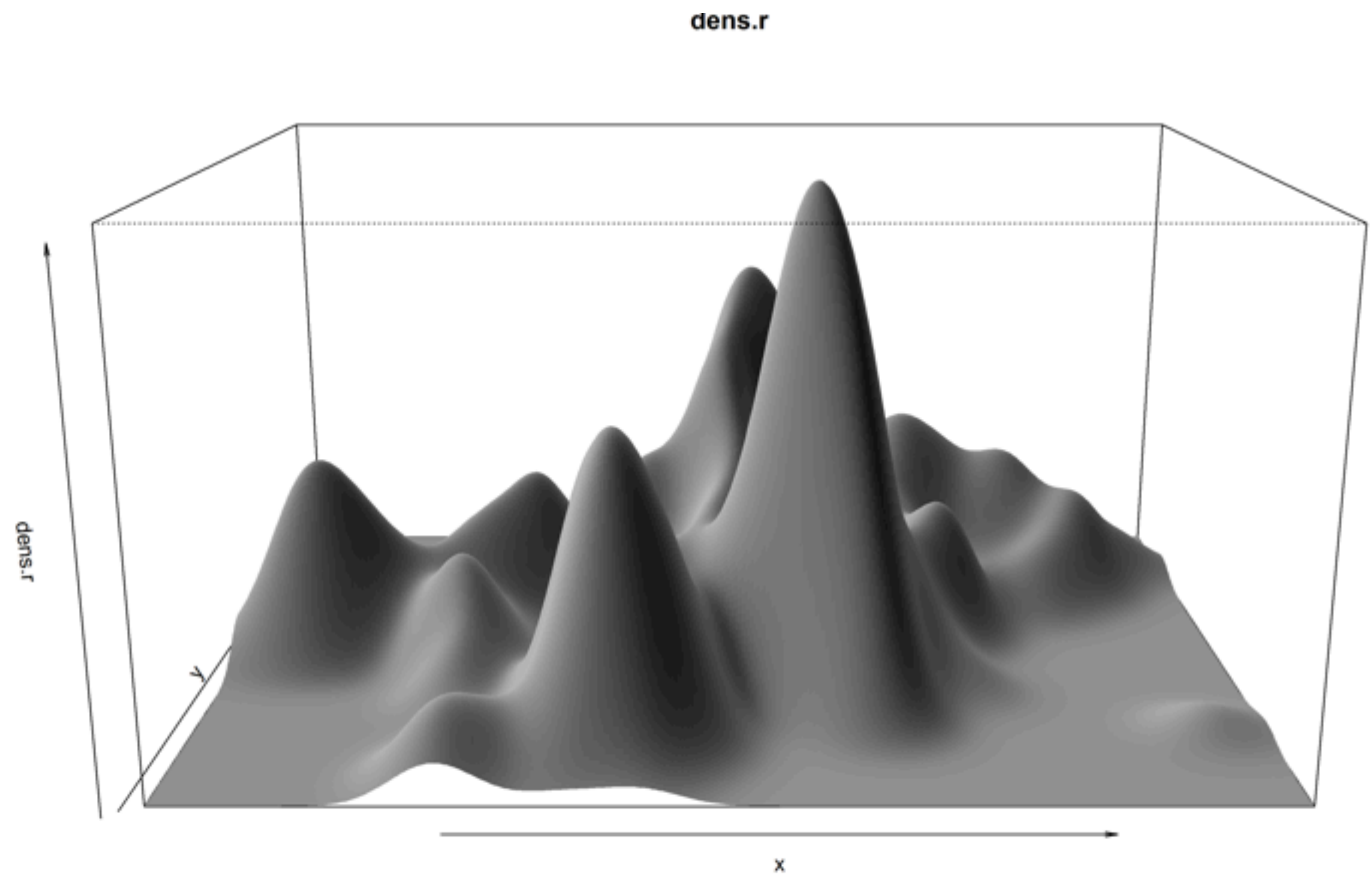
Mean Shift Algorithm

A ‘mode seeking’ algorithm

Fukunaga & Hostetler (1975)

Shift the window





Kernel Density Estimation

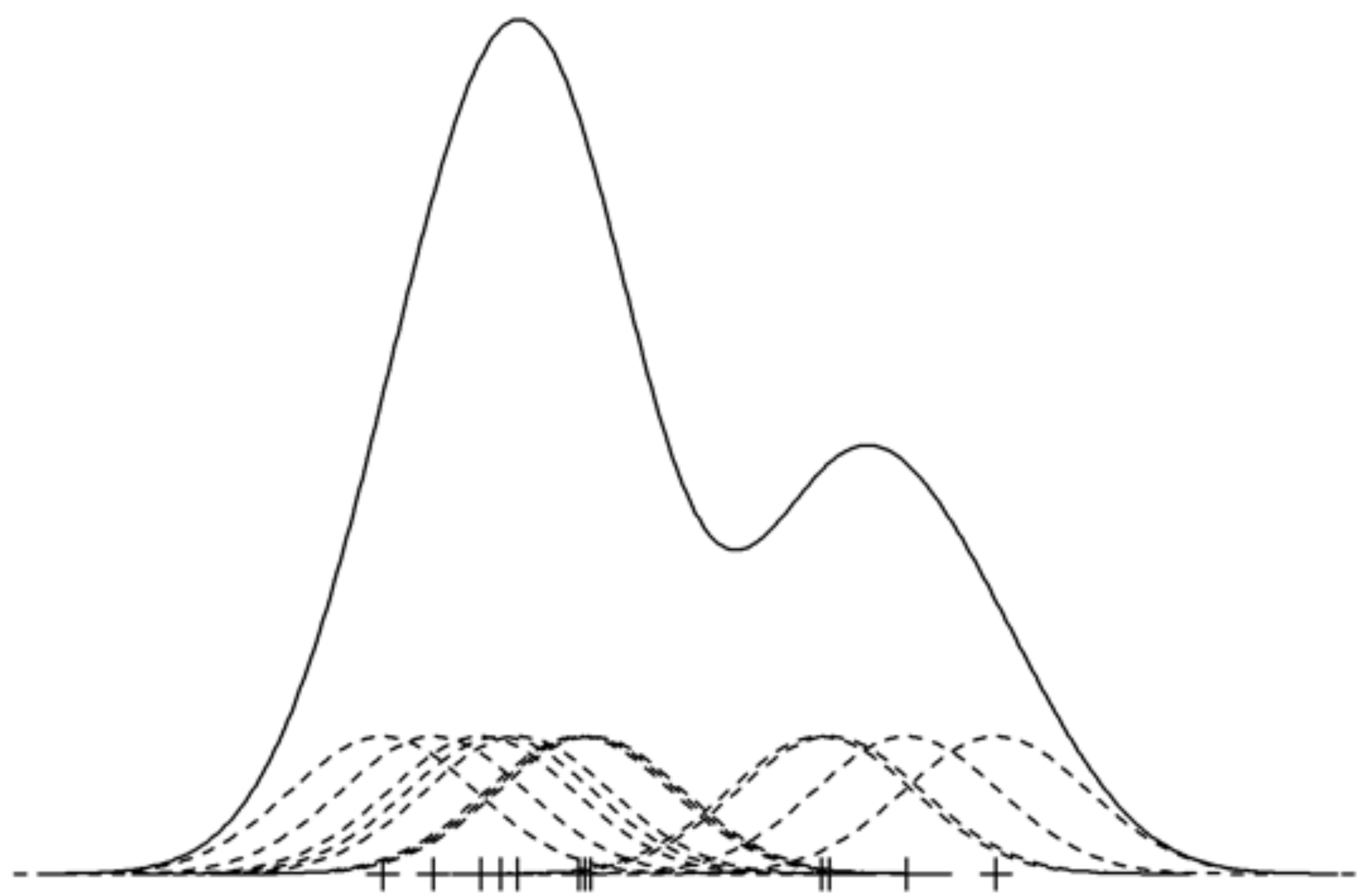
16-385 Computer Vision (Kris Kitani)

Carnegie Mellon University

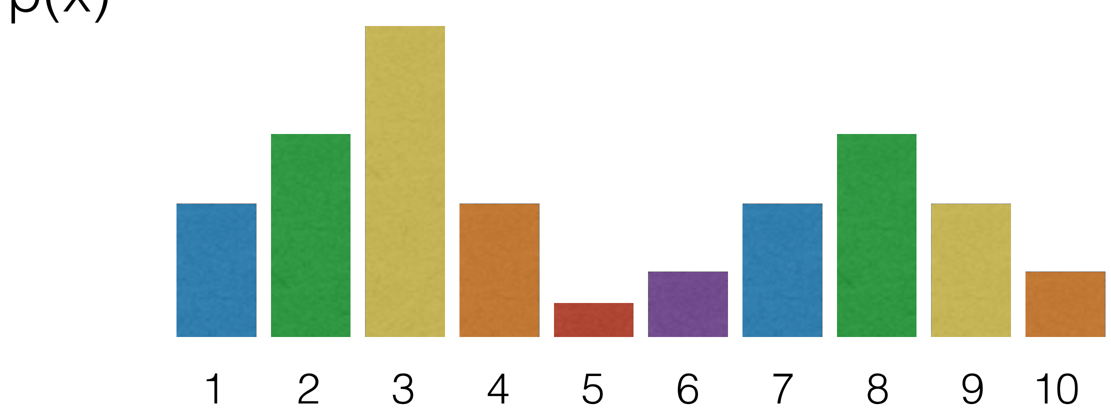
To understand the mean shift algorithm …

Kernel Density Estimation

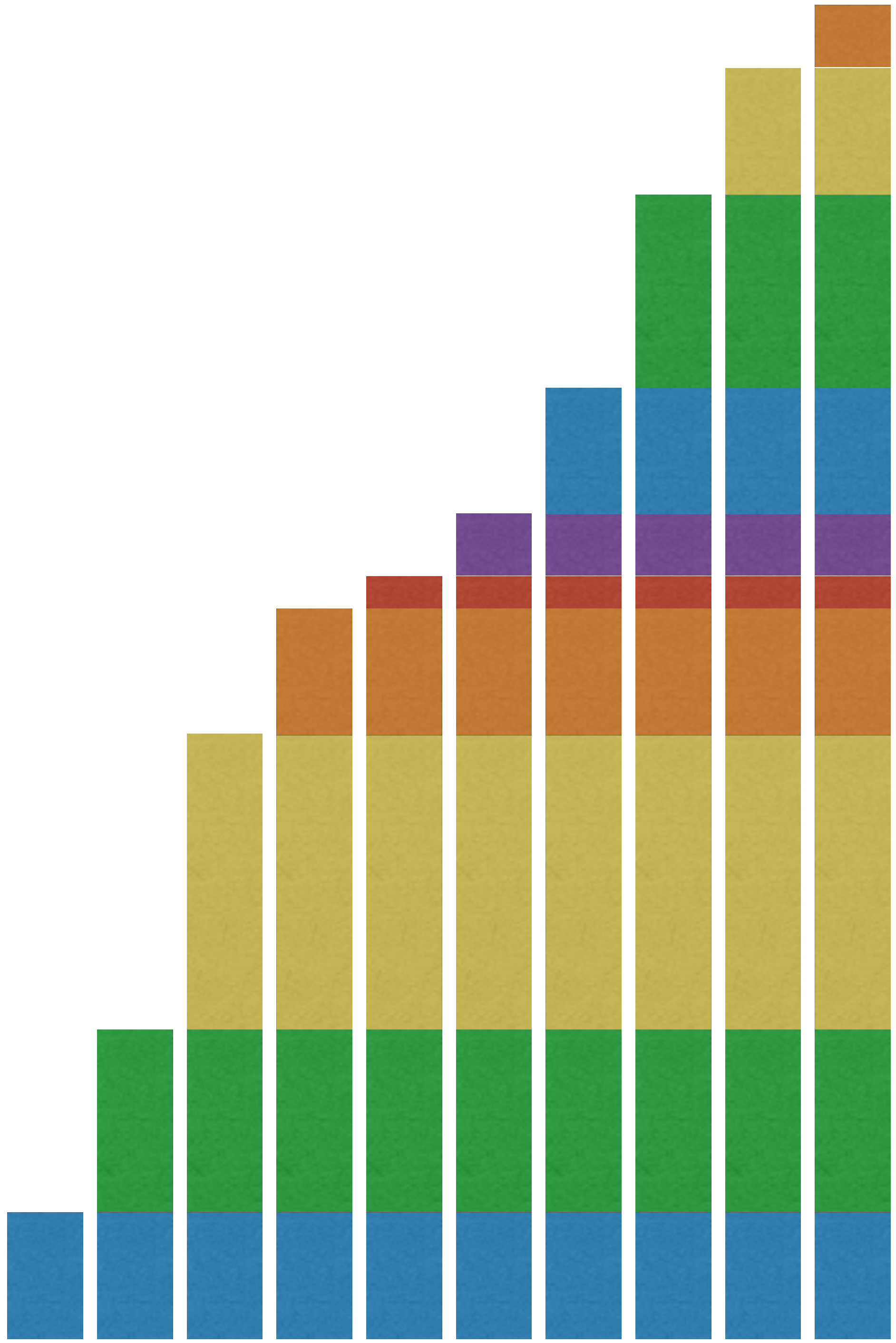
Approximate the underlying PDF from samples



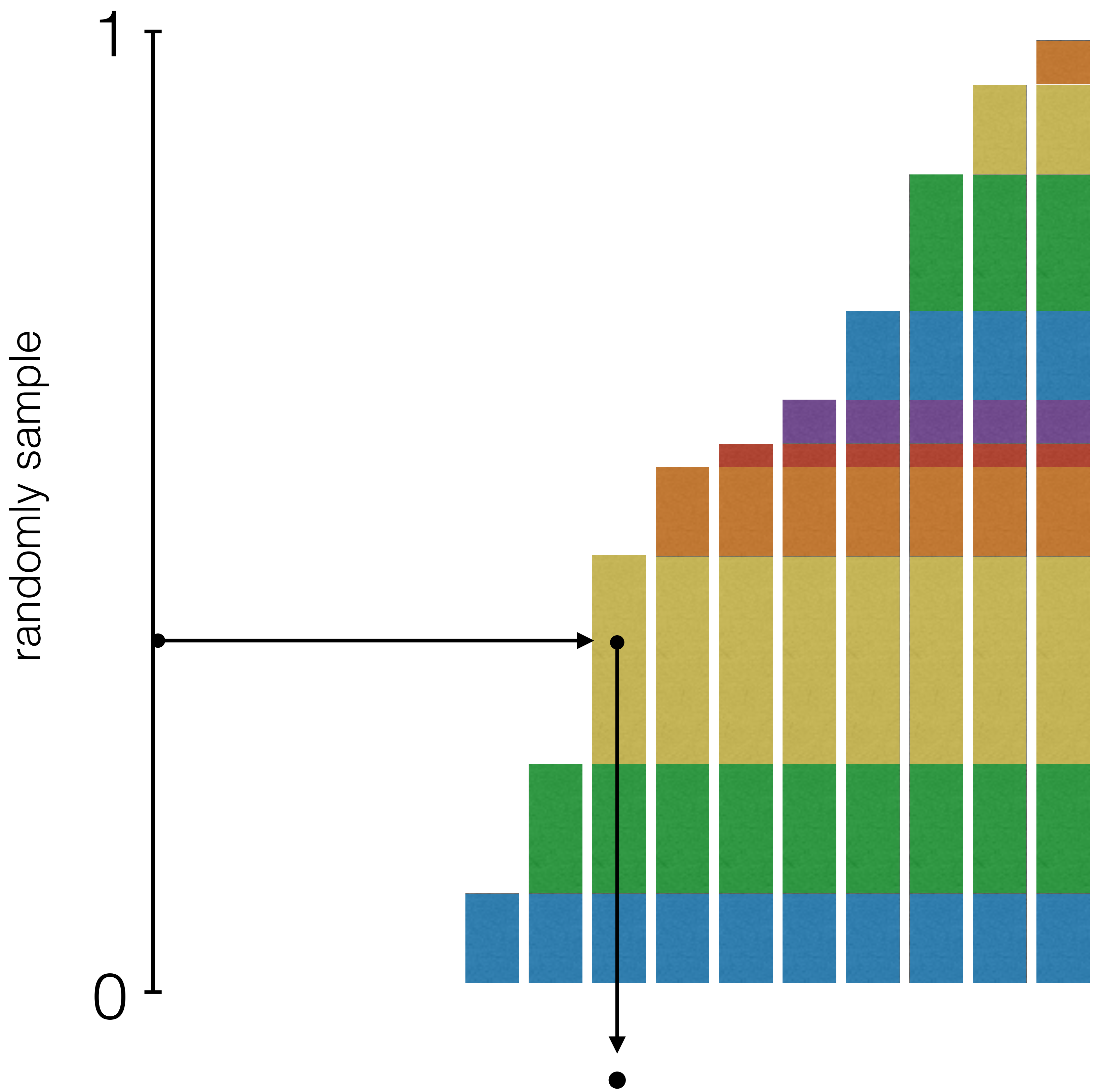
Put ‘bump’ on every sample to approximate the PDF

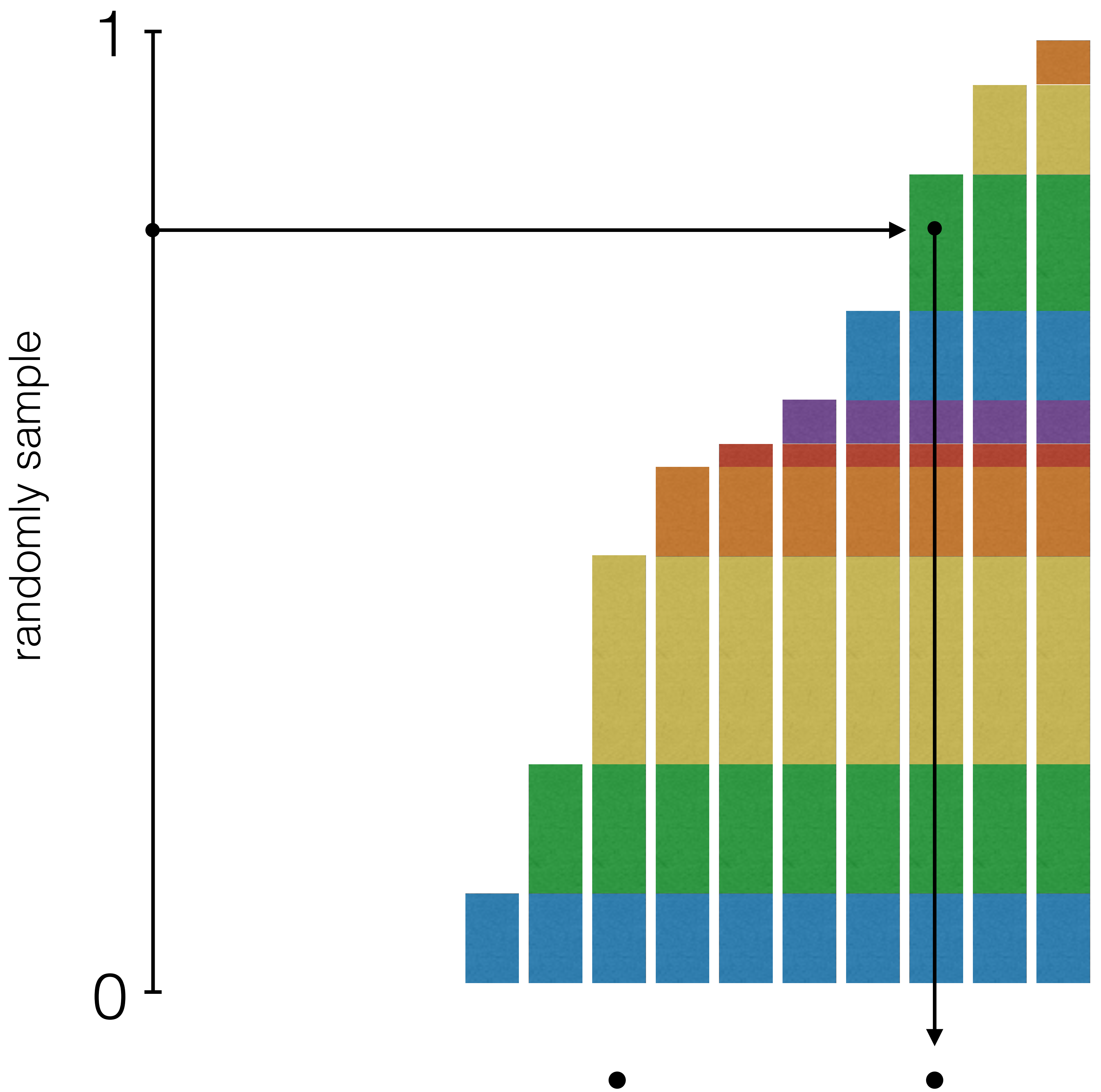


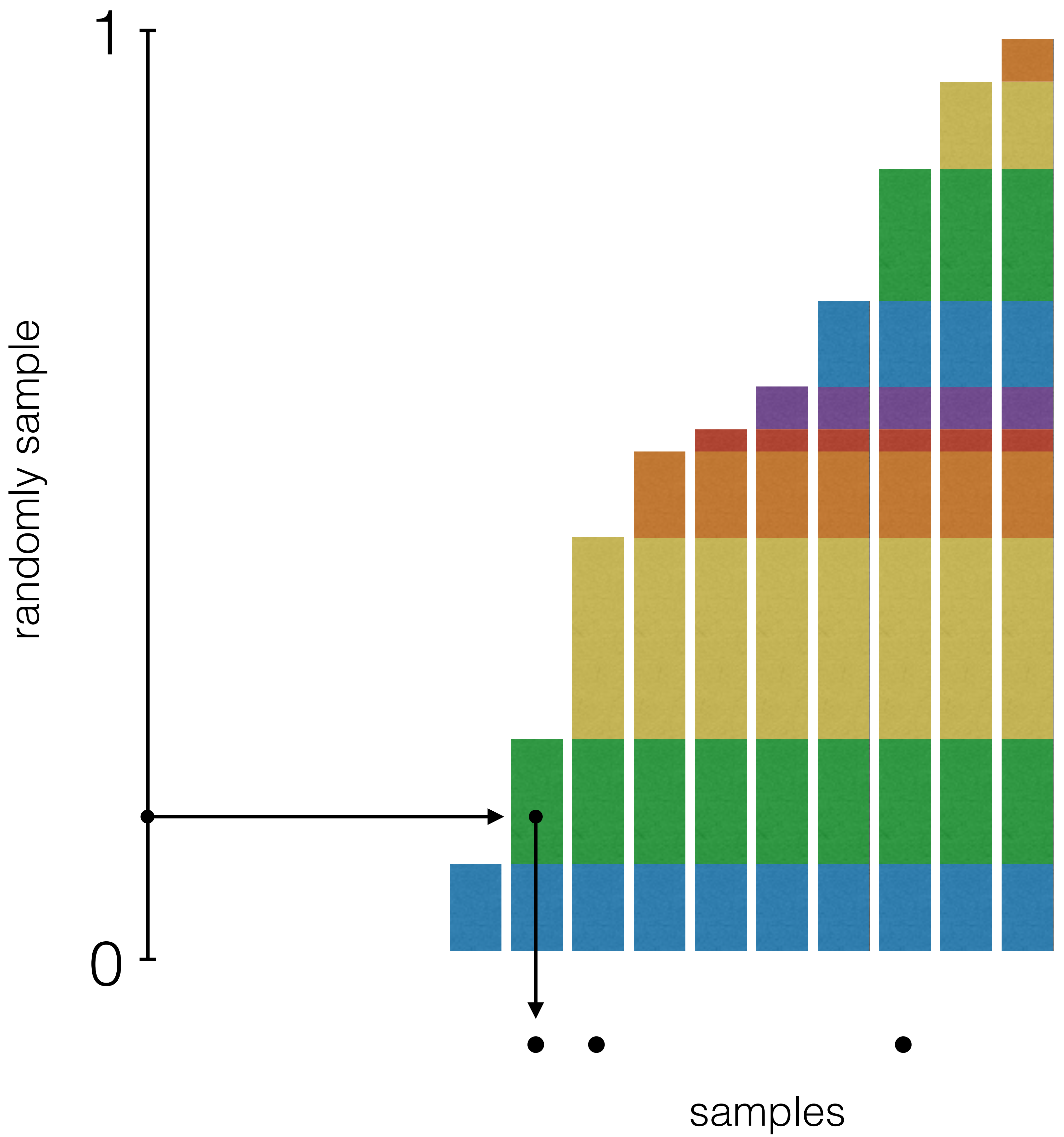
probability density function



cumulative density function


1
randomly sample
0


1
0
randomly sample

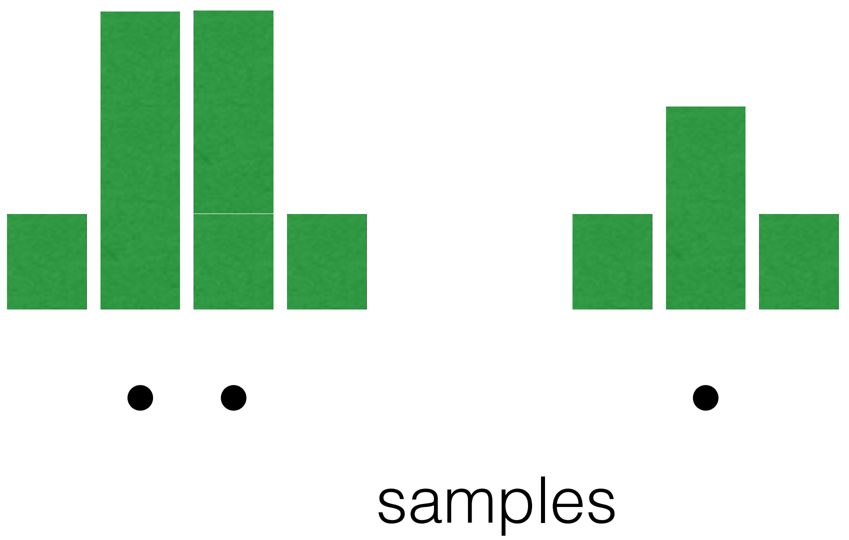

1
0
randomly sample
samples

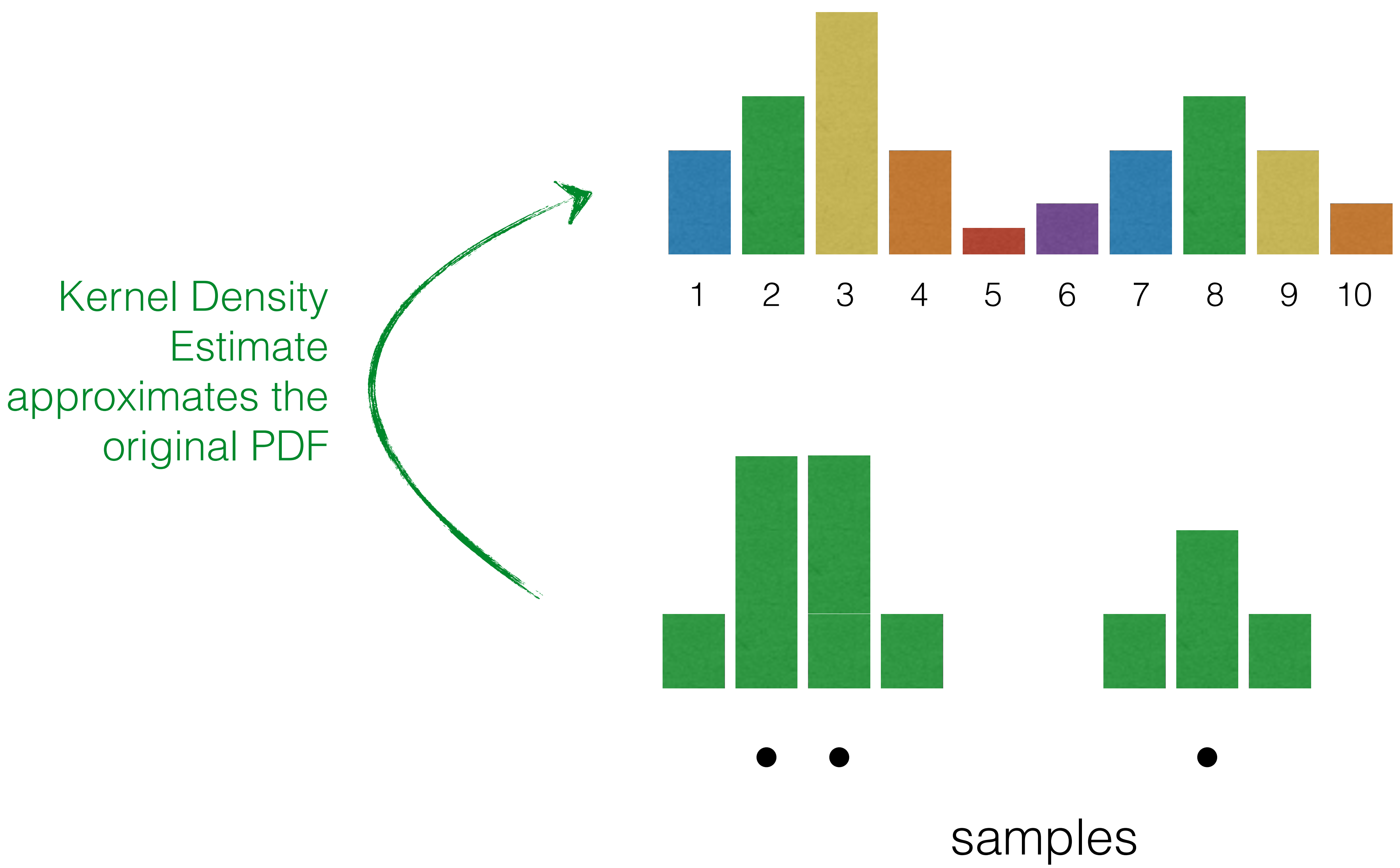
place Gaussian bumps on the samples…

samples

samples

samples

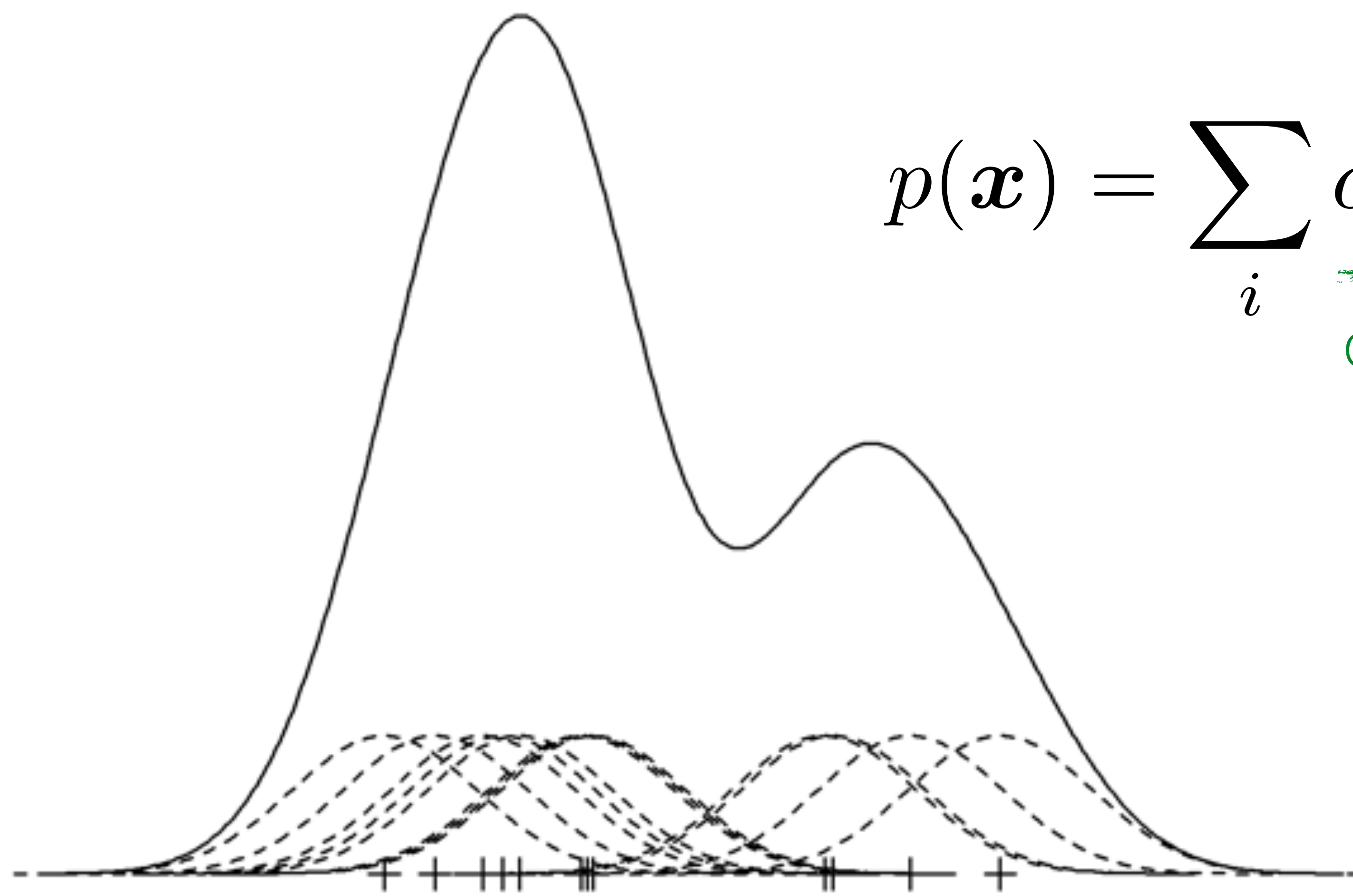

samples



Kernel Density
Estimate
approximates the
original PDF
1
2
3
4
5
6
7
8
9
10
samples

Kernel Density Estimation

Approximate the underlying PDF from samples from it



$$p(\boldsymbol{x}) = \sum_i \underbrace{c_i e^{-\frac{(\boldsymbol{x}-\boldsymbol{x}_i)^2}{2\sigma^2}}}_{\text{Gaussian ‘bump’ aka ‘kernel’}}$$

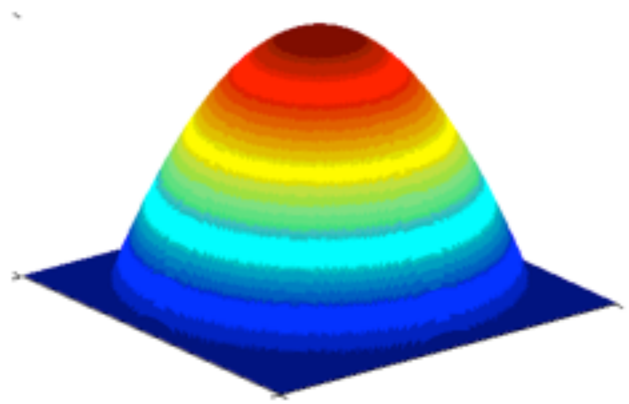
Put ‘bump’ on every sample to approximate the PDF

Kernel Function

$$K(\boldsymbol{x}, \boldsymbol{x}')$$

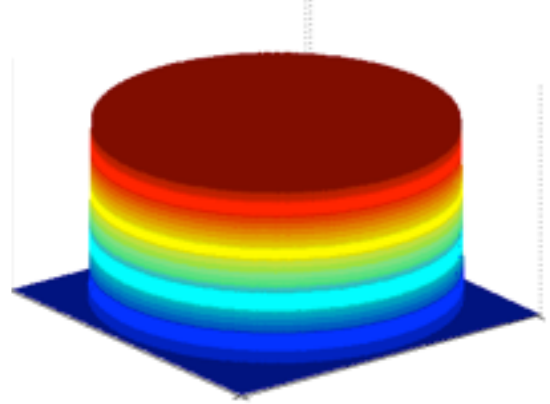
a ‘distance’ between two points

Epanechnikov kernel



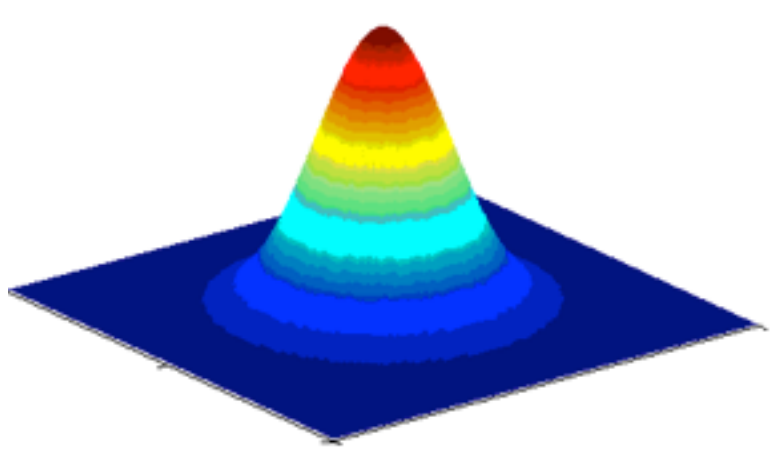
$$K(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} c(1 - \|\boldsymbol{x} - \boldsymbol{x}'\|^2) & \|\boldsymbol{x} - \boldsymbol{x}'\|^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Uniform kernel



$$K(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} c & \|\boldsymbol{x} - \boldsymbol{x}'\|^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Normal kernel



$$K(\boldsymbol{x}, \boldsymbol{x}') = c \exp\left(\frac{1}{2}\|\boldsymbol{x} - \boldsymbol{x}'\|^2\right)$$

Radially symmetric kernels

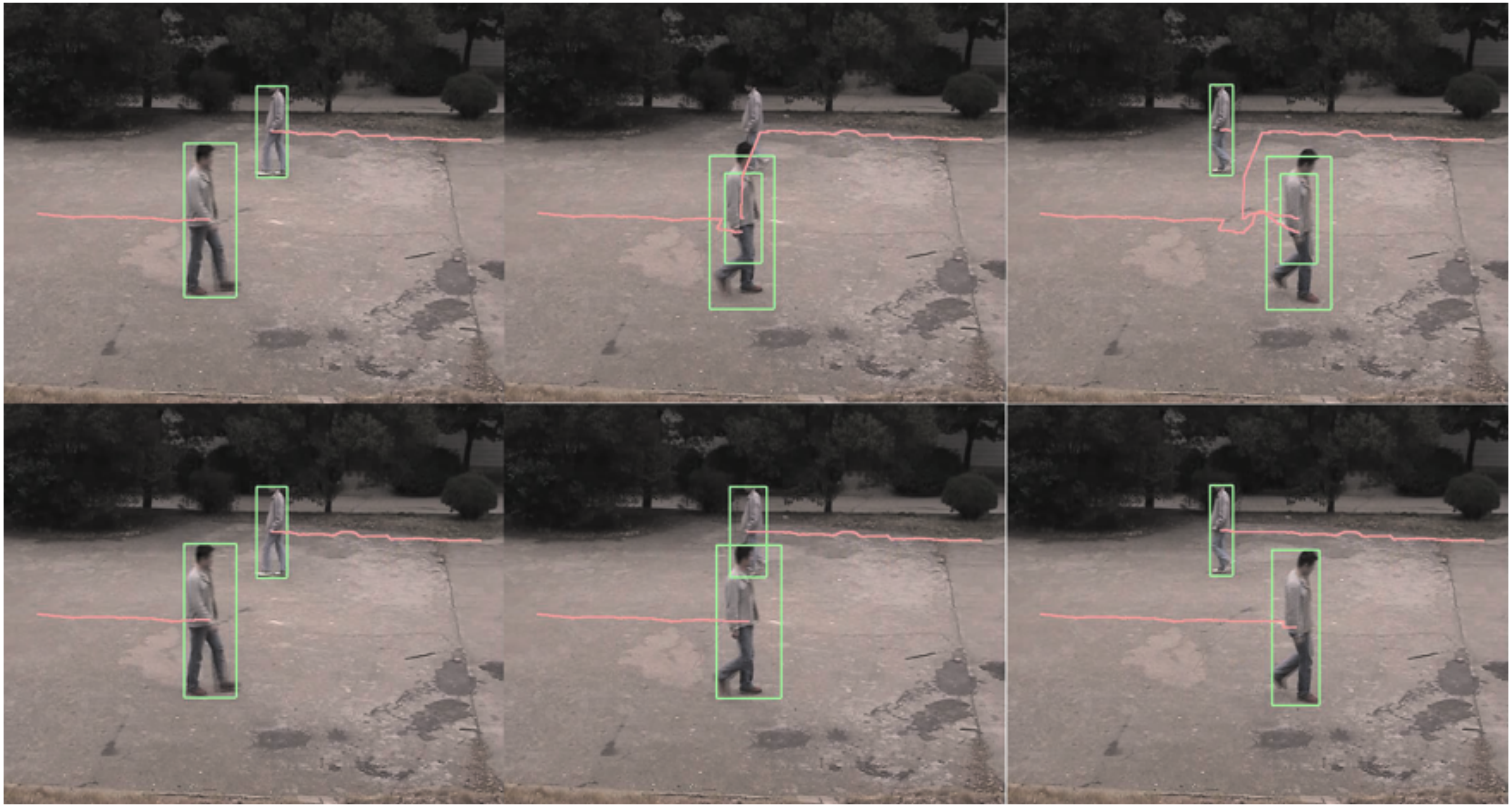
Radially symmetric kernels

…can be written in terms of its *profile*

$$K(\boldsymbol{x}, \boldsymbol{x}') = c \cdot k(\|\boldsymbol{x} - \boldsymbol{x}'\|^2)$$

profile

Connecting KDE and the Mean Shift Algorithm



Mean-Shift Tracker

16-385 Computer Vision (Kris Kitani)

Carnegie Mellon University

Mean-Shift Tracking

Given a set of points:

$$\{\boldsymbol{x}_s\}_{s=1}^{S} \qquad \boldsymbol{x}_s \in \mathcal{R}^d$$

and a kernel:

$$K(\boldsymbol{x}, \boldsymbol{x}')$$

Find the mean sample point:

$$\boldsymbol{x}$$

Mean-Shift Algorithm

`Initialize` $\boldsymbol{x}$

`While` $v(\boldsymbol{x}) > \epsilon$

1. `Compute mean-shift`

$$m(\boldsymbol{x}) = \frac{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s)\boldsymbol{x}_s}{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s)}$$

$$v(\boldsymbol{x}) = m(\boldsymbol{x}) - \boldsymbol{x}$$

2. `Update` $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})$

Where does this algorithm come from?

Mean-Shift Algorithm

Initialize $\boldsymbol{x}$

While $v(\boldsymbol{x}) > \epsilon$

Where does this come from?

1. Compute mean-shift

$$m(\boldsymbol{x}) = \frac{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s)\boldsymbol{x}_s}{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s)}$$

$$v(\boldsymbol{x}) = m(\boldsymbol{x}) - \boldsymbol{x}$$

2. Update $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})$

Where does this algorithm come from?

How is the KDE related to the mean shift algorithm?

Recall:

Kernel density estimate

(radially symmetric kernels)

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_n k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

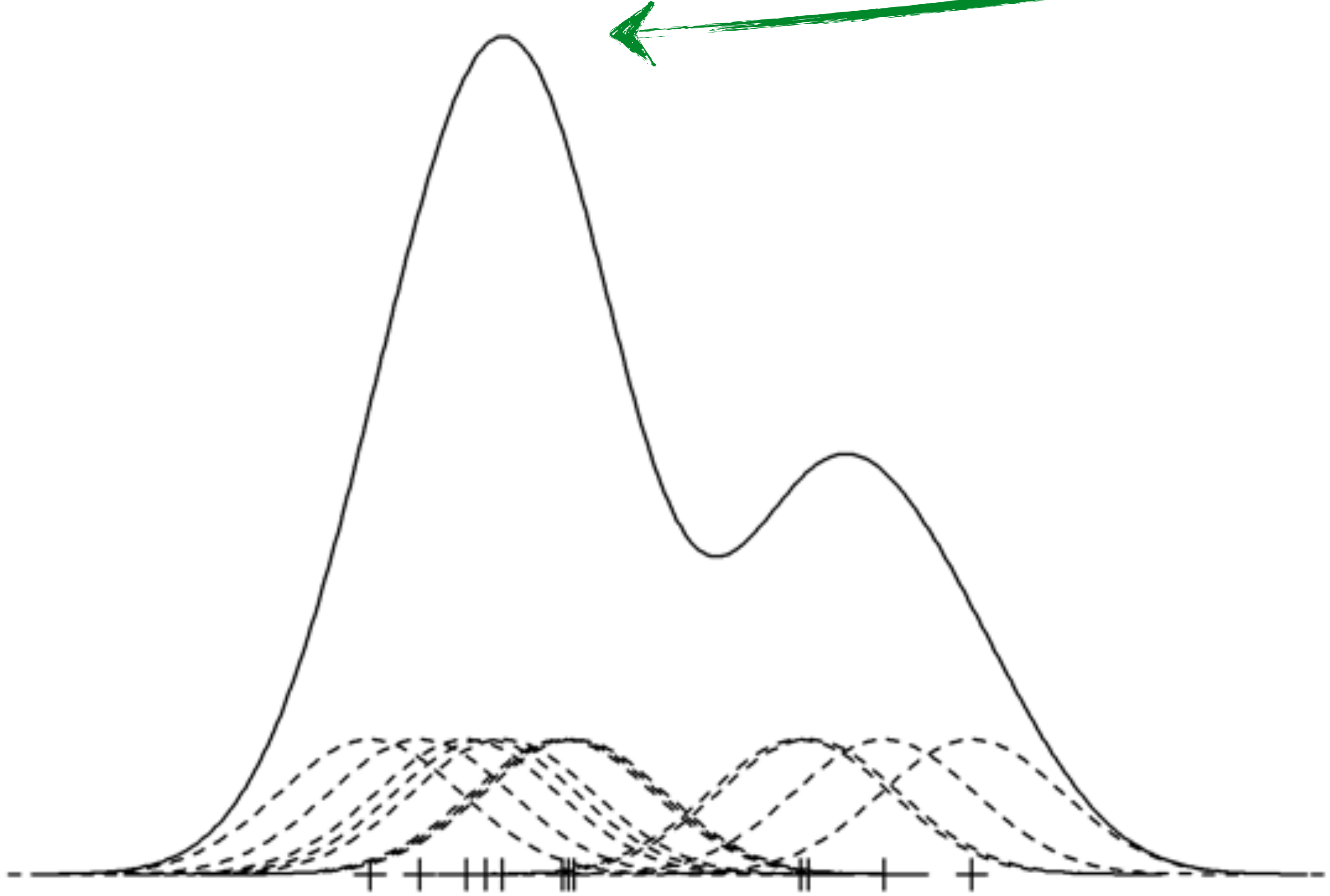
We can show that:

Gradient of the PDF is related to the mean shift vector

$$\nabla P(\boldsymbol{x}) \propto m(\boldsymbol{x})$$

The mean shift is a 'step' in the direction of the gradient of the KDE

In mean-shift tracking, we are trying to find this



which means we are trying to…

We are trying to optimize this:

$$\begin{aligned}\boldsymbol{x} &= \arg\max_{\boldsymbol{x}} P(\boldsymbol{x}) \\ &= \arg\max_{\boldsymbol{x}} \frac{1}{N} c \sum_{n} k(||\boldsymbol{x} - \boldsymbol{x}_n||^2)\end{aligned}$$

usually non-linear   non-parametric

How do we optimize this non-linear function?

We are trying to optimize this:

$$\boldsymbol{x} = \arg\max_{\boldsymbol{x}} P(\boldsymbol{x})$$
$$= \arg\max_{\boldsymbol{x}} \frac{1}{N} c \sum_n k(||\boldsymbol{x} - \boldsymbol{x}_n||^2)$$

usually non-linear   non-parametric

How do we optimize this non-linear function?

compute partial derivatives, gradient descent

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_{n} k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Compute the gradient

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_{n} k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} c \sum_{n} \nabla k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Expand the gradient (algebra)

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_n k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} c \sum_n \nabla k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Expand gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n (\boldsymbol{x} - \boldsymbol{x}_n) k'(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_n k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} c \sum_n \nabla k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Expand gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n (\boldsymbol{x} - \boldsymbol{x}_n) k'(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Call the gradient of the kernel function g

$$k'(\cdot) = -g(\cdot)$$

$$P(\boldsymbol{x}) = \frac{1}{N} c \sum_n k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} c \sum_n \nabla k(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

Expand gradient

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n (\boldsymbol{x} - \boldsymbol{x}_n) k'(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

change of notation
(kernel-shadow pairs)

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n (\boldsymbol{x}_n - \boldsymbol{x}) g(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

keep this in memory: $k'(\cdot) = -g(\cdot)$

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n (\boldsymbol{x}_n - \boldsymbol{x}) g(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

multiply it out

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n \boldsymbol{x}_n g(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2) - \frac{1}{N} 2c \sum_n \boldsymbol{x} g(\|\boldsymbol{x} - \boldsymbol{x}_n\|^2)$$

too long!
(use short hand notation)

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n \boldsymbol{x}_n g_n - \frac{1}{N} 2c \sum_n \boldsymbol{x} g_n$$

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n \boldsymbol{x}_n g_n - \frac{1}{N} 2c \sum_n \boldsymbol{x} g_n$$

multiply by one!

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n \boldsymbol{x}_n g_n \left(\frac{\sum_n g_n}{\sum_n g_n} \right) - \frac{1}{N} 2c \sum_n \boldsymbol{x} g_n$$

collecting like terms…

$$\nabla P(\boldsymbol{x}) = \frac{1}{N} 2c \sum_n g_n \left(\frac{\sum_n \boldsymbol{x}_n g_n}{\sum_n g_n} - \boldsymbol{x} \right)$$

Does this look familiar?

mean shift

$$\nabla P(\boldsymbol{x}) = \underbrace{\frac{1}{N} 2c \sum_n g_n}_{\text{constant}} \underbrace{\left(\frac{\sum_n \boldsymbol{x}_n g_n}{\sum_n g_n} - \boldsymbol{x} \right)}_{\text{mean shift!}}$$

The **mean shift** is a ‘step’ in the direction of the gradient of the KDE

$$\boldsymbol{v}(\boldsymbol{x}) = \left(\frac{\sum_n \boldsymbol{x}_n g_n}{\sum_n g_n} - \boldsymbol{x} \right) = \frac{\nabla P(\boldsymbol{x})}{\frac{1}{N} 2c \sum_n g_n}$$

Gradient ascent with adaptive step size

Mean-Shift Algorithm

Initialize $\boldsymbol{x}$

While $v(\boldsymbol{x}) > \epsilon$

1. Compute mean-shift

$$m(\boldsymbol{x}) = \frac{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s)\boldsymbol{x}_s}{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s)}$$

$$v(\boldsymbol{x}) = m(\boldsymbol{x}) - \boldsymbol{x}$$

2. Update $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})$

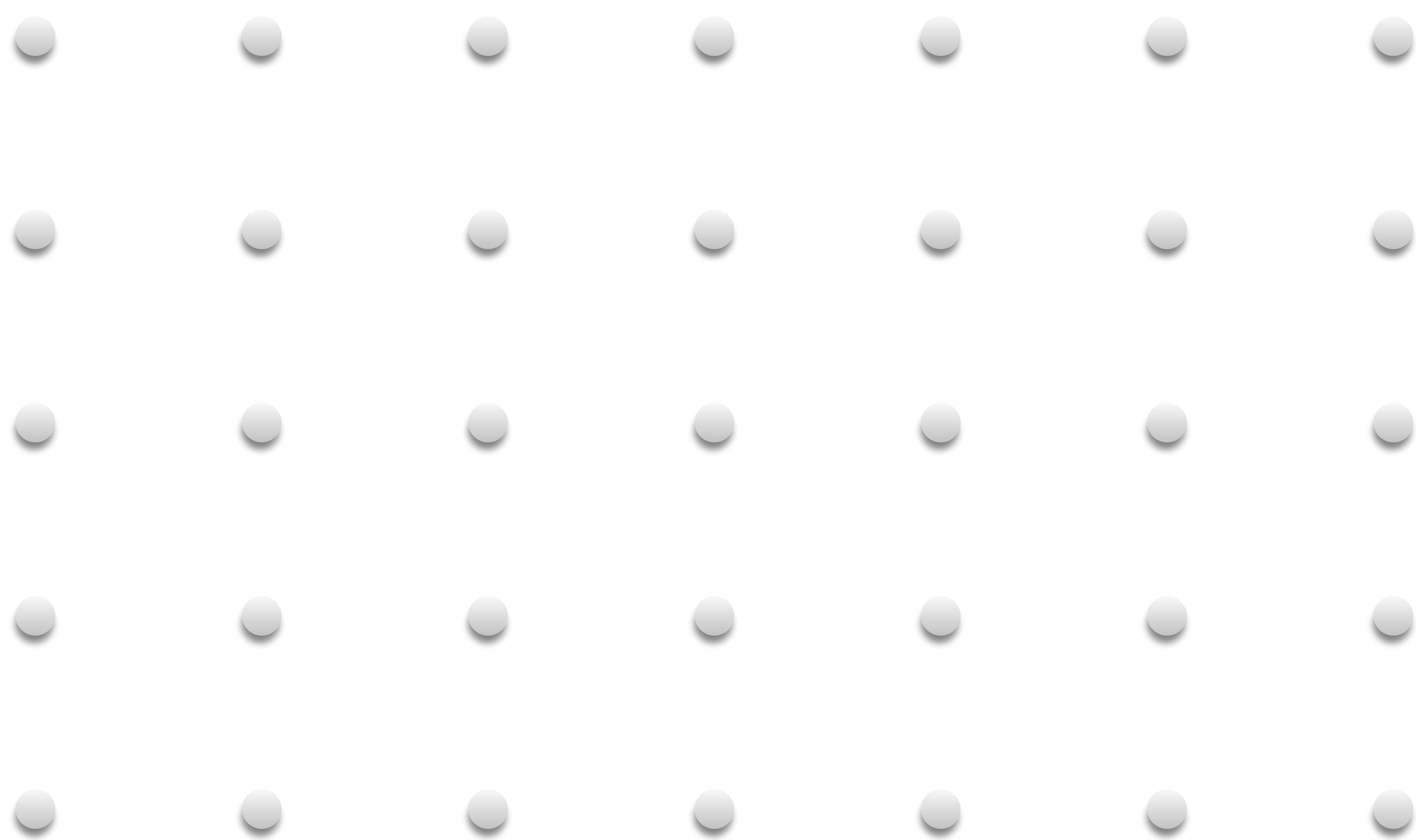
gradient with
adaptive step size

$$\frac{\nabla P(\boldsymbol{x})}{\frac{1}{N} 2c \sum_n g_n}$$

Everything up to now has been about distributions over samples…

Dealing with images

Pixels for a lattice, spatial density is the same everywhere!



What can we do?

Consider a set of points: $\{\boldsymbol{x}_s\}_{s=1}^S$ $\boldsymbol{x}_s \in \mathcal{R}^d$

Associated weights: $w(\boldsymbol{x}_s)$

Sample mean: $m(\boldsymbol{x}) = \frac{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s) \boldsymbol{x}_s}{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s)}$

Mean shift: $m(\boldsymbol{x}) - \boldsymbol{x}$

Mean-Shift Algorithm

Initialize $\boldsymbol{x}$

While $v(\boldsymbol{x}) > \epsilon$

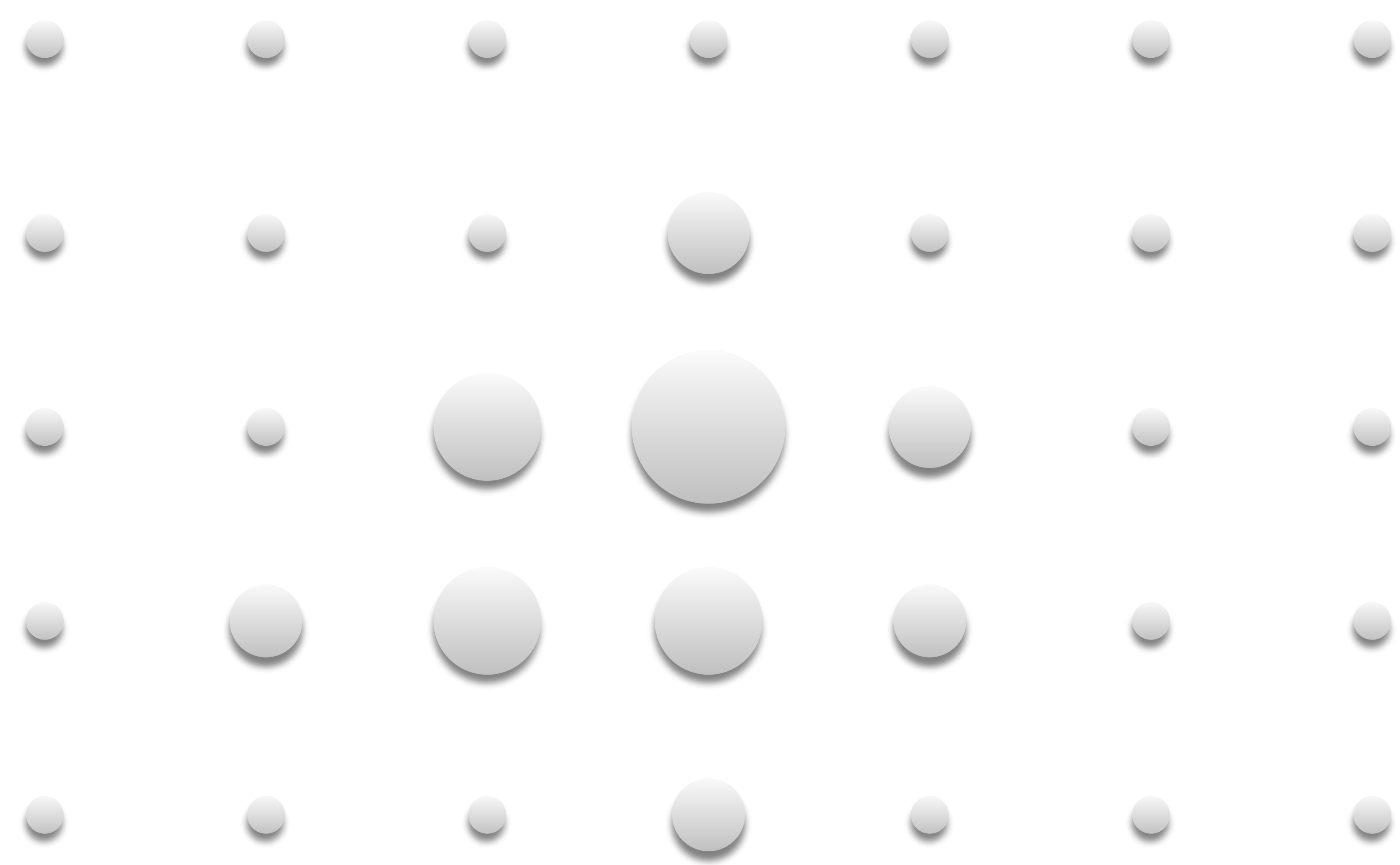
1. Compute mean-shift

$$m(\boldsymbol{x}) = \frac{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s) \boldsymbol{x}_s}{\sum_s K(\boldsymbol{x}, \boldsymbol{x}_s) w(\boldsymbol{x}_s)}$$

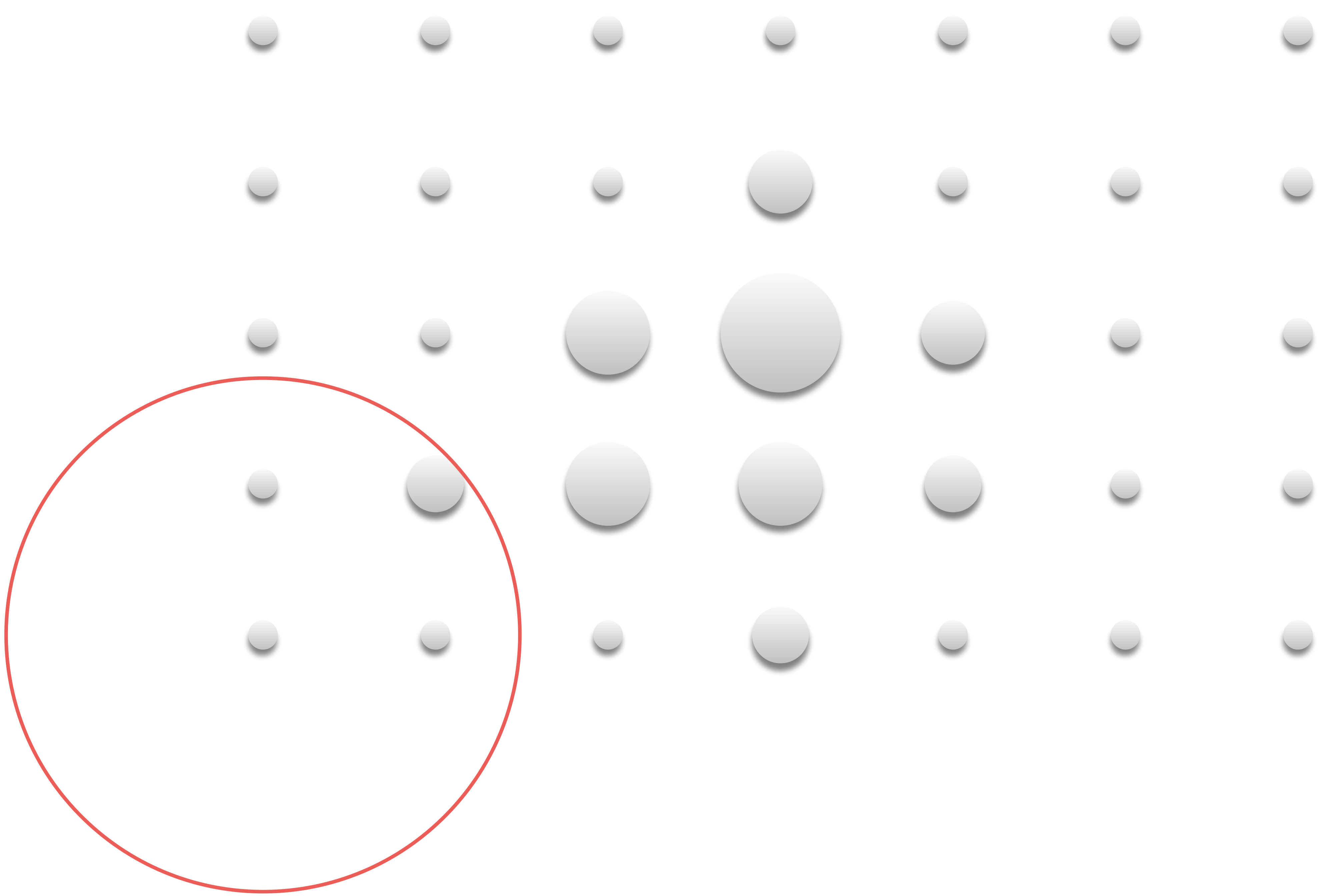
$$v(\boldsymbol{x}) = m(\boldsymbol{x}) - \boldsymbol{x}$$

2. Update $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})$

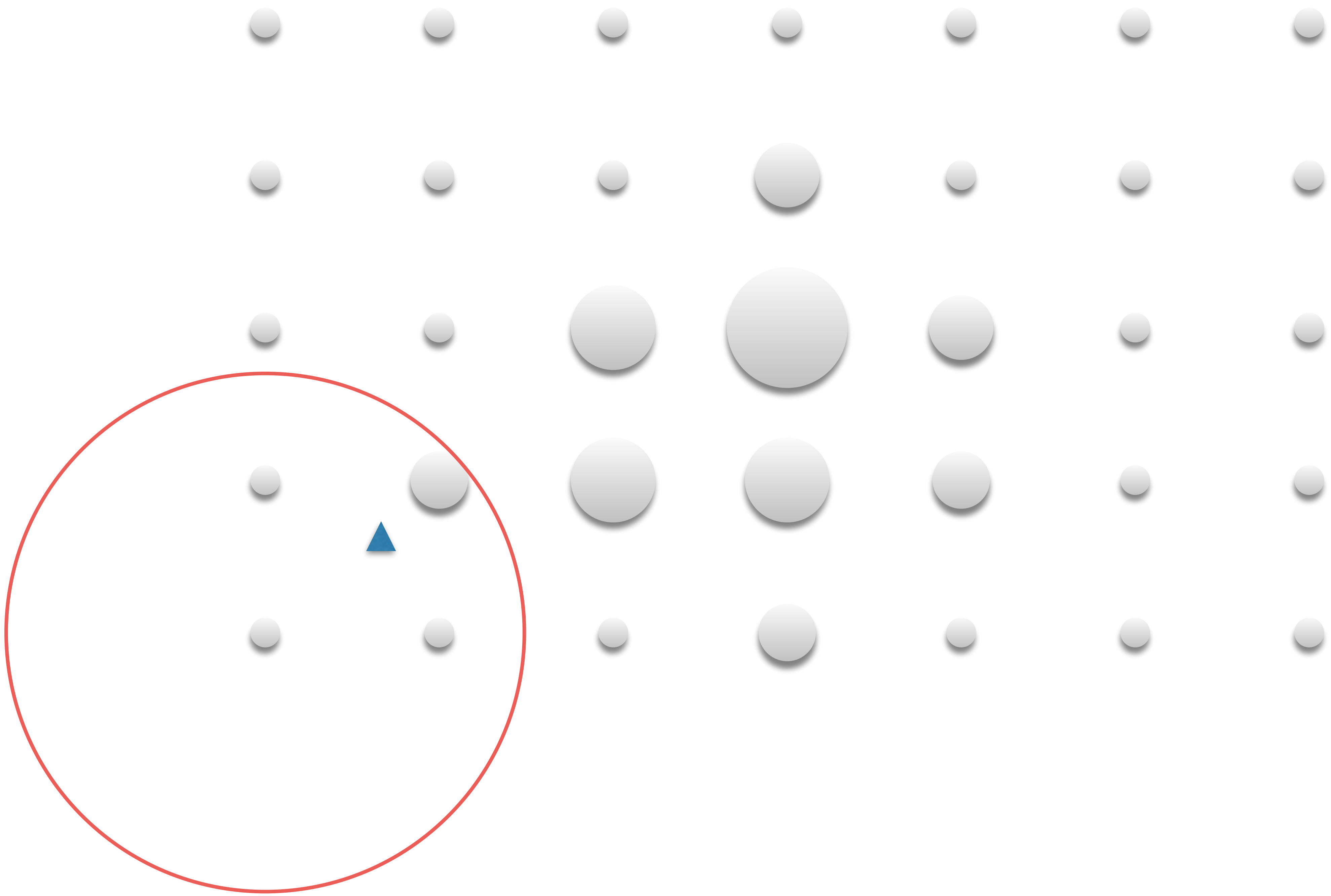
For images, each pixel is point with a weight



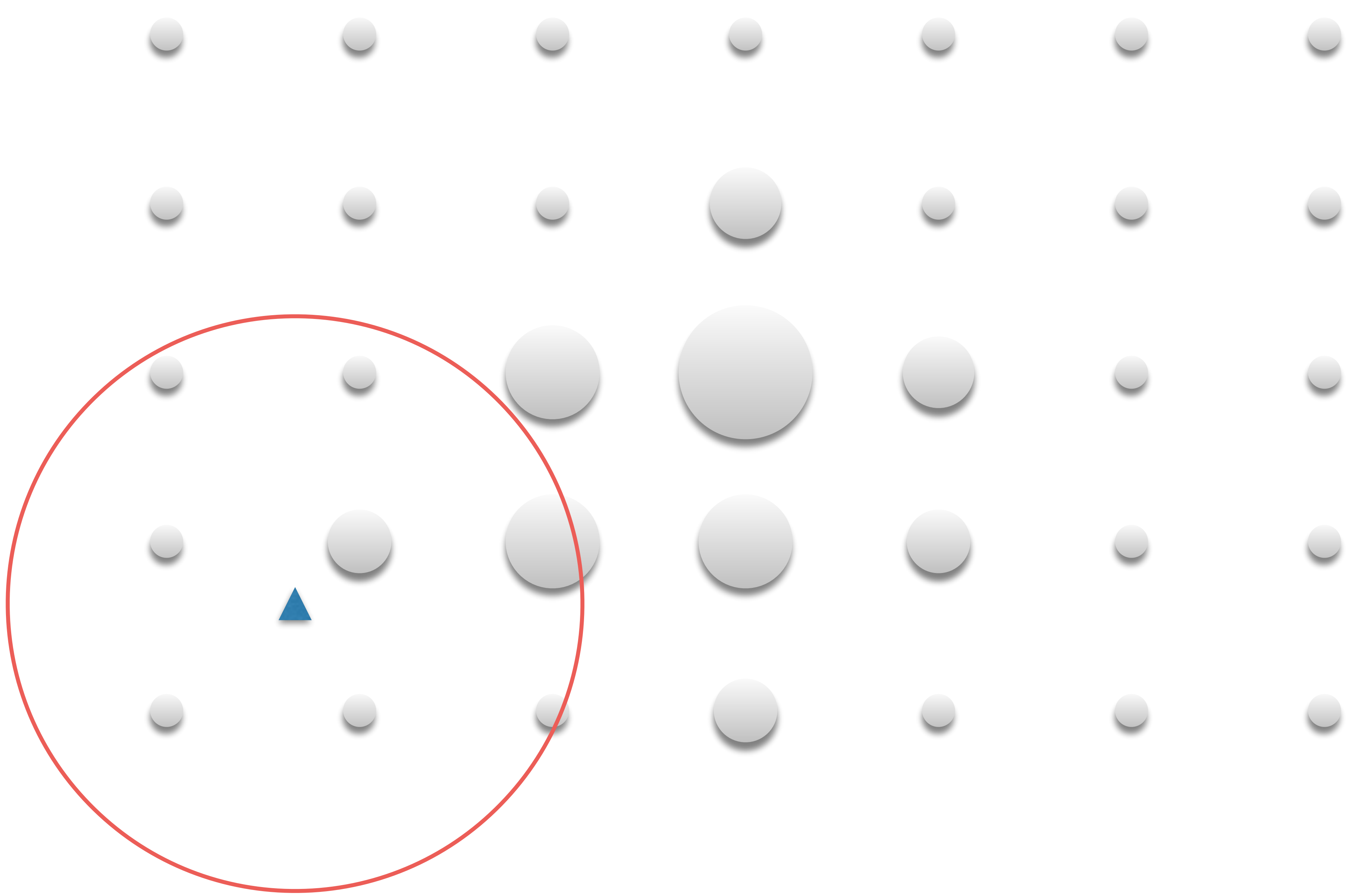
For images, each pixel is point with a weight



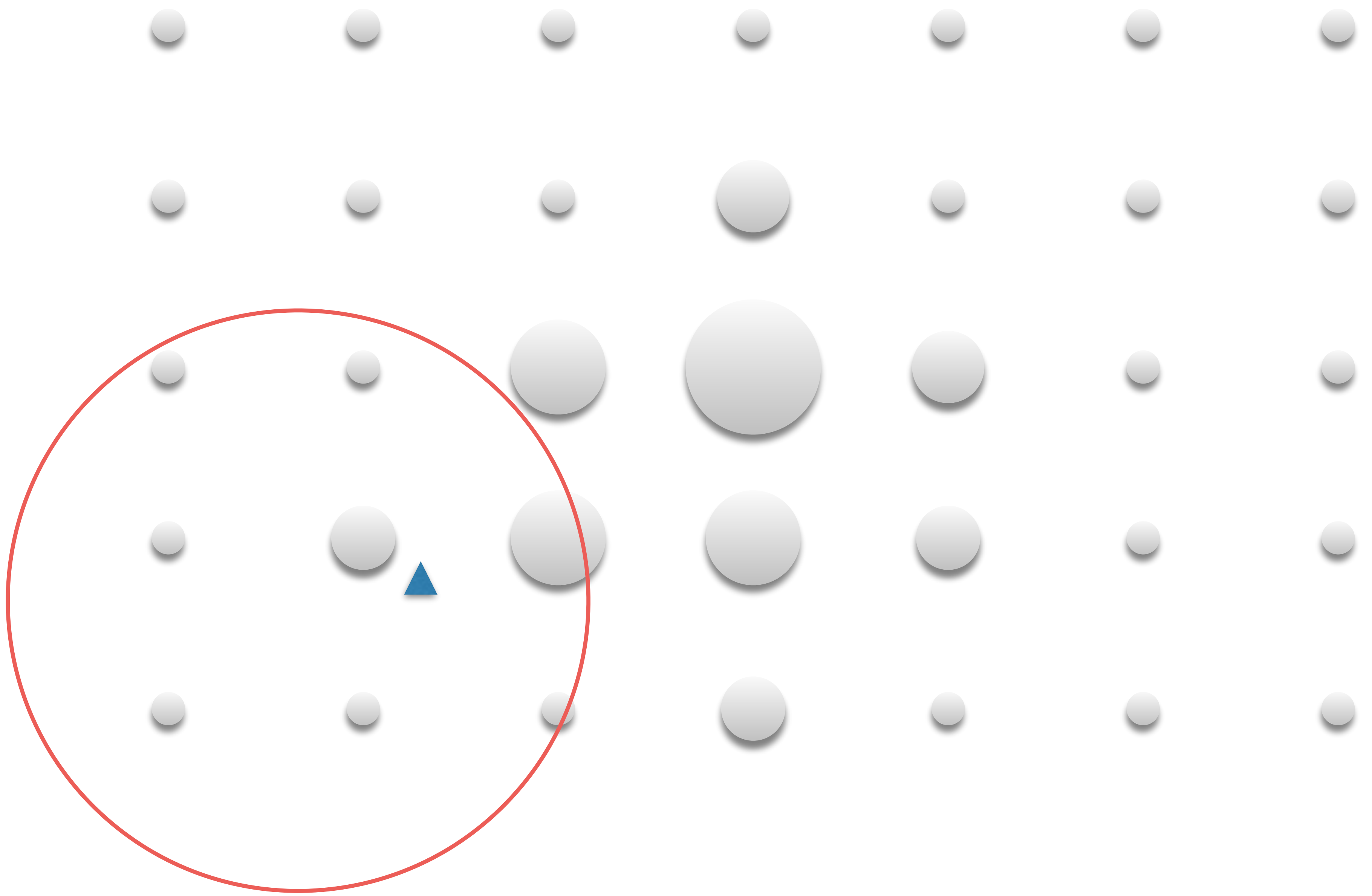
For images, each pixel is point with a weight



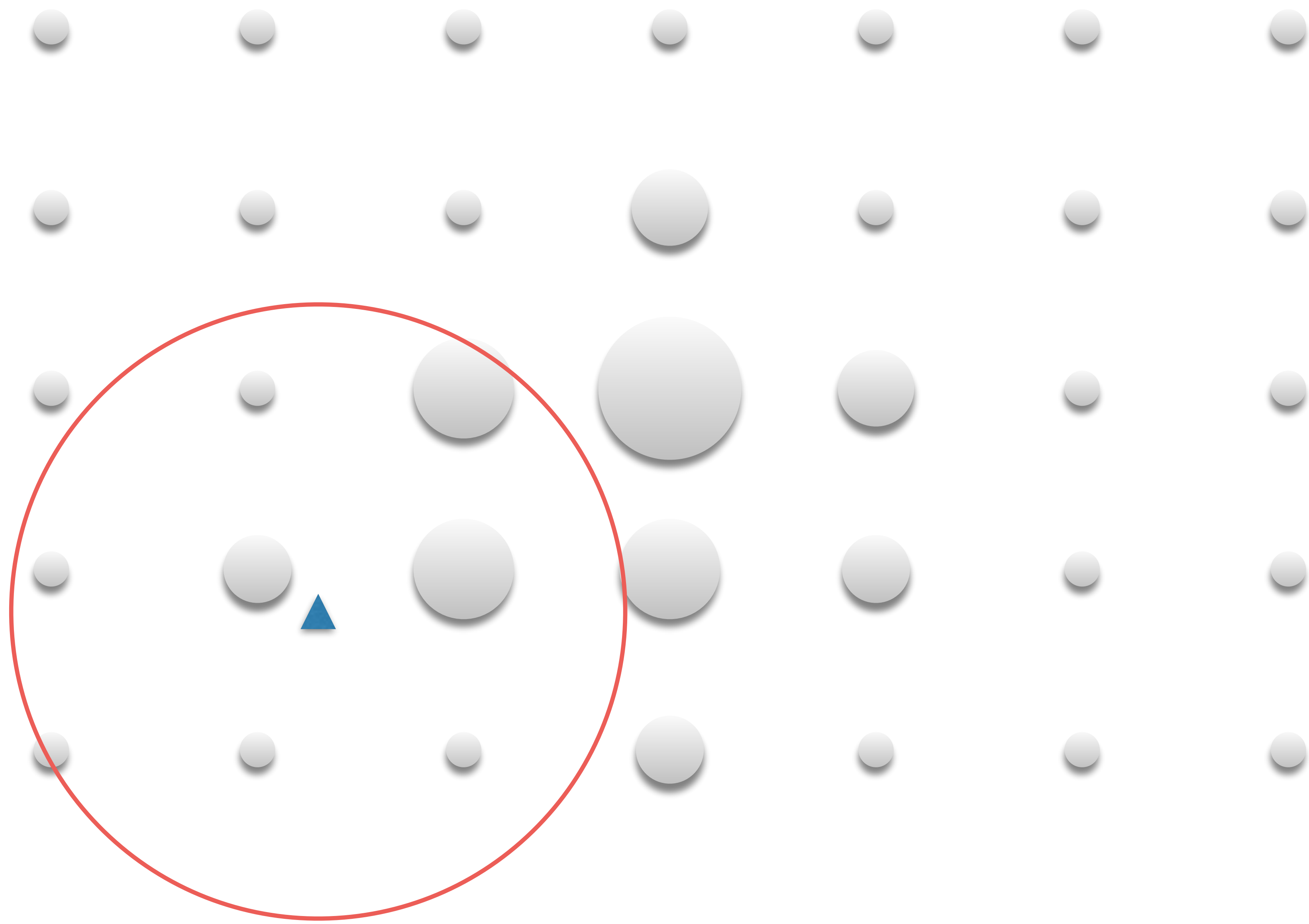
For images, each pixel is point with a weight



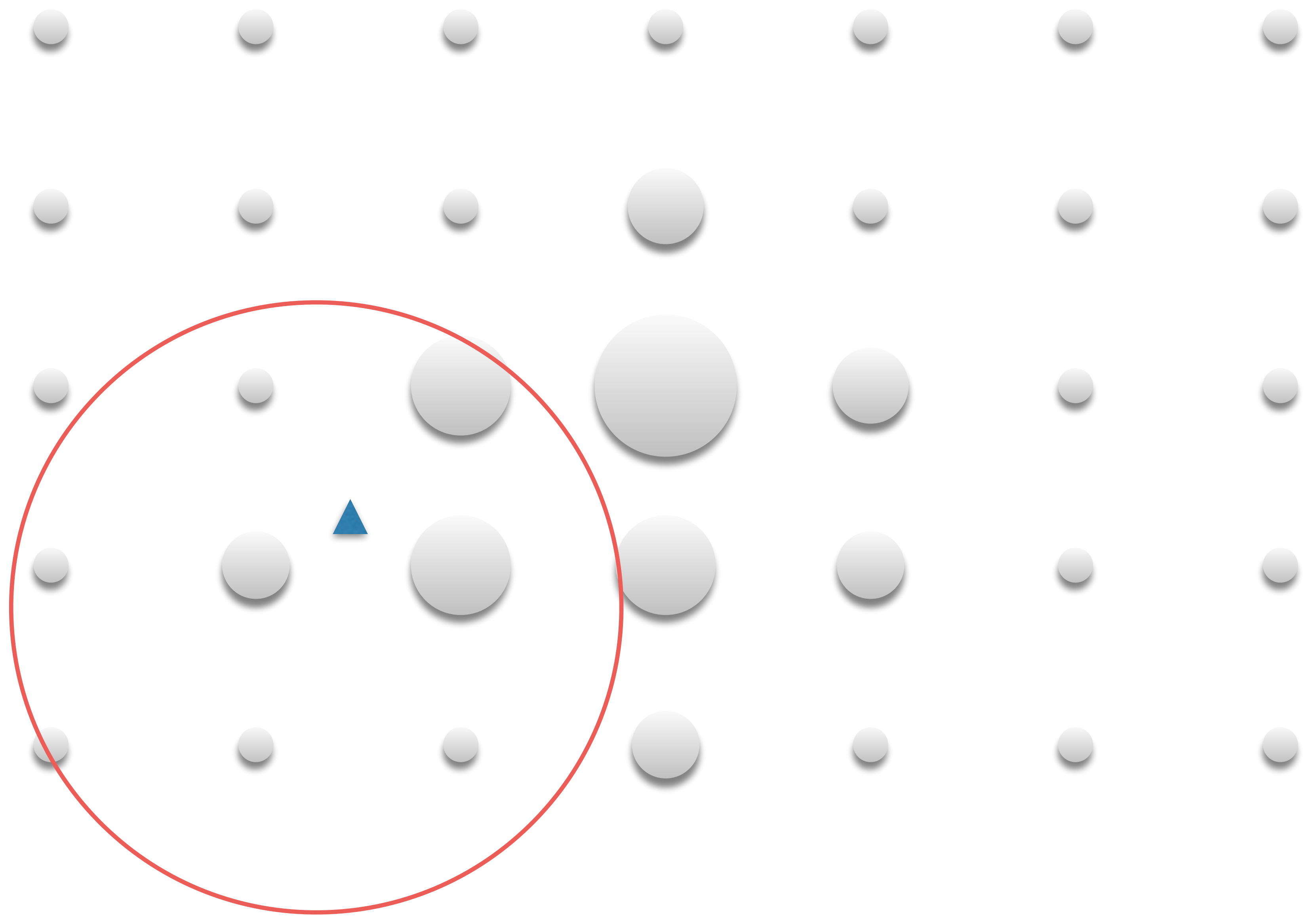
For images, each pixel is point with a weight



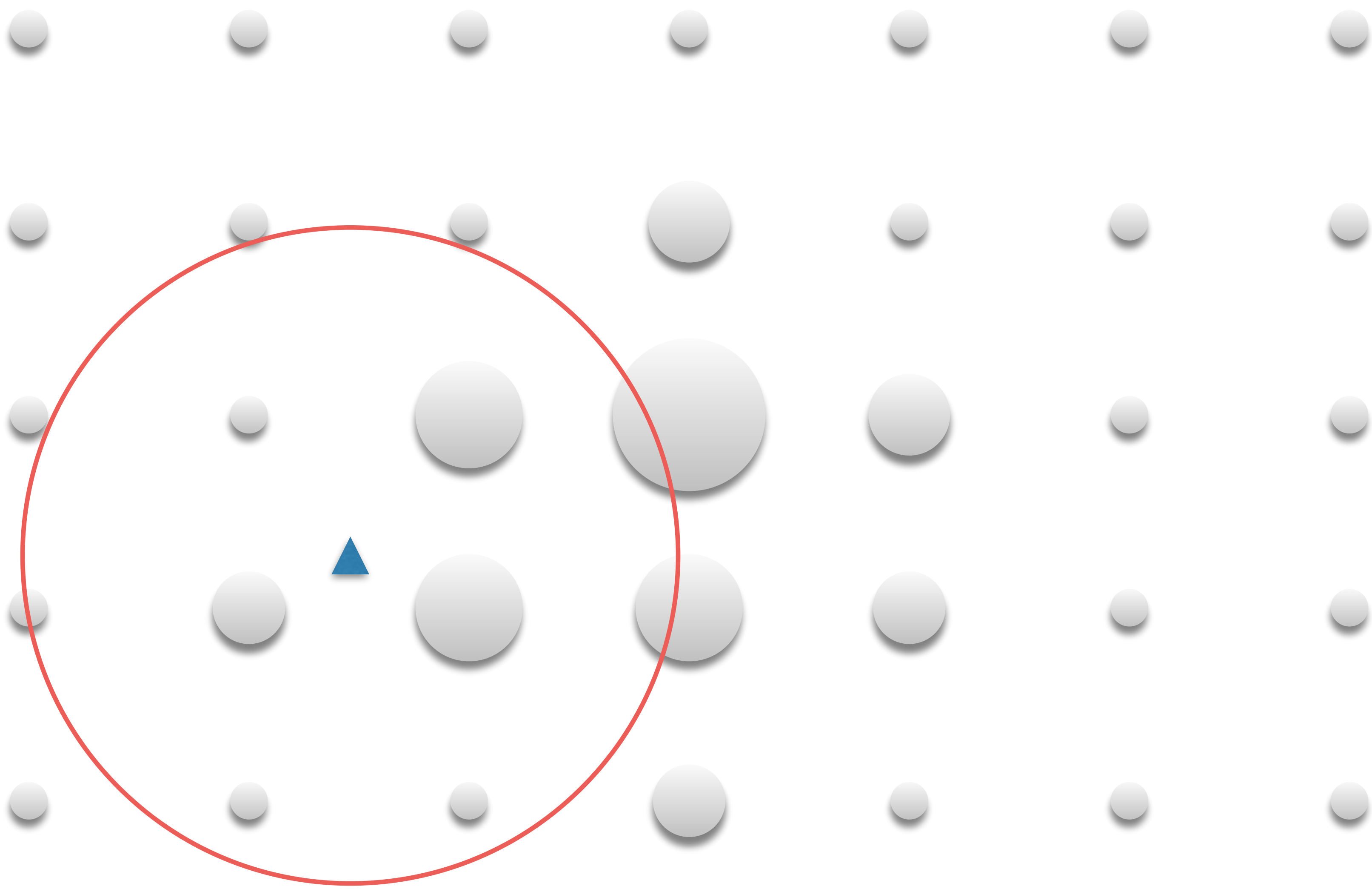
For images, each pixel is point with a weight



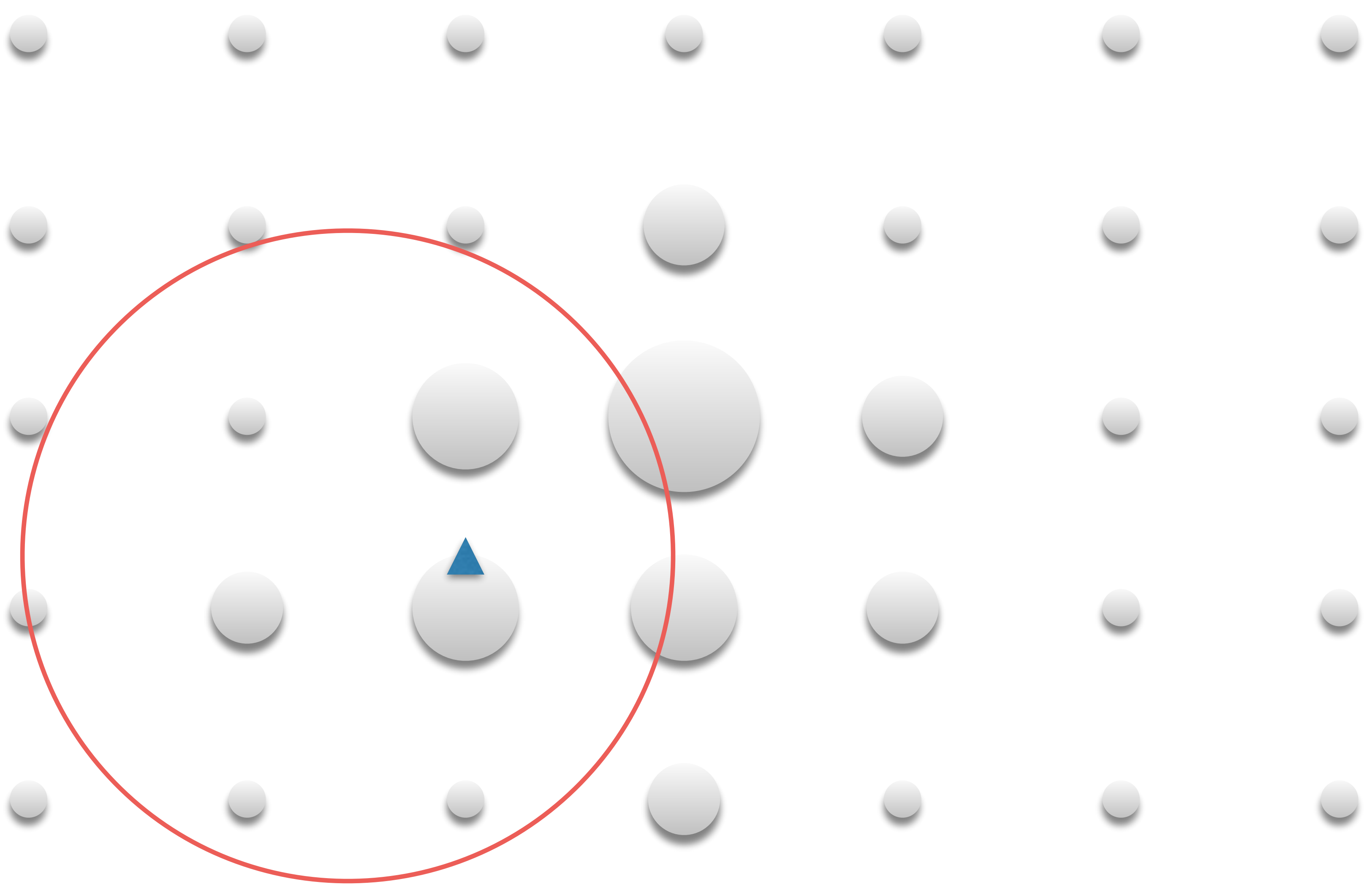
For images, each pixel is point with a weight



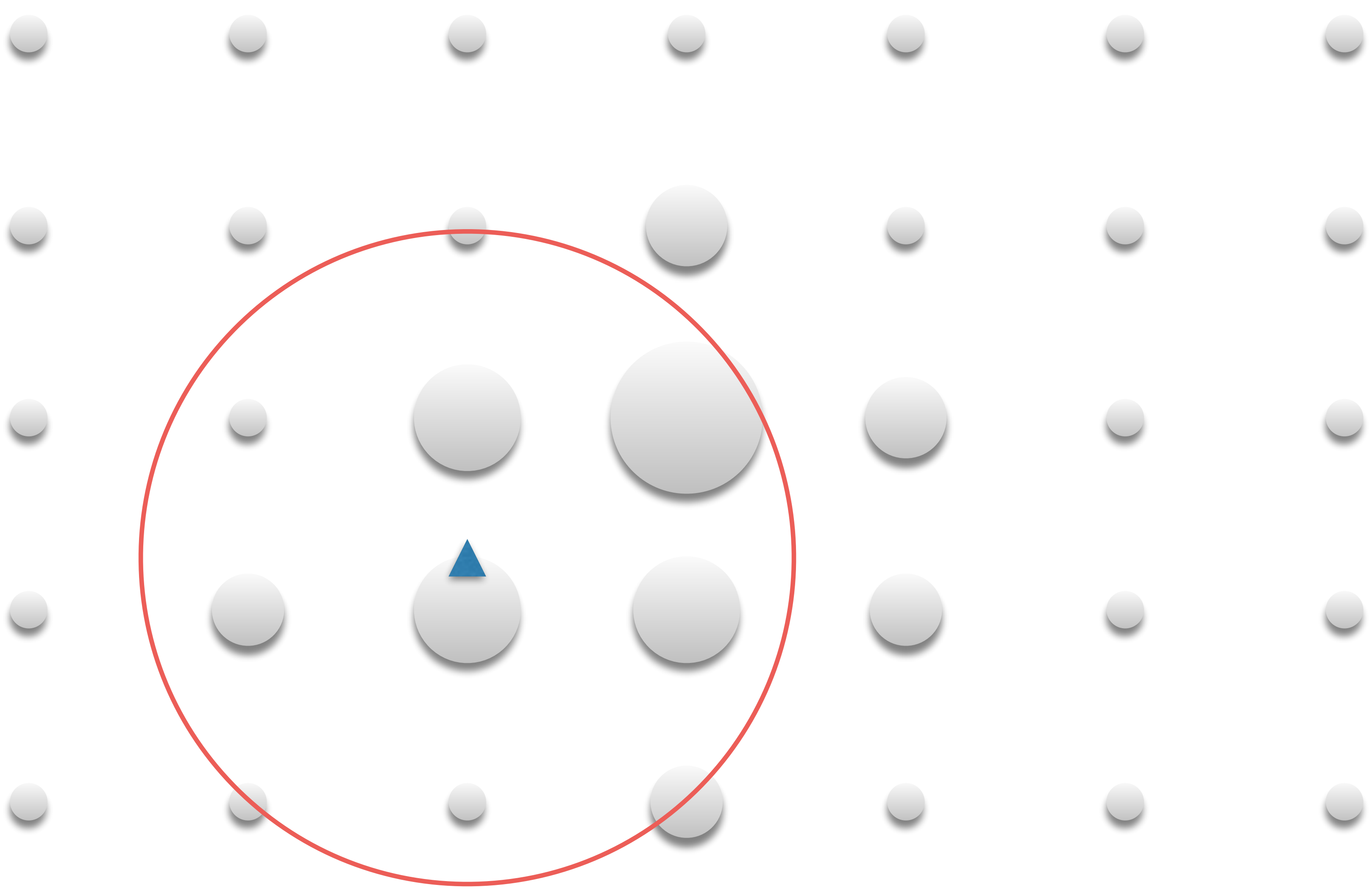
For images, each pixel is point with a weight



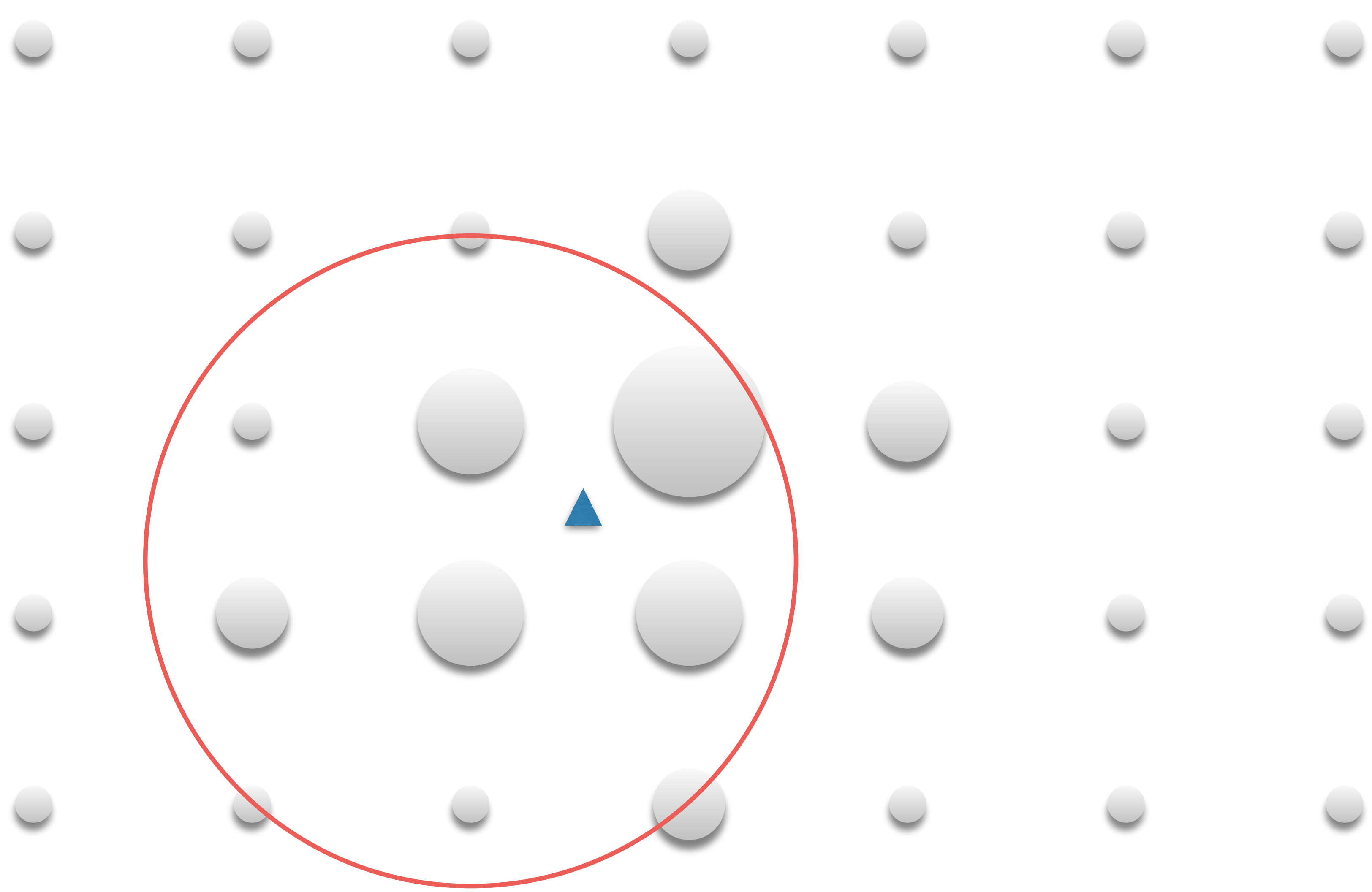
For images, each pixel is point with a weight



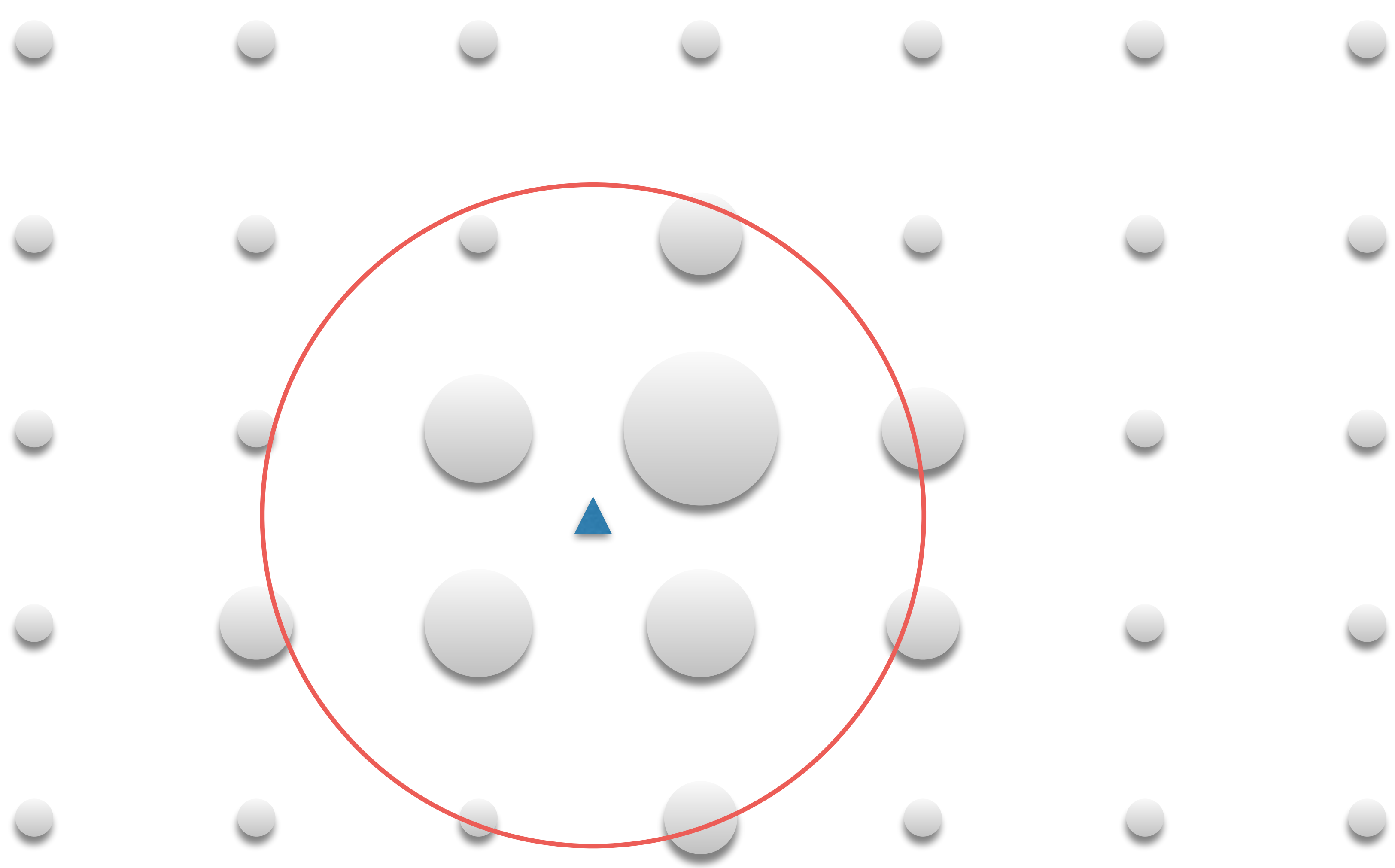
For images, each pixel is point with a weight



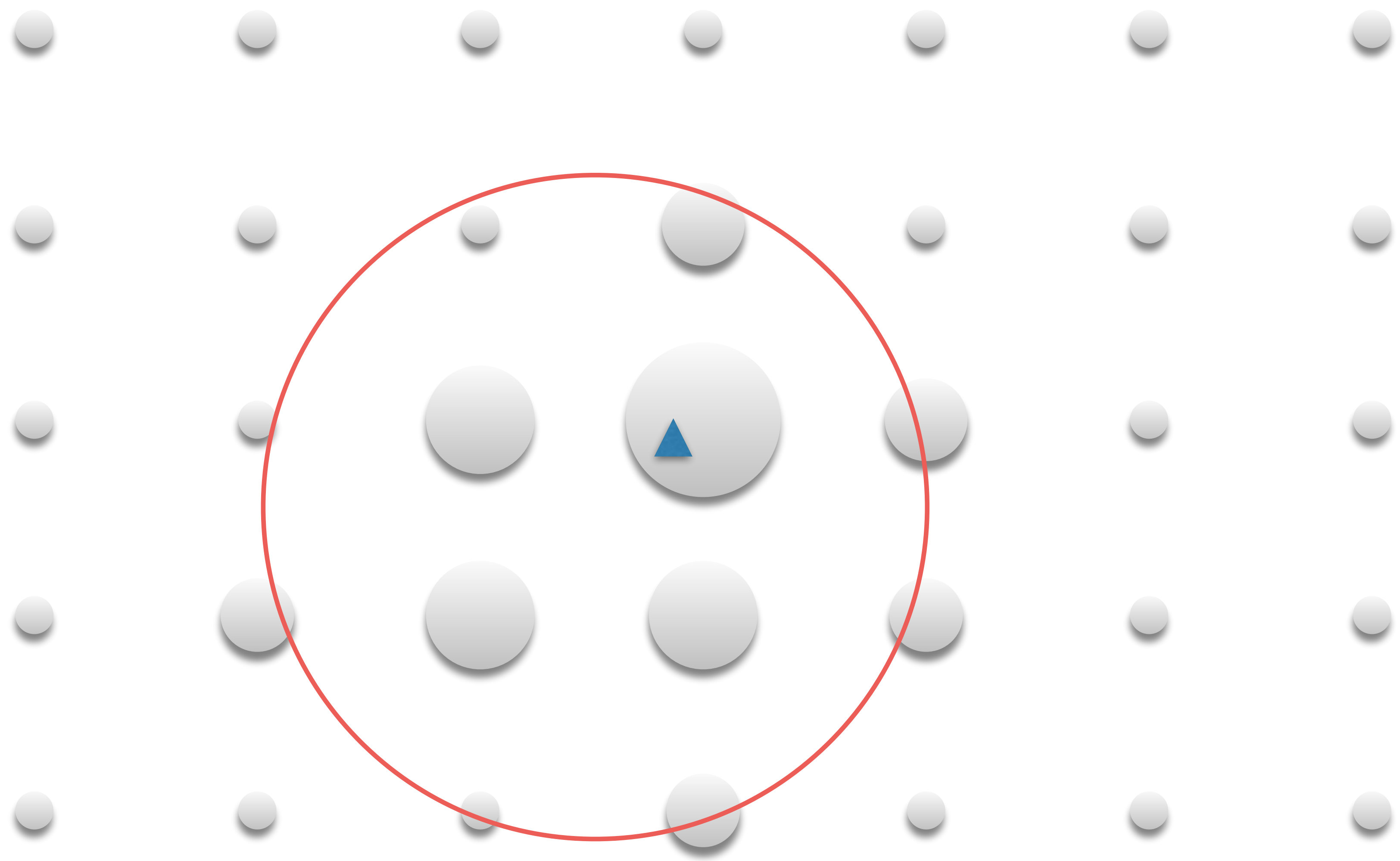
For images, each pixel is point with a weight



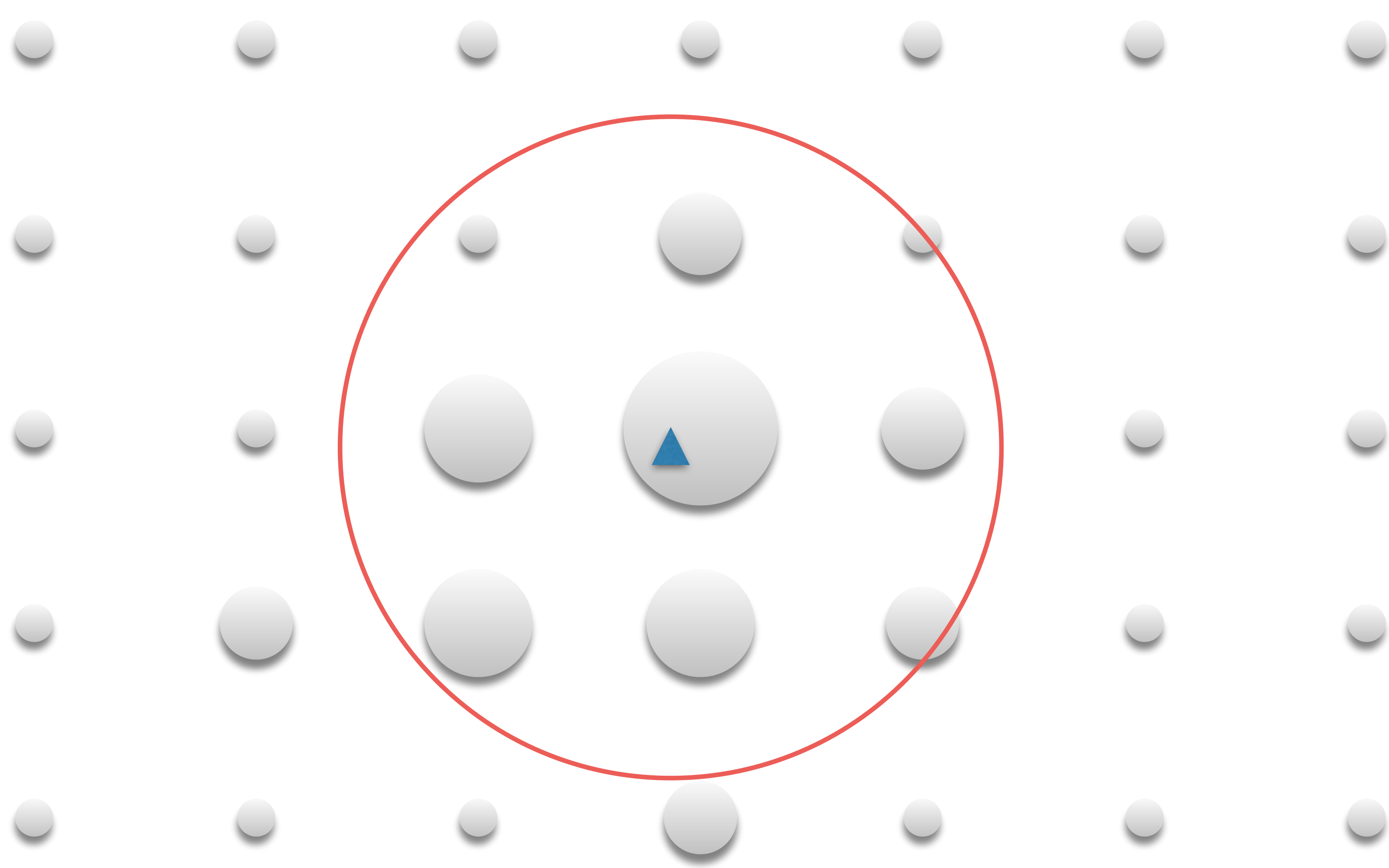
For images, each pixel is point with a weight



For images, each pixel is point with a weight

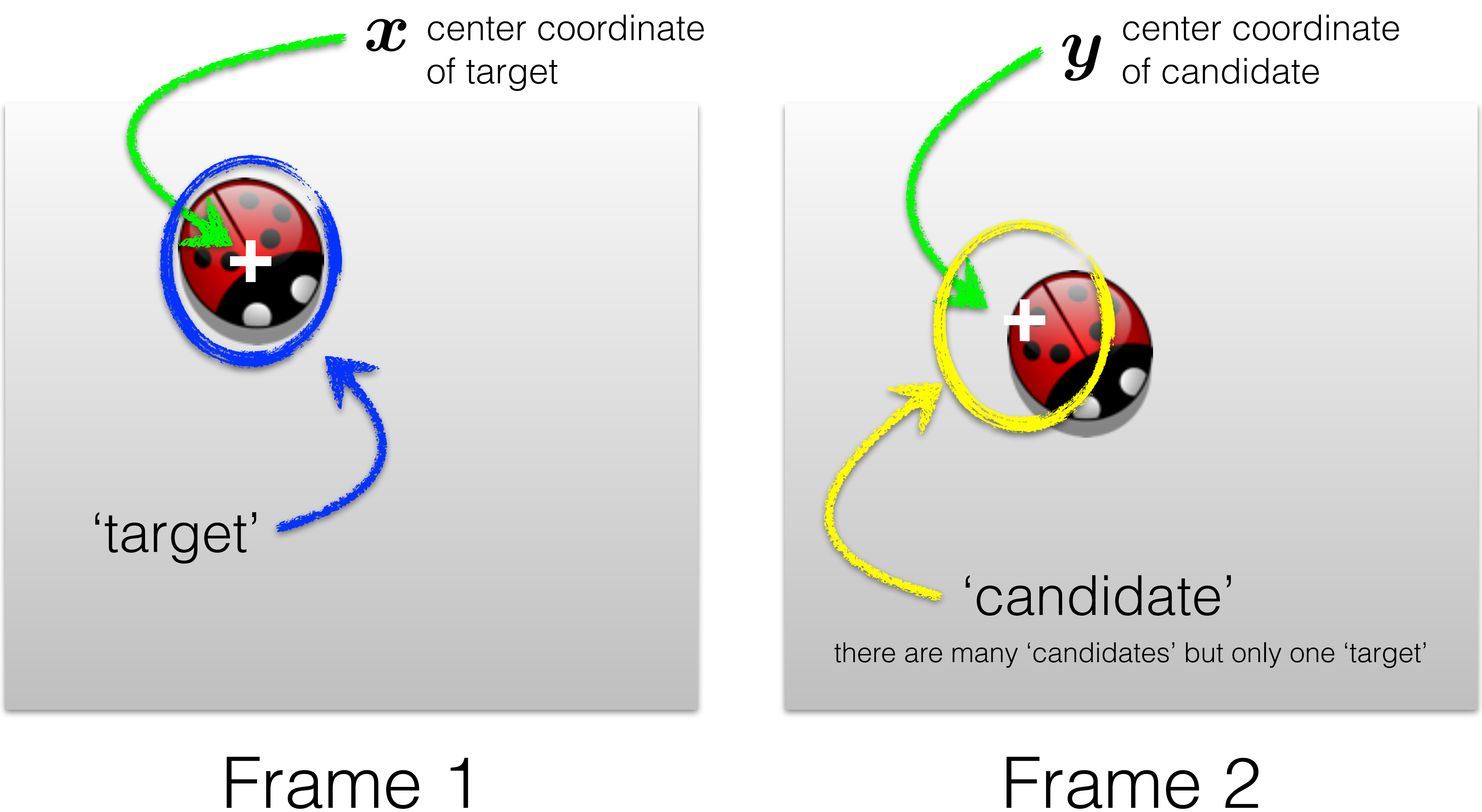


For images, each pixel is point with a weight



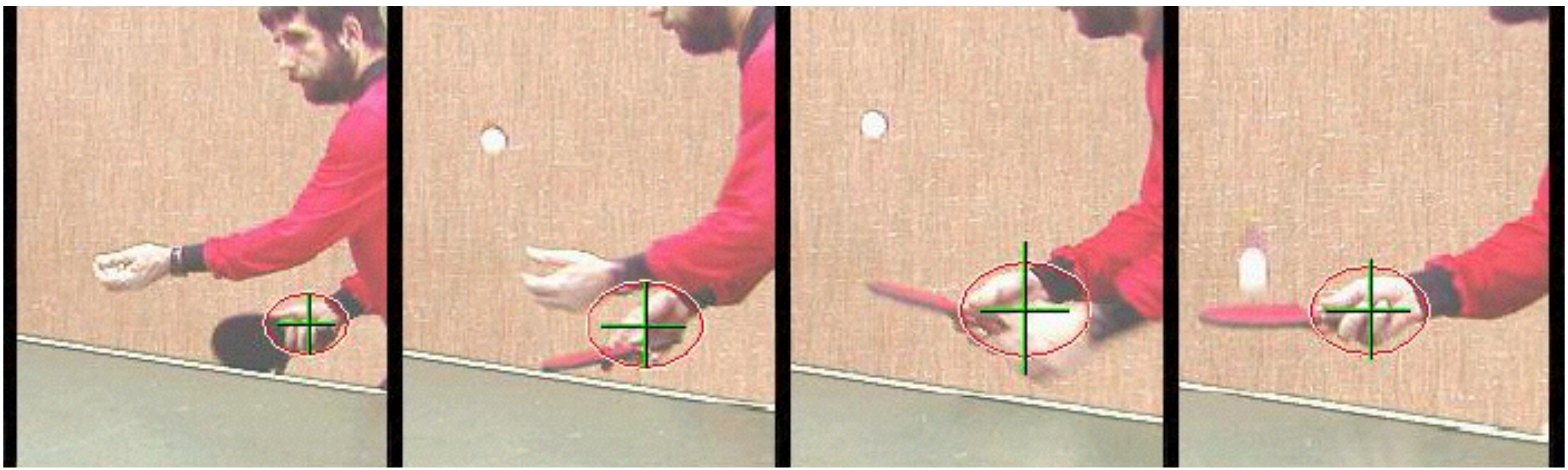
Finally… mean shift tracking in video

Goal: find the best candidate location in frame 2

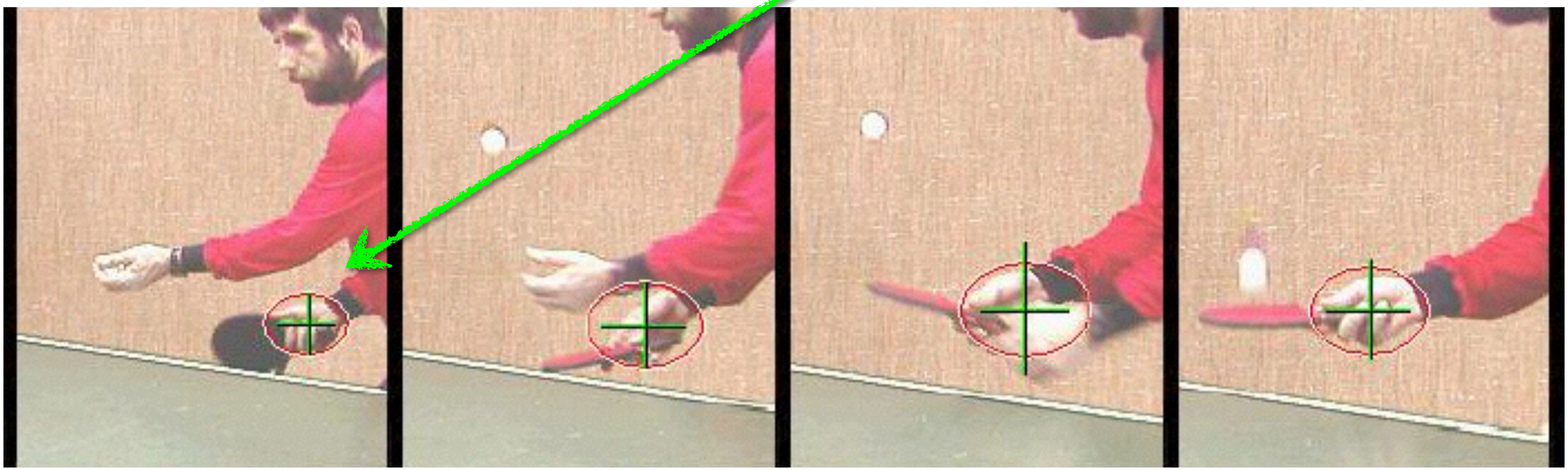


Use the mean shift algorithm
to find the best candidate location

Non-rigid object tracking

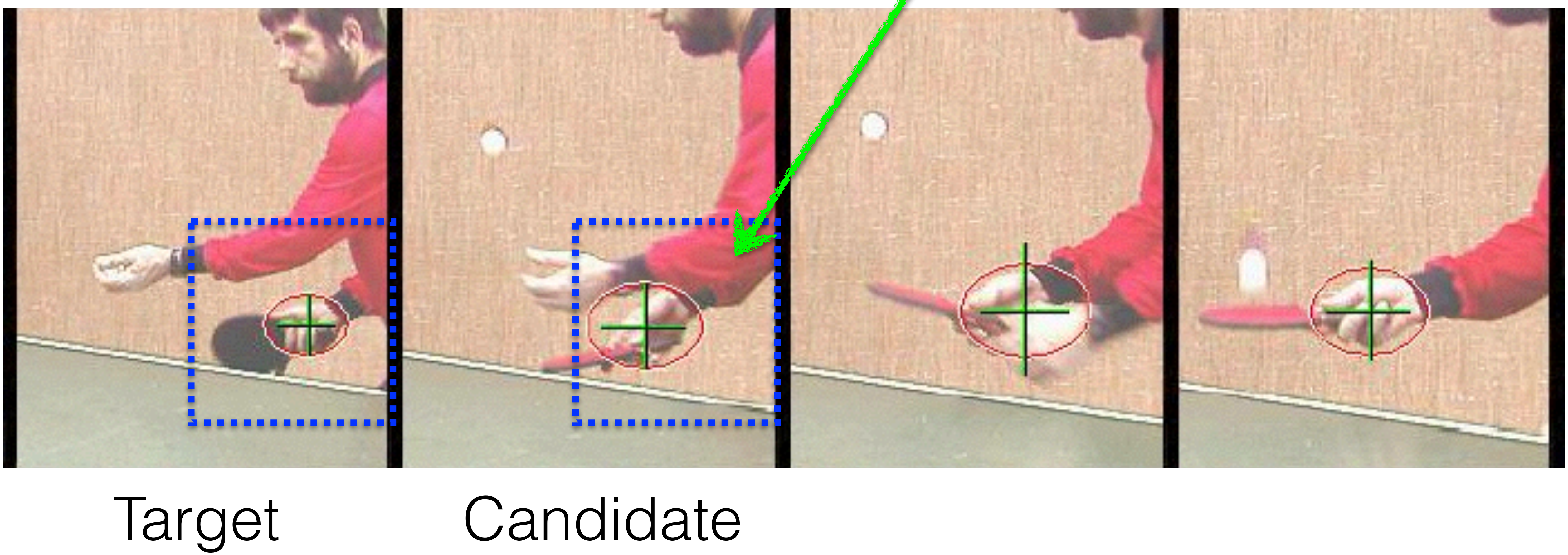


Compute a descriptor for the target

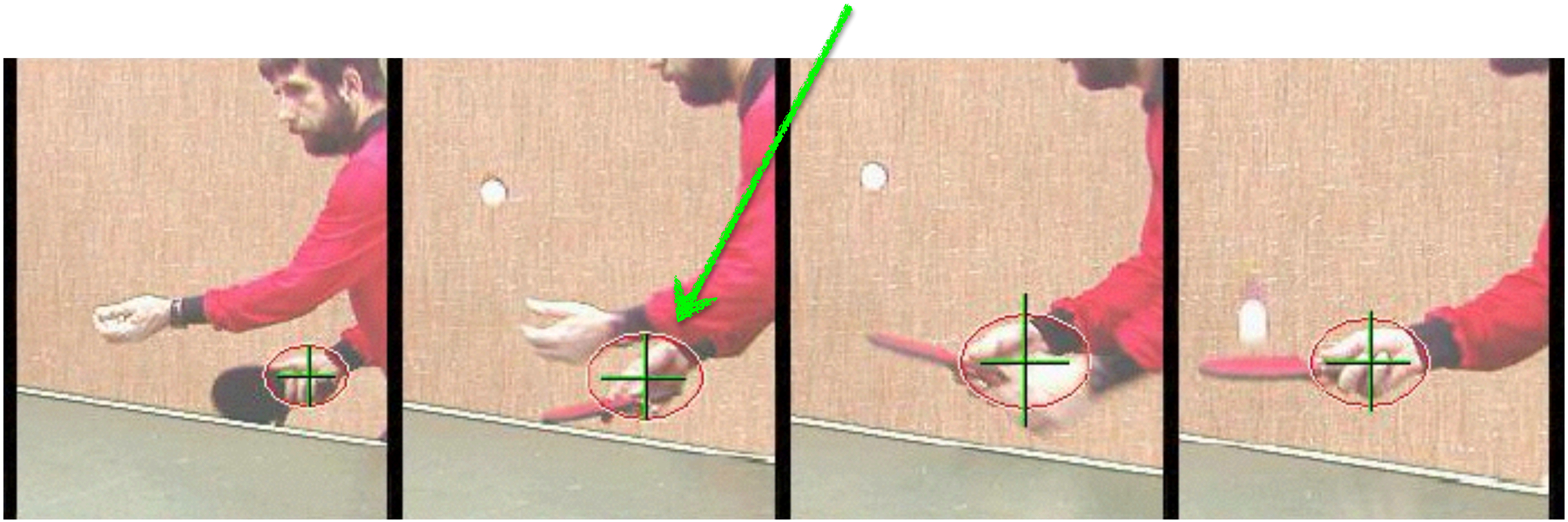


Target

Search for similar descriptor in neighborhood in next frame

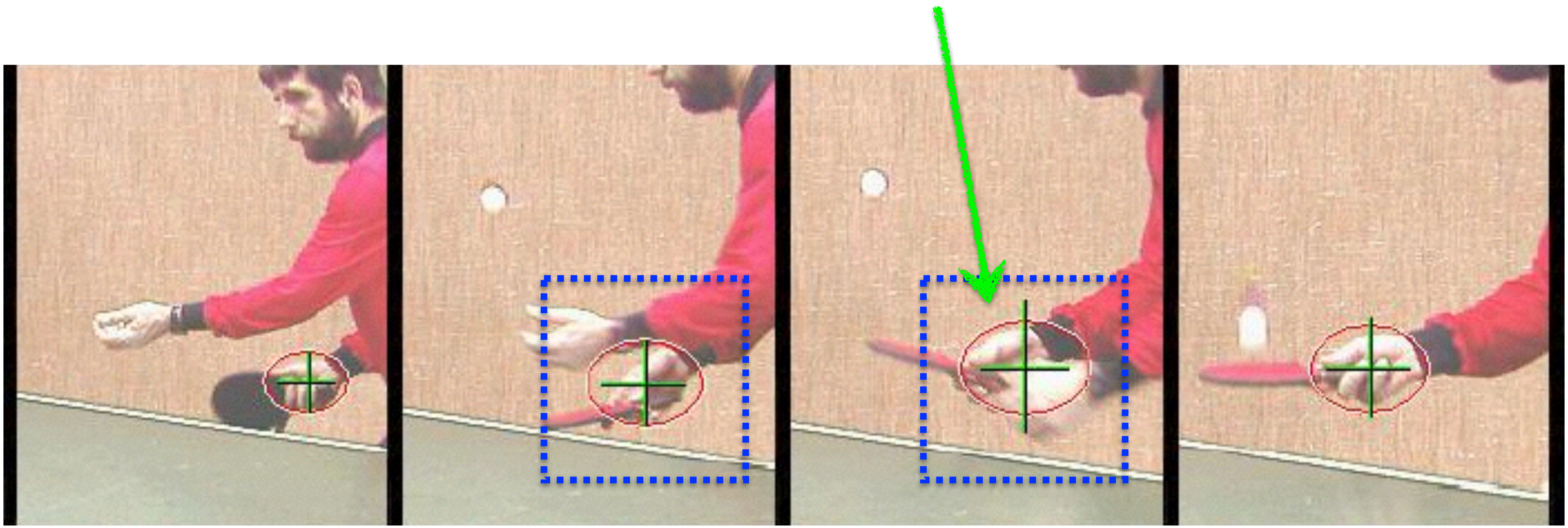


Compute a descriptor for the new target



Target

Search for similar descriptor in neighborhood in next frame



Target Candidate

How do we model the target and candidate regions?

Modeling the target

M-dimensional **target** descriptor

$$\boldsymbol{q} = \{q_1, \ldots, q_M\}$$

(centered at target center)

a ‘fancy’ (confusing) way to write a weighted histogram

$$q_m = C \sum_n k(\|\boldsymbol{x}_n\|^2)\delta[b(\boldsymbol{x}_n) - m]$$

Normalization factor

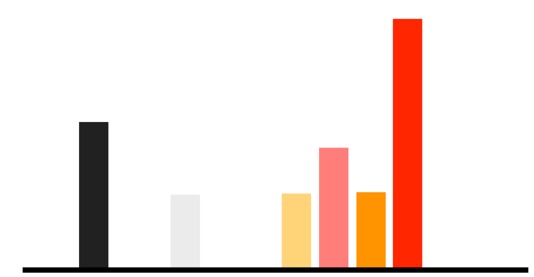
sum over all pixels

function of inverse distance (weight)

Kronecker delta function

quantization function

bin ID



A normalized color histogram (weighted by distance)

Modeling the candidate

M-dimensional **candidate** descriptor

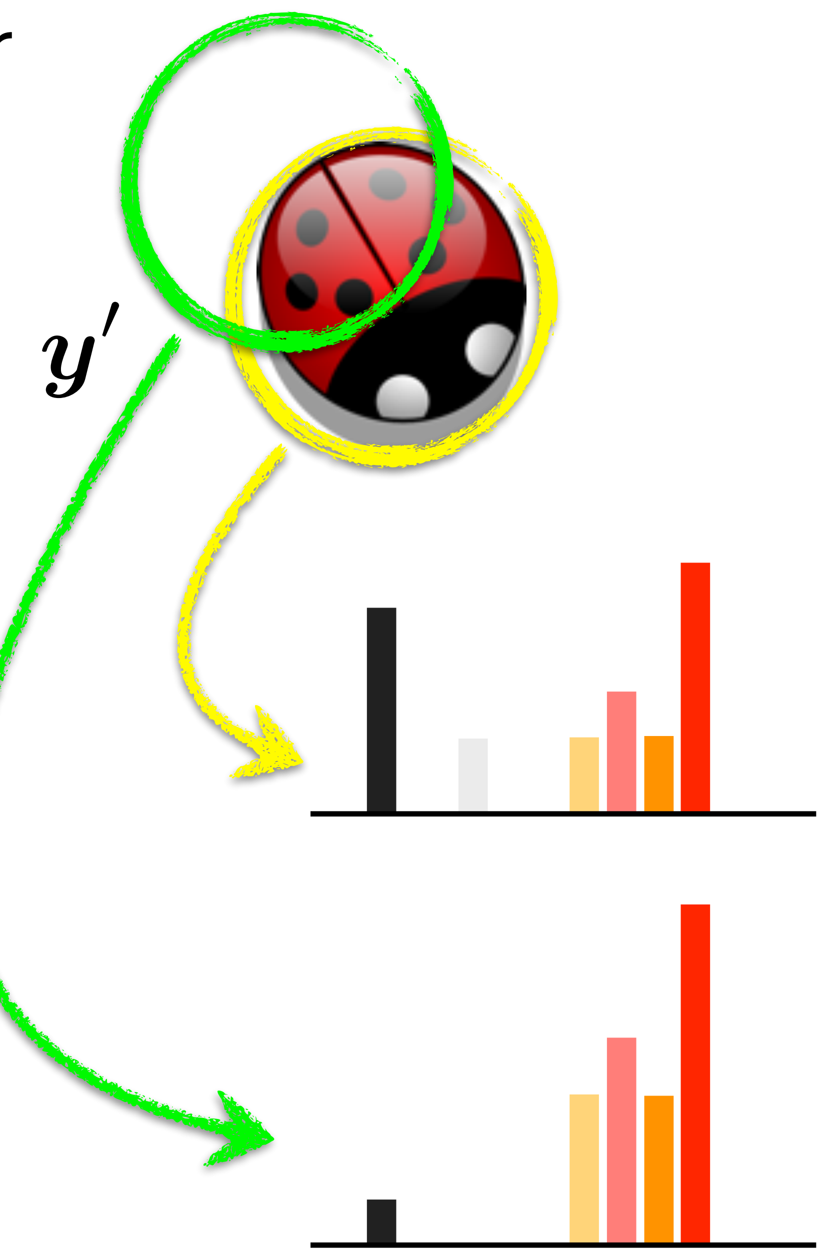
$$\boldsymbol{p}(\boldsymbol{y}) = \{p_1(\boldsymbol{y}), \ldots, p_M(\boldsymbol{y})\}$$

(centered at location **y**)

a weighted histogram at y

$$p_m = C_h \sum_n k\left(\left\|\frac{\boldsymbol{y}-\boldsymbol{x}_n}{h}\right\|^2\right)\delta[b(\boldsymbol{x}_n)-m]$$

bandwidth



Similarity between the target and candidate

Distance function

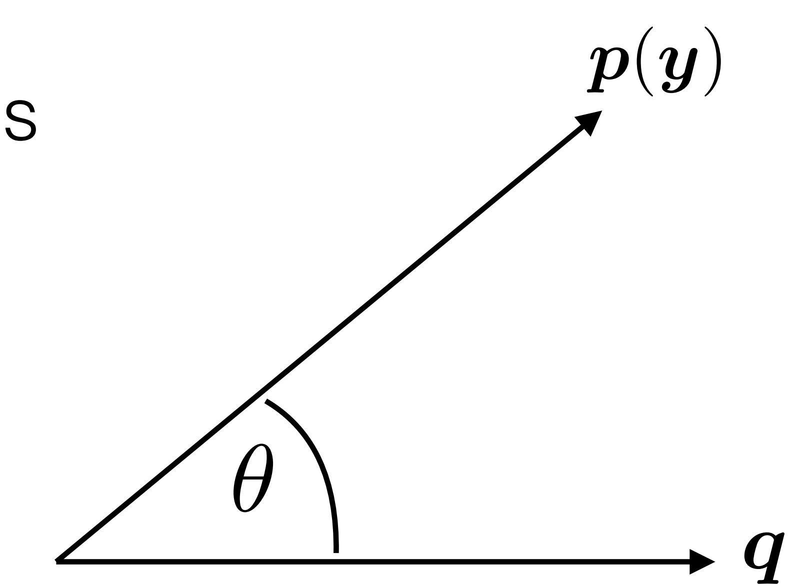
$$d(\boldsymbol{y}) = \sqrt{1 - \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]}$$

Bhattacharyya Coefficient

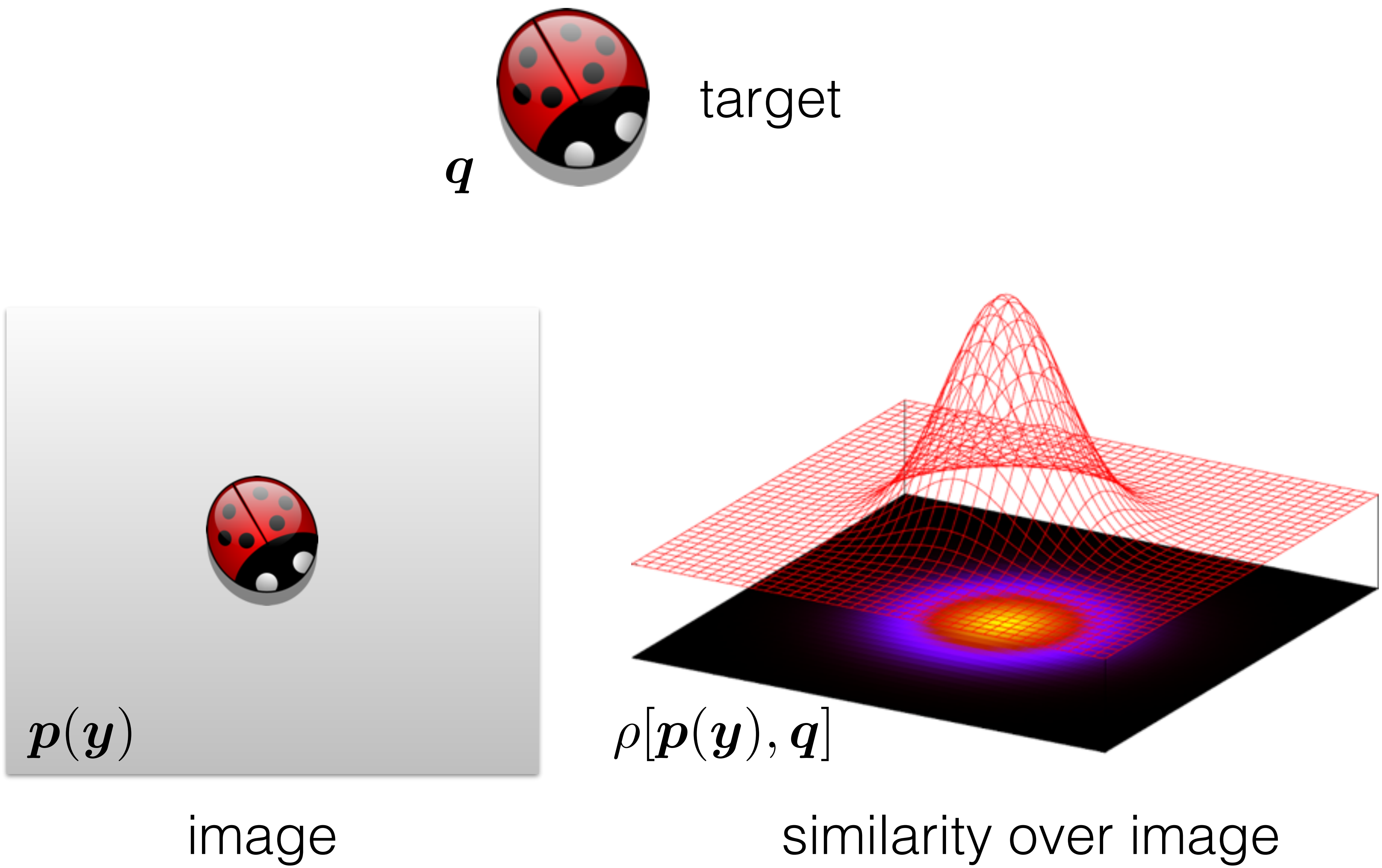
$$\rho(y) \equiv \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] = \sum_m \sqrt{p_m(\boldsymbol{y}) q_u}$$

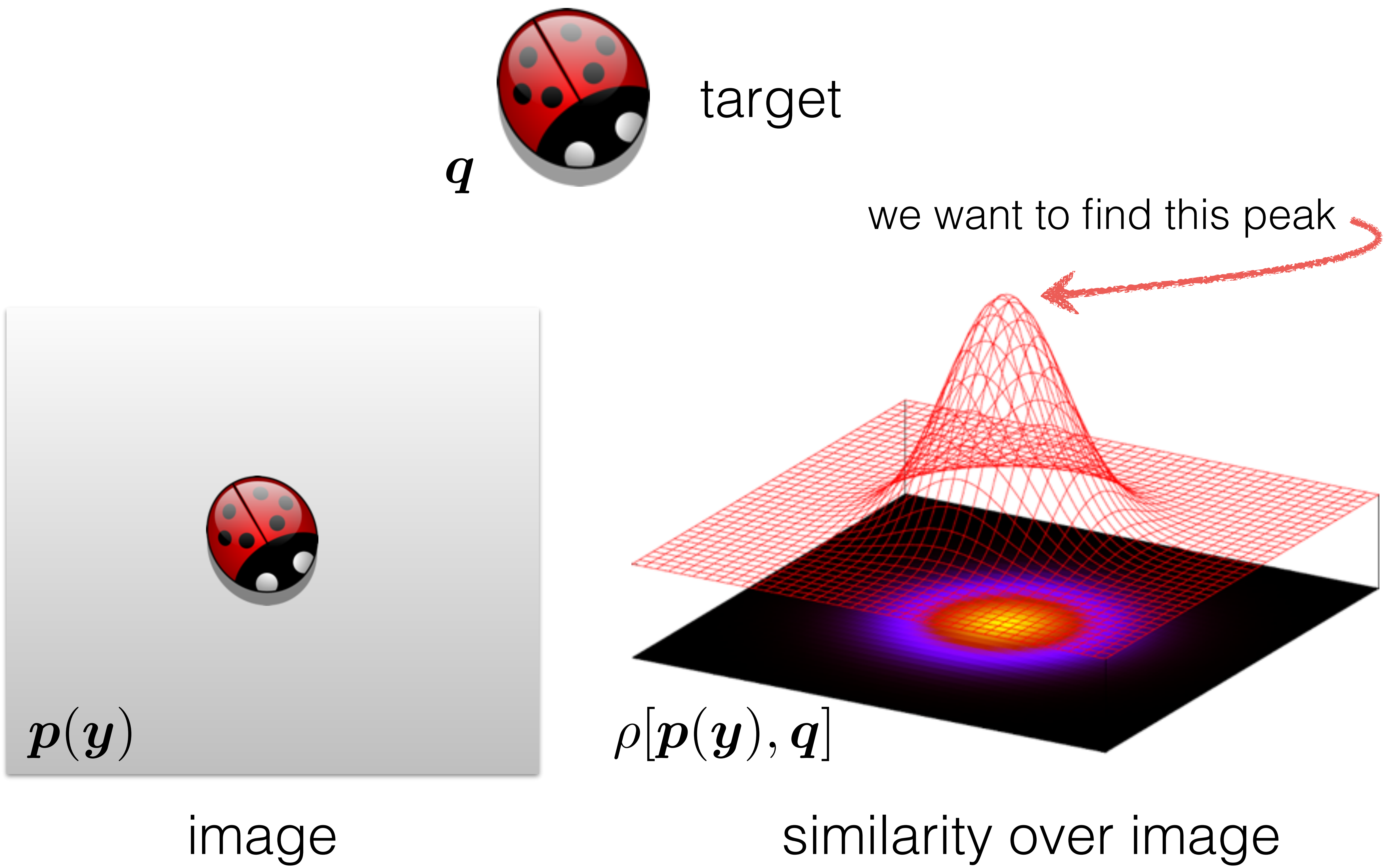
Just the Cosine distance between two unit vectors

$$\rho(\boldsymbol{y}) = \cos\theta_{\boldsymbol{y}} = \frac{\boldsymbol{p}(\boldsymbol{y})^\top \boldsymbol{q}}{\|\boldsymbol{p}\| \|\boldsymbol{q}\|} = \sum_m \sqrt{p_m(\boldsymbol{y}) q_m}$$



Now we can compute the similarity between a target and multiple candidate regions


$\boldsymbol{q}$
target
$\boldsymbol{p}(\boldsymbol{y})$
$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$
image
similarity over image



q
target
we want to find this peak
$p(y)$
image
$\rho[p(y), q]$
similarity over image

Objective function

$$\min_{\boldsymbol{y}} d(\boldsymbol{y})$$ same as $$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

Assuming a good initial guess

$$\rho[\boldsymbol{p}(\boldsymbol{y}_0 + \boldsymbol{y}), \boldsymbol{q}]$$

Linearize around the initial guess (Taylor series expansion)

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \underbrace{\frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m}}_{\text{function at specified value}} + \frac{1}{2} \sum_m p_m(\boldsymbol{y}) \underbrace{\sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}}_{\text{derivative}}$$

Linearized objective

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m} + \frac{1}{2} \sum_m p_m(\boldsymbol{y}) \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}$$

$$p_m = C_h \sum_n k \left(\left\| \frac{\boldsymbol{y} - \boldsymbol{x}_n}{h} \right\|^2 \right) \delta[b(\boldsymbol{x}_n) - m]$$

Remember definition of this?

Fully expanded

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m} + \frac{1}{2} \sum_m \left\{ C_h \sum_n k \left(\left\| \frac{\boldsymbol{y} - \boldsymbol{x}_n}{h} \right\|^2 \right) \delta[b(\boldsymbol{x}_n) - m] \right\} \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}$$

Fully expanded linearized objective

$$\rho[\boldsymbol{p}(\boldsymbol{y}),\boldsymbol{q}] \approx \frac{1}{2}\sum_m \sqrt{p_m(\boldsymbol{y}_0)q_m} + \frac{1}{2}\sum_m \left\{ C_h \sum_n k\left(\left\|\frac{\boldsymbol{y}-\boldsymbol{x}_n}{h}\right\|^2\right)\delta[b(\boldsymbol{x}_n)-m]\right\}\sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}$$

Moving terms around…

$$\rho[\boldsymbol{p}(\boldsymbol{y}),\boldsymbol{q}] \approx \boxed{\frac{1}{2}\sum_m \sqrt{p_m(\boldsymbol{y}_0)q_m}} + \boxed{\frac{C_h}{2}\sum_n w_n k\left(\left\|\frac{\boldsymbol{y}-\boldsymbol{x}_n}{h}\right\|^2\right)}$$

Does not depend on unknown **y**

Weighted kernel density estimate

where $$w_n = \sum_m \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}}\delta[b(\boldsymbol{x}_n)-m]$$

Weight is bigger when $q_m > p_m(\boldsymbol{y}_0)$

OK, why are we doing all this math?

We want to maximize this

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

We want to maximize this

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

Fully expanded linearized objective

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m} + \frac{C_h}{2} \sum_n w_n k \left(\left\| \frac{\boldsymbol{y} - \boldsymbol{x}_n}{h} \right\|^2 \right)$$

where $\quad w_n = \sum_m \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}} \delta[b(\boldsymbol{x}_n) - m]$

We want to maximize this

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

Fully expanded linearized objective

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \cancel{\frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m}} + \frac{C_h}{2} \sum_n w_n k \left(\left\| \frac{\boldsymbol{y} - \boldsymbol{x}_n}{h} \right\|^2 \right)$$

doesn't depend on unknown **y**

where $$w_n = \sum_m \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}} \delta[b(\boldsymbol{x}_n) - m]$$

We want to maximize this

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

only need to maximize this!

Fully expanded linearized objective

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \cancel{\frac{1}{2} \sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m}} + \frac{C_h}{2} \sum_n w_n k \left(\left\| \frac{\boldsymbol{y} - \boldsymbol{x}_n}{h} \right\|^2 \right)$$

doesn't depend on unknown **y**

where $\quad w_n = \sum_m \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}} \delta[b(\boldsymbol{x}_n) - m]$

We want to maximize this

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

Fully expanded linearized objective

$$\rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}] \approx \cancel{\frac{1}{2}\sum_m \sqrt{p_m(\boldsymbol{y}_0) q_m}} + \frac{C_h}{2} \sum_n w_n k \left(\left\| \frac{\boldsymbol{y} - \boldsymbol{x}_n}{h} \right\|^2 \right)$$

doesn't depend on unknown **y**

where $\quad w_n = \sum_m \sqrt{\frac{q_m}{p_m(\boldsymbol{y}_0)}} \delta[b(\boldsymbol{x}_n) - m]$

what can we use to solve this weighted KDE?

Mean Shift Algorithm!

$$\frac{C_h}{2}\sum_n w_n k\left(\left\|\frac{\boldsymbol{y}-\boldsymbol{x}_n}{h}\right\|^2\right)$$

the new sample of mean of this KDE is

$$\boldsymbol{y}_1 = \frac{\sum_n \boldsymbol{x}_n w_n g\left(\left\|\frac{\boldsymbol{y}_0-\boldsymbol{x}_n}{h}\right\|^2\right)}{\sum_n w_n g\left(\left\|\frac{\boldsymbol{y}_0-\boldsymbol{x}_n}{h}\right\|^2\right)}$$

(new candidate location)

(this was derived earlier)

Mean-Shift Object Tracking

For each frame:

1. Initialize location $\boldsymbol{y}_0$
   Compute $\boldsymbol{q}$
   Compute $\boldsymbol{p}(\boldsymbol{y}_0)$

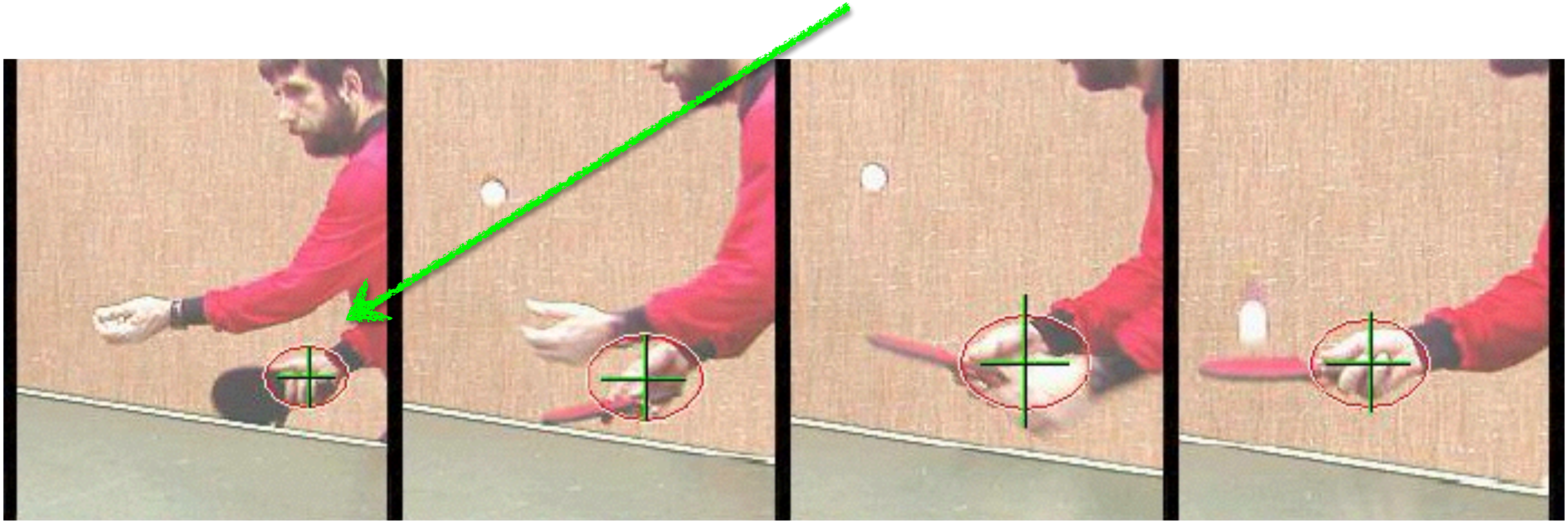
2. Derive weights w_n

3. Shift to new candidate location (mean shift) $\boldsymbol{y}_1$

4. Compute $\boldsymbol{p}(\boldsymbol{y}_1)$

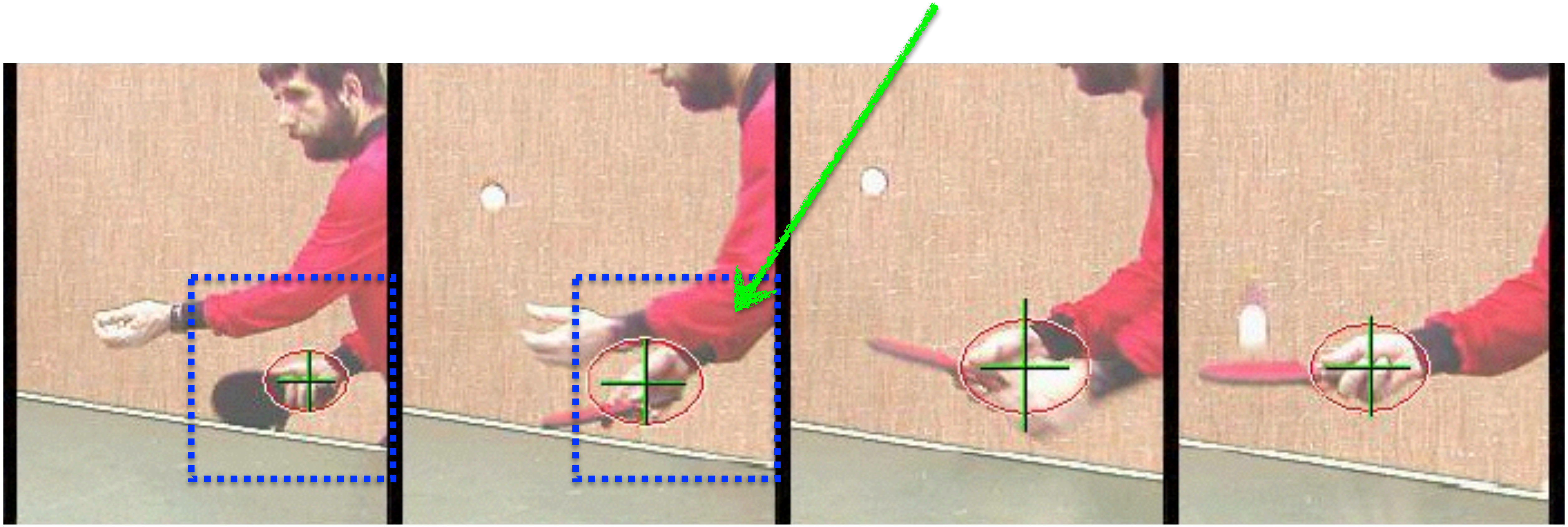
5. If $\|\boldsymbol{y}_0 - \boldsymbol{y}_1\| < \epsilon$ return
   Otherwise $\boldsymbol{y}_0 \leftarrow \boldsymbol{y}_1$ and go back to 2

Compute a descriptor for the target



Target
$\boldsymbol{q}$

Search for similar descriptor in neighborhood in next frame

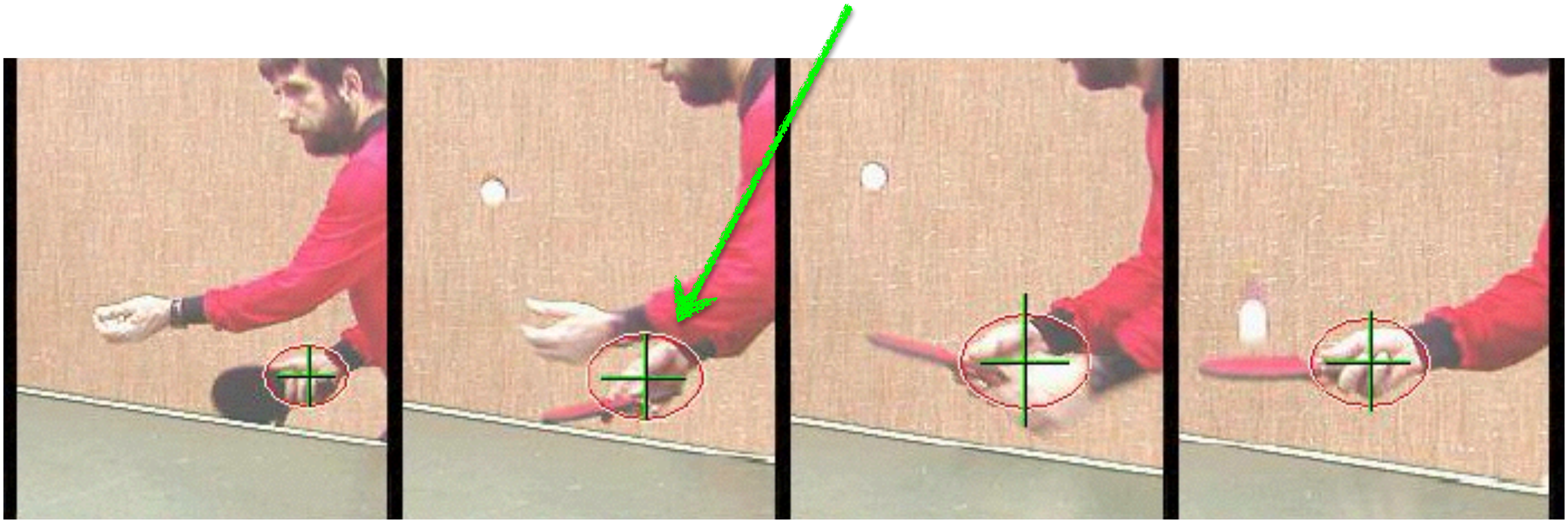


Target

Candidate

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$

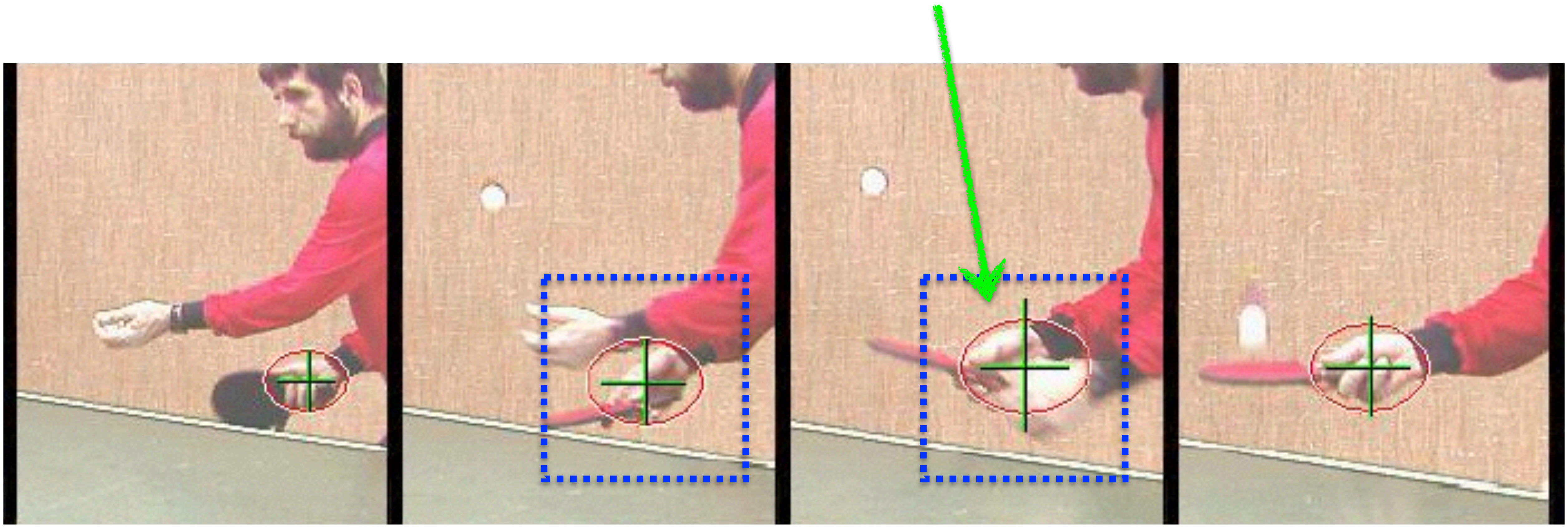
Compute a descriptor for the new target



Target

$\boldsymbol{q}$

Search for similar descriptor in neighborhood in next frame



Target

Candidate

$$\max_{\boldsymbol{y}} \rho[\boldsymbol{p}(\boldsymbol{y}), \boldsymbol{q}]$$