Previously, we’ve assumed that the forms of the underlying densities were of some particular known parametric form.

But, what if this is not the case?

Indeed, for most real-world pattern recognition scenarios this assumption is suspect.

For example, most real-world entities have multimodal distributions whereas all classical parametric densities are unimodal.
Previously, we’ve assumed that the forms of the underlying densities were of some particular known parametric form.

But, what if this is not the case?

Indeed, for most real-world pattern recognition scenarios this assumption is suspect.

For example, most real-world entities have multimodal distributions whereas all classical parametric densities are unimodal.

We will examine nonparametric procedures that can be used with arbitrary distributions and without the assumption that the underlying form of the densities are known.

- Histograms.
- Kernel Density Estimation / Parzen Windows.
- k-Nearest Neighbor Density Estimation.
- Real Example in Figure-Ground Segmentation
Histograms

$p(X, Y)

Y = 2

Y = 1

X
Histograms

$p(X, Y)$

$Y = 2$

$Y = 1$

$p(X)$

$X$
Histograms

$p(X, Y)$

$Y = 2$

$Y = 1$

$p(X)$

$p(Y)$

$p(X | Y = 1)$
Histograms

Histograms

$p(X, Y)$

$X$

$p(Y)$

$X$

$p(X)$

$p(X|Y = 1)$

$Y = 2$

$Y = 1$

$X$

$p(Y)$

$X$

$p(X|Y = 1)$

$X$
Consider a single continuous variable $x$ and let’s say we have a set $\mathcal{D}$ of $N$ of them $\{x_1, \ldots, x_N\}$. Our goal is to model $p(x)$ from $\mathcal{D}$.

Standard histograms simply partition $x$ into distinct bins of width $\Delta_i$ and then count the number $n_i$ of observations $x$ falling into bin $i$. To turn this count into a normalized probability density, we simply divide by the total number of observations $N$ and by the width $\Delta_i$ of the bins. This gives us:

$$p_i = \frac{n_i}{N \Delta_i}$$

Hence the model for the density $p(x)$ is constant over the width of each bin. (And often the bins are chosen to have the same width $\Delta$.)
Consider a single continuous variable $x$ and let’s say we have a set $\mathcal{D}$ of $N$ of them $\{x_1, \ldots, x_N\}$. Our goal is to model $p(x)$ from $\mathcal{D}$.

Standard histograms simply partition $x$ into distinct bins of width $\Delta_i$ and then count the number $n_i$ of observations $x$ falling into bin $i$. To turn this count into a normalized probability density, we simply divide by the total number of observations $N$ and by the width $\Delta_i$ of the bins. This gives us:

$$ p_i = \frac{n_i}{N \Delta_i} \quad (1) $$

Hence the model for the density $p(x)$ is constant over the width of each bin. (And often the bins are chosen to have the same width $\Delta_i = \Delta$.)
Consider a single continuous variable \( x \) and let’s say we have a set \( D \) of \( N \) of them \( \{x_1, \ldots, x_N\} \). Our goal is to model \( p(x) \) from \( D \).

Standard histograms simply partition \( x \) into distinct bins of width \( \Delta_i \) and then count the number \( n_i \) of observations \( x \) falling into bin \( i \).

To turn this count into a normalized probability density, we simply divide by the total number of observations \( N \) and by the width \( \Delta_i \) of the bins.

This gives us:

\[
p_i = \frac{n_i}{N\Delta_i} \tag{1}
\]
Consider a single continuous variable $x$ and let’s say we have a set $D$ of $N$ of them $\{x_1, \ldots, x_N\}$. Our goal is to model $p(x)$ from $D$.

Standard histograms simply partition $x$ into distinct bins of width $\Delta_i$ and then count the number $n_i$ of observations $x$ falling into bin $i$.

To turn this count into a normalized probability density, we simply divide by the total number of observations $N$ and by the width $\Delta_i$ of the bins.

This gives us:

$$p_i = \frac{n_i}{N\Delta_i} \tag{1}$$

Hence the model for the density $p(x)$ is constant over the width of each bin. (And often the bins are chosen to have the same width $\Delta_i = \Delta$.)
<table>
<thead>
<tr>
<th>Bin Number</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bin Count</td>
<td>3</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>
Histogram Density as a Function of Bin Width

\[ \Delta = 0.04 \]

\[ \Delta = 0.08 \]

\[ \Delta = 0.25 \]
The green curve is the underlying true density from which the samples were drawn. It is a mixture of two Gaussians.
The green curve is the underlying true density from which the samples were drawn. It is a mixture of two Gaussians.

When $\Delta$ is very small (top), the resulting density is quite spiky and hallucinates a lot of structure not present in $p(x)$.

When $\Delta$ is very big (bottom), the resulting density is quite smooth and consequently fails to capture the bimodality of $p(x)$.

In principle, a histogram density model is also dependent on the choice of the edge location of each bin.
The green curve is the underlying true density from which the samples were drawn. It is a mixture of two Gaussians.

When $\Delta$ is very small (top), the resulting density is quite spiky and hallucinates a lot of structure not present in $p(x)$.

When $\Delta$ is very big (bottom), the resulting density is quite smooth and consequently fails to capture the bimodality of $p(x)$. 
The green curve is the underlying true density from which the samples were drawn. It is a mixture of two Gaussians. When $\Delta$ is very small (top), the resulting density is quite spiky and hallucinates a lot of structure not present in $p(x)$. When $\Delta$ is very big (bottom), the resulting density is quite smooth and consequently fails to capture the bimodality of $p(x)$. It appears that the best results are obtained for some intermediate value of $\Delta$, which is given in the middle figure.
The green curve is the underlying true density from which the samples were drawn. It is a mixture of two Gaussians.

When $\Delta$ is very small (top), the resulting density is quite spiky and hallucinates a lot of structure not present in $p(x)$.

When $\Delta$ is very big (bottom), the resulting density is quite smooth and consequently fails to capture the bimodality of $p(x)$.

It appears that the best results are obtained for some intermediate value of $\Delta$, which is given in the middle figure.

In principle, a histogram density model is also dependent on the choice of the edge location of each bin.
What are the advantages and disadvantages of the histogram density estimator?

Advantages:
- Simple to evaluate and simple to use.
- One can throw away \( D \) once the histogram is computed.
- Can be computed sequentially if data continues to come in.

Disadvantages:
- The estimated density has discontinuities due to the bin edges rather than any property of the underlying density.
- Scales poorly (curse of dimensionality): we would have \( M^D \) bins if we divided each variable in a \( D \)-dimensional space into \( M \) bins.
What are the advantages and disadvantages of the histogram density estimator?

**Advantages:**
- Simple to evaluate and simple to use.
- One can throw away $D$ once the histogram is computed.
- Can be computed sequentially if data continues to come in.

**Disadvantages:**
- The estimated density has discontinuities due to the bin edges rather than any property of the underlying density.
- Scales poorly (curse of dimensionality): we would have $M^D$ bins if we divided each variable in a $D$-dimensional space into $M$ bins.
What are the advantages and disadvantages of the histogram density estimator?

**Advantages:**
- Simple to evaluate and simple to use.
- One can throw away $D$ once the histogram is computed.
- Can be computed sequentially if data continues to come in.

**Disadvantages:**
- The estimated density has discontinuities due to the bin edges rather than any property of the underlying density.
- Scales poorly (curse of dimensionality): we would have $M^D$ bins if we divided each variable in a $D$-dimensional space into $M$ bins.
What can we learn from Histogram Density Estimation?

Lesson 1: To estimate the probability density at a particular location, we should consider the data points that lie within some local neighborhood of that point.

- This requires we define some distance measure.
- There is a natural smoothness parameter describing the spatial extent of the regions (this was the bin width for the histograms).
What can we learn from Histogram Density Estimation?

- **Lesson 1**: To estimate the probability density at a particular location, we should consider the data points that lie within some local neighborhood of that point.
  - This requires we define some distance measure.
  - There is a natural smoothness parameter describing the spatial extent of the regions (this was the bin width for the histograms).

- **Lesson 2**: The value of the smoothing parameter should neither be too large or too small in order to obtain good results.
What can we learn from Histogram Density Estimation?

- **Lesson 1:** To estimate the probability density at a particular location, we should consider the data points that lie within some local neighborhood of that point.
  - This requires we define some distance measure.
  - There is a natural smoothness parameter describing the spatial extent of the regions (this was the bin width for the histograms).

- **Lesson 2:** The value of the smoothing parameter should neither be too large or too small in order to obtain good results.

With these two lessons in mind, we proceed to kernel density estimation and nearest neighbor density estimation, two closely related methods for density estimation.
Consider again samples $x$ from underlying density $p(x)$. Let $R$ denote a small region containing $x$. The probability mass associated with $R$ is given by

$$P = \int_R p(x') \, dx'$$

Suppose we have $n$ samples $x \in D$. The probability of each sample falling into $R$ is $P$. How will the total number of $k$ points falling into $R$ be distributed? This will be a binomial distribution:

$$P(k) = \binom{n}{k} P^k (1 - P)^{n-k}$$
Consider again samples \( x \) from underlying density \( p(x) \).

Let \( \mathcal{R} \) denote a small region containing \( x \).

The probability mass associated with \( \mathcal{R} \) is given by

\[
P = \int_{\mathcal{R}} p(x') \, dx'
\]  

(2)
Consider again samples $x$ from underlying density $p(x)$. Let $\mathcal{R}$ denote a small region containing $x$. The probability mass associated with $\mathcal{R}$ is given by

$$P = \int_{\mathcal{R}} p(x') dx'$$ (2)

Suppose we have $n$ samples $x \in \mathcal{D}$. The probability of each sample falling into $\mathcal{R}$ is $P$.

How will the total number of $k$ points falling into $\mathcal{R}$ be distributed?
The Space-Averaged / Smoothed Density

- Consider again samples $x$ from underlying density $p(x)$.
- Let $\mathcal{R}$ denote a small region containing $x$.
- The probability mass associated with $\mathcal{R}$ is given by

$$ P = \int_{\mathcal{R}} p(x') dx' $$

(2)

- Suppose we have $n$ samples $x \in \mathcal{D}$. The probability of each sample falling into $\mathcal{R}$ is $P$.
- How will the total number of $k$ points falling into $\mathcal{R}$ be distributed?
- This will be a **binomial distribution**:

$$ P_k = \binom{n}{k} P^k (1 - P)^{n-k} $$

(3)
The expected value for $k$ is thus

$$E[k] = nP$$

(4)
The Space-Averaged / Smoothed Density

- The expected value for $k$ is thus

$$E[k] = nP$$  \hspace{1cm} (4)

- The binomial for $k$ peaks very sharply about the mean. So, we expect $k/n$ to be a very good estimate for the probability $P$ (and thus for the space-averaged density).
The Space-Averaged / Smoothed Density

- The expected value for $k$ is thus
  \[ E[k] = nP \]  

  (4)

- The binomial for $k$ peaks very sharply about the mean. So, we expect $k/n$ to be a very good estimate for the probability $P$ (and thus for the space-averaged density).

- This estimate is increasingly accurate as $n$ increases.

FIGURE 4.1. The relative probability an estimate given by Eq. 4 will yield a particular value for the probability density, here where the true probability was chosen to be 0.7. Each curve is labeled by the total number of patterns $n$ sampled, and is scaled to give the same maximum (at the true probability). The form of each curve is binomial, as given by Eq. 2. For large $n$, such binomials peak strongly at the true probability. In the limit $n \to \infty$, the curve approaches a delta function, and we are guaranteed that our estimate will give the true probability. From: Richard O. Duda, Peter E. Hart, and David G. Stork, Pattern Classification. Copyright © 2001 by John Wiley & Sons, Inc.
The Space-Averaged / Smoothed Density

Assuming continuous $p(x)$ and that $\mathcal{R}$ is so small that $p(x)$ does not appreciably vary within it, we can write:

$$\int_{\mathcal{R}} p(x')dx' \approx p(x)V \tag{5}$$

where $x$ is a point within $\mathcal{R}$ and $V$ is the volume enclosed by $\mathcal{R}$.
The Space-Averaged / Smoothed Density

- Assuming continuous \( p(x) \) and that \( \mathcal{R} \) is so small that \( p(x) \) does not appreciably vary within it, we can write:

\[
\int_{\mathcal{R}} p(x') \, dx' \approx p(x) V \tag{5}
\]

where \( x \) is a point within \( \mathcal{R} \) and \( V \) is the volume enclosed by \( \mathcal{R} \).

- After some rearranging, we get the following estimate for \( p(x) \)

\[
p(x) \approx \frac{k}{nV} \tag{6}
\]
Example

- Simulated an example of the density at 0.5 for an underlying zero-mean, unit variance Gaussian.
- Varied the volume used to estimate the density.
- Red=1000, Green=2000, Blue=3000, Yellow=4000, Black=5000.
The validity of our estimate depends on two contradictory assumptions:

1. The region $\mathcal{R}$ must be sufficiently small so that the density is approximately constant over the region.
2. The region $\mathcal{R}$ must be sufficiently large so that the number $k$ of points falling inside it is sufficient to yield a sharply peaked binomial.
Practical Concerns

- The validity of our estimate depends on two contradictory assumptions:
  1. The region $\mathcal{R}$ must be sufficiently small the the density is approximately constant over the region.
  2. The region $\mathcal{R}$ must be sufficiently large that the number $k$ of points falling inside it is sufficient to yield a sharply peaked binomial.

- Another way of looking it is to fix the volume $V$ and increase the number of training samples. Then, the ratio $k/n$ will converge as desired. But, this will only yield an estimate of the space-averaged density ($P/V$).
Practical Concerns

The validity of our estimate depends on two contradictory assumptions:

1. The region $\mathcal{R}$ must be sufficiently small so that the density is approximately constant over the region.
2. The region $\mathcal{R}$ must be sufficiently large that the number $k$ of points falling inside it is sufficient to yield a sharply peaked binomial.

Another way of looking at it is to fix the volume $V$ and increase the number of training samples. Then, the ratio $k/n$ will converge as desired. But, this will only yield an estimate of the space-averaged density ($P/V$).

We want $p(x)$, so we need to let $V$ approach 0. However, with a fixed $n$, $\mathcal{R}$ will become so small, that no points will fall into it and our estimate would be useless: $p(x) \approx 0$. 
Practical Concerns

- The validity of our estimate depends on two contradictory assumptions:
  1. The region $\mathcal{R}$ must be sufficiently small the the density is approximately constant over the region.
  2. The region $\mathcal{R}$ must be sufficiently large that the number $k$ of points falling inside it is sufficient to yield a sharply peaked binomial.

- Another way of looking it is to fix the volume $V$ and increase the number of training samples. Then, the ratio $k/n$ will converge as desired. But, this will only yield an estimate of the space-averaged density ($P/V$).

- We want $p(x)$, so we need to let $V$ approach 0. However, with a fixed $n$, $\mathcal{R}$ will become so small, that no points will fall into it and our estimate would be useless: $p(x) \approx 0$.

- Note that in practice, we cannot let $V$ to become arbitrarily small because the number of samples is always limited.
How can we skirt these limitations when an unlimited number of samples if available?

- To estimate the density at $x$, form a sequence of regions $R_1, R_2, \ldots$ containing $x$ with the $R_1$ having 1 sample, $R_2$ having 2 samples and so on.
How can we skirt these limitations when an unlimited number of samples if available?

- To estimate the density at $x$, form a sequence of regions $\mathcal{R}_1, \mathcal{R}_2, \ldots$ containing $x$ with the $\mathcal{R}_1$ having 1 sample, $\mathcal{R}_2$ having 2 samples and so on.

- Let $V_n$ be the volume of $\mathcal{R}_n$, $k_n$ be the number of samples falling in $\mathcal{R}_n$, and $p_n(x)$ be the $n$th estimate for $p(x)$:

$$p_n(x) = \frac{k_n}{nV_n} \quad (7)$$
How can we skirt these limitations when an unlimited number of samples if available?

- To estimate the density at \( x \), form a sequence of regions \( \mathcal{R}_1, \mathcal{R}_2, \ldots \) containing \( x \) with the \( \mathcal{R}_1 \) having 1 sample, \( \mathcal{R}_2 \) having 2 samples and so on.

- Let \( V_n \) be the volume of \( \mathcal{R}_n \), \( k_n \) be the number of samples falling in \( \mathcal{R}_n \), and \( p_n(x) \) be the \( n \)th estimate for \( p(x) \):

\[
p_n(x) = \frac{k_n}{nV_n}
\]

- If \( p_n(x) \) is to converge to \( p(x) \) we need the following three conditions

\[
\lim_{n \to \infty} V_n = 0 \quad (8)
\]
\[
\lim_{n \to \infty} k_n = \infty \quad (9)
\]
\[
\lim_{n \to \infty} \frac{k_n}{n} = 0 \quad (10)
\]
\( \lim_{n \to \infty} V_n = 0 \) ensures that our space-averaged density will converge to \( p(x) \).
\lim_{n \to \infty} V_n = 0 \text{ ensures that our space-averaged density will converge to } p(x).

\lim_{n \to \infty} k_n = \infty \text{ basically ensures that the frequency ratio will converge to the probability } P \text{ (the binomial will be sufficiently peaked).}
- $\lim_{n \to \infty} V_n = 0$ ensures that our space-averaged density will converge to $p(x)$.
- $\lim_{n \to \infty} k_n = \infty$ basically ensures that the frequency ratio will converge to the probability $P$ (the binomial will be sufficiently peaked).
- $\lim_{n \to \infty} k_n / n = 0$ is required for $p_n(x)$ to converge at all. It also says that although a huge number of samples will fall within the region $\mathcal{R}_n$, they will form a negligibly small fraction of the total number of samples.
- \( \lim_{n \to \infty} V_n = 0 \) ensures that our space-averaged density will converge to \( p(x) \).

- \( \lim_{n \to \infty} k_n = \infty \) basically ensures that the frequency ratio will converge to the probability \( P \) (the binomial will be sufficiently peaked).

- \( \lim_{n \to \infty} k_n/n = 0 \) is required for \( p_n(x) \) to converge at all. It also says that although a huge number of samples will fall within the region \( \mathcal{R}_n \), they will form a negligibly small fraction of the total number of samples.

- There are two common ways of obtaining regions that satisfy these conditions:
• \( \lim_{n \to \infty} V_n = 0 \) ensures that our space-averaged density will converge to \( p(x) \).

• \( \lim_{n \to \infty} k_n = \infty \) basically ensures that the frequency ratio will converge to the probability \( P \) (the binomial will be sufficiently peaked).

• \( \lim_{n \to \infty} k_n/n = 0 \) is required for \( p_n(x) \) to converge at all. It also says that although a huge number of samples will fall within the region \( R_n \), they will form a negligibly small fraction of the total number of samples.

• There are two common ways of obtaining regions that satisfy these conditions:
  
  1. Shrink an initial region by specifying the volume \( V_n \) as some function of \( n \) such as \( V_n = 1/\sqrt{n} \). Then, we need to show that \( p_n(x) \) converges to \( p(x) \). (This is like the Parzen window we’ll talk about next.)
- \( \lim_{n \to \infty} V_n = 0 \) ensures that our space-averaged density will converge to \( p(x) \).
- \( \lim_{n \to \infty} k_n = \infty \) basically ensures that the frequency ratio will converge to the probability \( P \) (the binomial will be sufficiently peaked).
- \( \lim_{n \to \infty} k_n/n = 0 \) is required for \( p_n(x) \) to converge at all. It also says that although a huge number of samples will fall within the region \( R_n \), they will form a negligibly small fraction of the total number of samples.

There are two common ways of obtaining regions that satisfy these conditions:

1. Shrink an initial region by specifying the volume \( V_n \) as some function of \( n \) such as \( V_n = 1/\sqrt{n} \). Then, we need to show that \( p_n(x) \) converges to \( p(x) \). (This is like the Parzen window we’ll talk about next.)
2. Specify \( k_n \) as some function of \( n \) such as \( k_n = \sqrt{n} \). Then, we grow the volume \( V_n \) until it encloses \( k_n \) neighbors of \( x \). (This is the k-nearest-neighbor).
\begin{itemize}
  \item \(\lim_{n \to \infty} V_n = 0\) ensures that our space-averaged density will converge to \(p(x)\).
  \item \(\lim_{n \to \infty} k_n = \infty\) basically ensures that the frequency ratio will converge to the probability \(P\) (the binomial will be sufficiently peaked).
  \item \(\lim_{n \to \infty} k_n / n = 0\) is required for \(p_n(x)\) to converge at all. It also says that although a huge number of samples will fall within the region \(R_n\), they will form a negligibly small fraction of the total number of samples.
  \item There are two common ways of obtaining regions that satisfy these conditions:
    \begin{enumerate}
    \item Shrink an initial region by specifying the volume \(V_n\) as some function of \(n\) such as \(V_n = 1/\sqrt{n}\). Then, we need to show that \(p_n(x)\) converges to \(p(x)\). (This is like the Parzen window we’ll talk about next.)
    \item Specify \(k_n\) as some function of \(n\) such as \(k_n = \sqrt{n}\). Then, we grow the volume \(V_n\) until it encloses \(k_n\) neighbors of \(x\). (This is the k-nearest-neighbor).
    \end{enumerate}
\end{itemize}
FIGURE 4.2. There are two leading methods for estimating the density at a point, here at the center of each square. The one shown in the top row is to start with a large volume centered on the test point and shrink it according to a function such as $V_n = 1/\sqrt{n}$. The other method, shown in the bottom row, is to decrease the volume in a data-dependent way, for instance letting the volume enclose some number $k_n = \sqrt{n}$ of sample points.

The sequences in both cases represent random variables that generally converge and allow the true density at the test point to be calculated. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.
Let’s temporarily assume the region $R$ is a $d$-dimensional hypercube with $h_n$ being the length of an edge.
Parzen Windows

Let’s temporarily assume the region $\mathcal{R}$ is a $d$-dimensional hypercube with $h_n$ being the length of an edge.

The volume of the hypercube is given by

$$V_n = h_n^d.$$  \hspace{1cm} (11)
Parzen Windows

- Let’s temporarily assume the region \( \mathcal{R} \) is a \( d \)-dimensional hypercube with \( h_n \) being the length of an edge.
- The volume of the hypercube is given by
  \[
  V_n = h_n^d.
  \]  
  \( (11) \)
- We can derive an analytic expression for \( k_n \):
  - Define a windowing function:
    \[
    \varphi(u) = \begin{cases} 
    1 & |u_j| \leq 1/2 \\
    0 & \text{otherwise} 
    \end{cases} 
    \]  
    \( j = 1, \ldots, d \)
    \( (12) \)
- This windowing function \( \varphi \) defines a unit hypercube centered at the origin.
- Hence, \( \varphi((x - x_i)/h_n) \) is equal to unity if \( x_i \) falls within the hypercube of volume \( V_n \) centered at \( x \), and is zero otherwise.
The number of samples in this hypercube is therefore given by

\[ k_n = \sum_{i=1}^{n} \varphi \left( \frac{x - x_i}{h_n} \right). \]  

(13)
The number of samples in this hypercube is therefore given by

$$k_n = \sum_{i=1}^{n} \varphi \left( \frac{x - x_i}{h_n} \right).$$  

(13)

Substituting in equation (7), $p_n(x) = k_n/(nV_n)$ yields the estimate

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \varphi \left( \frac{x - x_i}{h_n} \right).$$  

(14)
The number of samples in this hypercube is therefore given by

$$k_n = \sum_{i=1}^{n} \varphi \left( \frac{x - x_i}{h_n} \right).$$

(13)

Substituting in equation (7), $p_n(x) = k_n/(nV_n)$ yields the estimate

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \varphi \left( \frac{x - x_i}{h_n} \right).$$

(14)

Hence, the windowing function $\varphi$, in this context called a Parzen window, tells us how to weight all of the samples in $\mathcal{D}$ to determine $p(x)$ at a particular $x$. 
But, what undesirable trait from histograms are inherited by Parzen window density estimates of the form we've just defined? Discontinuities...
Example

- But, what undesirable trait from histograms are inherited by Parzen window density estimates of the form we’ve just defined?
But, what undesirable trait from histograms are inherited by Parzen window density estimates of the form we’ve just defined?

Discontinuities...
Generalizing the Kernel Function

- What if we allow a more general class of windowing functions rather than the hypercube?
- If we think of the windowing function as an interpolator, rather than considering the window function about \( x \) only, we can visualize it as a kernel sitting on each data sample \( x_i \) in \( D \).
Generalizing the Kernel Function

- What if we allow a more general class of windowing functions rather than the hypercube?
- If we think of the windowing function as an interpolator, rather than considering the window function about $x$ only, we can visualize it as a kernel sitting on each data sample $x_i$ in $\mathcal{D}$.
- And, if we require the following two conditions on the kernel function $\varphi$, then we can be assured that the resulting density $p_n(x)$ will be proper: non-negative and integrate to 1.

$$\varphi(x) \geq 0 \quad (15)$$
$$\int \varphi(u) du = 1 \quad (16)$$

- For our previous case of $V_n = h_n^d$, then it follows $p_n(x)$ will also satisfy these conditions.
Example: A Univariate Gaussian Kernel

- A popular choice of the kernel is the Gaussian kernel:
  \[
  \varphi_h(u) = \frac{1}{\sqrt{2\pi}} \exp \left[ -\frac{1}{2} u^2 \right]
  \]  
  (17)

- The resulting density is given by:
  \[
  p(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_n \sqrt{2\pi}} \exp \left[ -\frac{1}{2h_n^2} (x - x_i)^2 \right]
  \]  
  (18)

- It will give us smoother estimates without the discontinuities from the hypercube kernel.
An important question is what effect does the window width $h_n$ have on $p_n(x)$?

Define $\delta_n(x)$ as

$$\delta_n(x) = \frac{1}{V_n} \varphi \left( \frac{x}{h_n} \right)$$  

(19)

and rewrite $p_n(x)$ as the average

$$p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \delta_n(x - x_i)$$

(20)
Effect of the Window Width

Slide II

- $h_n$ clearly affects both the amplitude and the width of $\delta_n(x)$.
Effect of the Window Width

$h_n$ clearly affects both the amplitude and the width of $\delta_n(x)$. 
Effect of Window Width (And, hence, Volume $V_n$)

- But, for any value of $h_n$, the distribution is normalized:

$$\int \delta(x - x_i)dx = \int \frac{1}{V_n} \varphi \left( \frac{x - x_i}{h_n} \right) dx = \int \varphi(u)du = 1 \quad (21)$$
Effect of Window Width (And, hence, Volume $V_n$)

- But, for any value of $h_n$, the distribution is normalized:
\[
\int \delta(x - x_i) dx = \int \frac{1}{V_n} \varphi \left( \frac{x - x_i}{h_n} \right) dx = \int \varphi(u) du = 1 \tag{21}
\]

- If $V_n$ is too large, the estimate will suffer from too little resolution.
Effect of Window Width (And, hence, Volume $V_n$)

But, for any value of $h_n$, the distribution is normalized:

$$
\int \delta(x - x_i) \, dx = \int \frac{1}{V_n} \varphi \left( \frac{x - x_i}{h_n} \right) \, dx = \int \varphi(u) \, du = 1 \quad (21)
$$

- If $V_n$ is too large, the estimate will suffer from too little resolution.
- If $V_n$ is too small, the estimate will suffer from too much variability.
Effect of Window Width (And, hence, Volume $V_n$)

- But, for any value of $h_n$, the distribution is normalized:

$$
\int \delta(x - x_i) \, dx = \int \frac{1}{V_n} \varphi \left( \frac{x - x_i}{h_n} \right) \, dx = \int \varphi(u) \, du = 1 \quad (21)
$$

- If $V_n$ is too large, the estimate will suffer from too little resolution.
- If $V_n$ is too small, the estimate will suffer from too much variability.
- In theory (with an unlimited number of samples), we can let $V_n$ slowly approach zero as $n$ increases and then $p_n(x)$ will converge to the unknown $p(x)$. But, in practice, we can, at best, seek some compromise.
Example: Revisiting the Univariate Gaussian Kernel

![Diagram showing Parzen-window estimates of a univariate normal density using different window widths and numbers of samples.](image-url)

- $h_i = 1$
- $h_i = 0.5$
- $h_i = 0.1$

- $n = 1$
- $n = 10$
- $n = 100$
- $n = \infty$

FIGURE 4.5. Parzen-window estimates of a univariate normal density using different window widths and numbers of samples. The vertical axes have been scaled to best show the structure in each graph. Note particularly that the $n = \infty$ estimates are the same (and match the true density function), regardless of window width. From: Richard O. Duda, Peter E. Hart, and David G. Stork, Pattern Classification. Copyright © 2001 by John Wiley & Sons, Inc.
Example: A Bimodal Distribution

![Graphs showing Parzen-window estimates of a bimodal distribution using different window widths and numbers of samples. Note particularly that the $n=\infty$ estimates are the same (and match the true distribution), regardless of window width. From: Richard O. Duda, Peter E. Hart, and David G. Stork, Pattern Classification. Copyright © 2001 by John Wiley & Sons, Inc.]
Parzen Window-Based Classifiers

- Estimate the densities for each category.
- Classify a query point by the label corresponding to the maximum posterior (i.e., one can include priors).

As you guessed it, the decision regions for a Parzen window-based classifier depend upon the kernel function.
Parzen Window-Based Classifiers

- Estimate the densities for each category.
- Classify a query point by the label corresponding to the maximum posterior (i.e., one can include priors).
- As you guessed it, the decision regions for a Parzen window-based classifier depend upon the kernel function.
During training, we can make the error arbitrarily low by making the window sufficiently small, but this will have an ill-effect during testing (which is our ultimate need).

Think of any possibilities for system rules of choosing the kernel?
During training, we can make the error arbitrarily low by making the window sufficiently small, but this will have an ill-effect during testing (which is our ultimate need).

Think of any possibilities for system rules of choosing the kernel?

One possibility is to use cross-validation. Break up the data into a training set and a validation set. Then, perform training on the training set with varying bandwidths. Select the bandwidth that minimizes the error on the validation set.
Parzen Window-Based Classifiers

- During training, we can made the error arbitrarily low by making the window sufficiently small, but this will have an ill-effect during testing (which is our ultimate need).
- Think of any possibilities for system rules of choosing the kernel?
- One possibility is to use cross-validation. Break up the data into a training set and a validation set. Then, perform training on the training set with varying bandwidths. Select the bandwidth that minimizes the error on the validation set.
- There is little theoretical justification for choosing one window width over another.
Selecting the best window / bandwidth is a severe limiting factor for Parzen window estimators.

$k_n$-NN methods circumvent this problem by making the window size a function of the actual training data.
$k_n$ Nearest Neighbor Methods

- Selecting the best window / bandwidth is a severe limiting factor for Parzen window estimators.
- $k_n$-NN methods circumvent this problem by making the window size a function of the actual training data.
- The basic idea here is to center our window around $x$ and let it grow until it capture $k_n$ samples, where $k_n$ is a function of $n$.
  - These samples are the $k_n$ nearest neighbors of $x$.
  - If the density is high near $x$ then the window will be relatively small leading to good resolution.
  - If the density is low near $x$, the window will grow large, but it will stop soon after it enters regions of higher density.
$k_n$ Nearest Neighbor Methods

- Selecting the best window / bandwidth is a severe limiting factor for Parzen window estimators.
- $k_n$-NN methods circumvent this problem by making the window size a function of the actual training data.
- The basic idea here is to center our window around $x$ and let it grow until it capture $k_n$ samples, where $k_n$ is a function of $n$.
  - These samples are the $k_n$ nearest neighbors of $x$.
  - If the density is high near $x$ then the window will be relatively small leading to good resolution.
  - If the density is low near $x$, the window will grow large, but it will stop soon after it enters regions of higher density.
- In either case, we estimate $p_n(x)$ according to

$$p_n(x) = \frac{k_n}{nV_n}$$

(22)
Kernel Density Estimation

We want \( k_n \) to go to infinity as \( n \) goes to infinity thereby assuring us that \( k_n/n \) will be a good estimate of the probability that a point will fall in the window of volume \( V_n \).

\[
p_n(x) = \frac{k_n}{nV_n}
\]
$p_n(x) = \frac{k_n}{nV_n}$

- We want $k_n$ to go to infinity as $n$ goes to infinity thereby assuring us that $k_n/n$ will be a good estimate of the probability that a point will fall in the window of volume $V_n$.
- But, we also want $k_n$ to grow sufficiently slowly so that the size of our window will go to zero.
We want $k_n$ to go to infinity as $n$ goes to infinity thereby assuring us that $k_n/n$ will be a good estimate of the probability that a point will fall in the window of volume $V_n$.

But, we also want $k_n$ to grow sufficiently slowly so that the size of our window will go to zero.

Thus, we want $k_n/n$ to go to zero.
\[ p_n(x) = \frac{k_n}{nV_n} \]

- We want \( k_n \) to go to infinity as \( n \) goes to infinity thereby assuring us that \( k_n/n \) will be a good estimate of the probability that a point will fall in the window of volume \( V_n \).
- But, we also want \( k_n \) to grow sufficiently slowly so that the size of our window will go to zero.
- Thus, we want \( k_n/n \) to go to zero.
- Recall these conditions from the earlier discussion; these will ensure that \( p_n(x) \) converges to \( p(x) \) as \( n \) approaches infinity.
Examples of $k_n$-NN Estimation

- Notice the discontinuities in the slopes of the estimate.
- We don’t expect the density estimate from 1 sample to be very good, but in the case of $k$-NN it will diverge!

- With $n = 1$ and $k_n = \sqrt{n} = 1$, the estimate for $p_n(x)$ is

$$p_n(x) = \frac{1}{2|x - x_1|}$$ (23)
But, as we increase the number of samples, the estimate will improve.
Limitations

- The $k_n$-NN Estimator suffers from an analogous flaw from which the Parzen window methods suffer. What is it?
Limitations

- The $k_n$-NN Estimator suffers from an analogous flaw from which the Parzen window methods suffer. What is it?
- How do we specify the $k_n$?
- We saw earlier that the specification of $k_n$ can lead to radically different density estimates (in practical situations where the number of training samples is limited).
Limitations

- The $k_n$-NN Estimator suffers from an analogous flaw from which the Parzen window methods suffer. What is it?
- How do we specify the $k_n$?
- We saw earlier that the specification of $k_n$ can lead to radically different density estimates (in practical situations where the number of training samples is limited).
- One could obtain a sequence of estimates by taking $k_n = k_1 \sqrt{n}$ and choose different values of $k_1$. 
Limitations

- The $k_n$-NN Estimator suffers from an analogous flaw from which the Parzen window methods suffer. What is it?
- How do we specify the $k_n$?
- We saw earlier that the specification of $k_n$ can lead to radically different density estimates (in practical situations where the number of training samples is limited).
- One could obtain a sequence of estimates by taking $k_n = k_1 \sqrt{n}$ and choose different values of $k_1$.
- But, like the Parzen window size, one choice is as good as another absent any additional information.
Limitations

- The $k_n$-NN Estimator suffers from an analogous flaw from which the Parzen window methods suffer. What is it?
- How do we specify the $k_n$?
- We saw earlier that the specification of $k_n$ can lead to radically different density estimates (in practical situations where the number of training samples is limited).
- One could obtain a sequence of estimates by taking $k_n = k_1 \sqrt{n}$ and choose different values of $k_1$.
- But, like the Parzen window size, one choice is as good as another absent any additional information.
- Similarly, in classification scenarios, we can base our judgement on classification error.
We can directly apply the $k$-NN methods to estimate the posterior probabilities $P(\omega_i|x)$ from a set of $n$ labeled samples.
We can directly apply the $k$-NN methods to estimate the posterior probabilities $P(\omega_i|x)$ from a set of $n$ labeled samples. Place a window of volume $V$ around $x$ and capture $k$ samples, with $k_i$ turning out to be of label $\omega_i$. The estimate for the joint probability is thus

$$p_n(x, \omega_i) = \frac{k_i}{knv}$$

A reasonable estimate for the posterior is thus

$$P_n(\omega_i|x) = \frac{p_n(x, \omega_i)}{\sum_c p_n(x, \omega_c)} = \frac{k_i}{kn}$$

Hence, the posterior probability for $\omega_i$ is simply the fraction of samples within the window that are labeled $\omega_i$. This is a simple and intuitive result.
We can directly apply the $k$-NN methods to estimate the posterior probabilities $P(\omega_i|x)$ from a set of $n$ labeled samples.

Place a window of volume $V$ around $x$ and capture $k$ samples, with $k_i$ turning out to be of label $\omega_i$.

The estimate for the joint probability is thus

$$p_n(x, \omega_i) = \frac{k_i}{nV}$$  \hspace{1cm} (24)
We can directly apply the $k$-NN methods to estimate the posterior probabilities $P(\omega_i|x)$ from a set of $n$ labeled samples.

Place a window of volume $V$ around $x$ and capture $k$ samples, with $k_i$ turning out to be of label $\omega_i$.

The estimate for the joint probability is thus

$$p_n(x, \omega_i) = \frac{k_i}{nV}$$ (24)

A reasonable estimate for the posterior is thus

$$P_n(\omega_i|x) = \frac{p_n(x, \omega_i)}{\sum_c p_n(x, \omega_c)} = \frac{k_i}{k}$$ (25)
We can directly apply the $k$-NN methods to estimate the posterior probabilities $P(\omega_i|x)$ from a set of $n$ labeled samples.

Place a window of volume $V$ around $x$ and capture $k$ samples, with $k_i$ turning out to be of label $\omega_i$.

The estimate for the joint probability is thus

$$p_n(x, \omega_i) = \frac{k_i}{nV}$$

(24)

A reasonable estimate for the posterior is thus

$$P_n(\omega_i|x) = \frac{p_n(x, \omega_i)}{\sum_c p_n(x, \omega_c)} = \frac{k_i}{k}$$

(25)

Hence, the posterior probability for $\omega_i$ is simply the fraction of samples within the window that are labeled $\omega_i$. This is a simple and intuitive result.
Figure-ground discrimination is an important low-level vision task.

- Want to separate the pixels that contain some foreground object (specified in some meaningful way) from the background.

![Example: Figure-Ground Discrimination](source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.)
This paper presents a method for figure-ground discrimination based on non-parametric densities for the foreground and background.

They use a subset of the pixels from each of the two regions.

They propose an algorithm called **iterative sampling-expectation** for performing the actual segmentation.

The required input is simply a region of interest (mostly) containing the object.
Given a set of \( n \) samples \( S = \{x_i\} \) where each \( x_i \) is a \( d \)-dimensional vector.

We know the kernel density estimate is defined as

\[
\hat{p}(y) = \frac{1}{n\sigma_1 \ldots \sigma_d} \sum_{i=1}^{n} \prod_{j=1}^{d} \varphi \left( \frac{y_j - x_{ij}}{\sigma_j} \right)
\]  

(26)

where the same kernel \( \varphi \) with different bandwidth \( \sigma_j \) is used in each dimension.
The representation used here is a function of RGB:

\[ r = \frac{R}{R + G + B} \]  
\[ g = \frac{G}{R + G + B} \]  
\[ s = \frac{R + G + B}{3} \]

Separating the chromaticity from the brightness allows them to use a wider bandwidth in the brightness dimension to account for variability due to shading effects.

And, much narrower kernels can be used on the \( r \) and \( g \) chromaticity channels to enable better discrimination.
The Color Density

Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.

- Given a sample of pixels $S = \{ \mathbf{x}_i = (r_i, g_i, s_i) \}$, the color density estimate is given by

$$\hat{P}(\mathbf{x} = (r, g, s)) = \frac{1}{n} \sum_{i=1}^{n} K_{\sigma_r}(r - r_i)K_{\sigma_g}(g - g_i)K_{\sigma_s}(s - s_i)$$  \hspace{1cm} (30)

where we have simplified the kernel definition:

$$K_{\sigma}(t) = \frac{1}{\sigma} \varphi \left( \frac{t}{\sigma} \right)$$  \hspace{1cm} (31)

- They use Gaussian kernels

$$K_{\sigma}(t) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left[ -\frac{1}{2} \left( \frac{t}{\sigma} \right)^2 \right]$$  \hspace{1cm} (32)

with a different bandwidth in each dimension.
The bandwidth for each channel is calculated directly from the image based on sample statistics.

\[ \sigma \approx 1.06\hat{\sigma}n^{-1/5} \]  

where \( \hat{\sigma}^2 \) is the sample variance.
Initialization: Choosing the Initial Scale
Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.

- For initialization, they compute a distance between the foreground and background distribution by varying the scale of a single Gaussian kernel (on the foreground).

- To evaluate the “significance” of a particular scale, they compute the normalized KL-divergence:

\[
\text{nKL}(\hat{P}_{fg} || \hat{P}_{bg}) = - \sum_{i=1}^{n} \hat{P}_{fg}(x_i) \log \frac{\hat{P}_{fg}(x_i)}{\hat{P}_{bg}(x_i)} \]

where \(\hat{P}_{fg}\) and \(\hat{P}_{bg}\) are the density estimates for the foreground and background regions respectively. To compute each, they use about 6% of the pixels (using all of the pixels would lead to quite slow performance).
Example: Figure-Ground Discrimination

Figure 2. Segmentation results at different scales where \( P_{CM} \) and \( P_{C8} \) are the PDFs of the figure and ground respectively, and \( P_{CM} \) and \( P_{C8} \) are the PDFs of the figure and ground respectively, and \( P_{DC} \) is a sampled pixel in \( CB \).

3. The Iterative Sampling - Expectation algorithm

We assume that the pixels in an image were generated by two processes — the figure and ground processes. Then the figure-ground discrimination problem involves assigning each pixel to the process that generated it. If we knew the probability density functions of the two processes, then we could assign each pixel to the process with the maximum likelihood. Likewise, if we knew the assignment of each pixel, then we could estimate the probability density functions of the two processes. This chicken-and-egg problem suggests an iterative framework for computing the segmentation. Unlike the traditional EM algorithm which assumes mixtured Gaussian models, we employ the kernel density estimation to approximate the color density distribution of each process. A set of pixels are sampled from the image for kernel density estimation. Thus, the maximization step in the EM algorithm is replaced with the sampling step. This gives the basic structure of an iterative sampling-expectation (SE) algorithm:

1. Start with a Gaussian spatial distribution for all pixels in the image. We select the scale of the initial Gaussian distribution which maximizes the normalized KL-divergence given in Eq. (3). Fig. 2 demonstrates that we can obtain the correct segmentation at the right scale.

2. S step: uniformly sample a set of pixels from the image for kernel density estimation.

3. E step: re-assign pixels to the two processes based on maximum likelihood estimation.

4. Repeat steps 2 and 3 until the segmentation becomes stable.

Since the assignments of pixels are soft, we cannot use the kernel density estimation in Eq. (1) directly. Instead we design a weighted kernel density estimation and make use of the samples from both the foreground and background. Given a set of samples \( CB \) and \( BP \) from the whole image, we estimate the probabilities of a pixel belonging to the foreground and background by first calculating the following two values

\[
DB \text{ and } DD
\]

Through normalization we get the soft assignments of each pixel to the foreground and background:

\[
P_{CM} \text{ and } P_{C8}
\]

where \( C6 \) is the number of samples (we use \( BI \) of the pixels in the image) and \( CS \) is the dimension of the feature space.
Given the initial segmentation, they need to refine the models and labels to adapt better to the image.

However, this is a chicken-and-egg problem. If we know the labels, we could compute the models, and if we knew the models, we could compute the best labels.
Given the initial segmentation, they need to refine the models and labels to adapt better to the image.

However, this is a chicken-and-egg problem. If we know the labels, we could compute the models, and if we knew the models, we could compute the best labels.

They propose an EM algorithm for this. The basic idea is to alternate between estimating the probability that each pixel is of the two classes, and then given this probability to refine the underlying models.

EM is guaranteed to converge (but only to a local minimum).
1. Initialize using the normalized KL-divergence.
1. Initialize using the normalized KL-divergence.

2. Uniformly sample a set of pixel from the image to use in the kernel density estimation. This is essentially the ‘M’ step (because we have a non-parametric density).
1. Initialize using the normalized KL-divergence.

2. Uniformly sample a set of pixel from the image to use in the kernel density estimation. This is essentially the ‘M’ step (because we have a non-parametric density).

3. Update the pixel assignment based on maximum likelihood (the ‘E’ step).

\[
\hat{P}_{fg}(y) = \frac{1}{Z} \sum_{i=1}^{n} \hat{P}_{fg}(x_i) \prod_{j=1}^{K} K(y_j - x_{ij}) \sigma_j
\]
1. Initialize using the normalized KL-divergence.

2. Uniformly sample a set of pixel from the image to use in the kernel density estimation. This is essentially the ‘M’ step (because we have a non-parametric density).

3. Update the pixel assignment based on maximum likelihood (the ‘E’ step).

4. Repeat until stable.
1. Initialize using the normalized KL-divergence.

2. Uniformly sample a set of pixels from the image to use in the kernel density estimation. This is essentially the ‘M’ step (because we have a non-parametric density).

3. Update the pixel assignment based on maximum likelihood (the ‘E’ step).

4. Repeat until stable.

One can use a hard assignment of the pixels and the kernel density estimator we’ve discussed, or a soft assignment of the pixels and then a weighted kernel density estimate (the weight is between the different classes).
1. Initialize using the normalized KL-divergence.

2. Uniformly sample a set of pixel from the image to use in the kernel density estimation. This is essentially the ‘M’ step (because we have a non-parametric density).

3. Update the pixel assignment based on maximum likelihood (the ‘E’ step).

4. Repeat until stable.

- One can use a hard assignment of the pixels and the kernel density estimator we’ve discussed, or a soft assignment of the pixels and then a weighted kernel density estimate (the weight is between the different classes).

- The overall probability of a pixel belonging to the foreground class

\[
\hat{P}_{fg}(y) = \frac{1}{Z} \sum_{i=1}^{n} \hat{P}_{fg}(x_i) \prod_{j=1}^{d} K \left( \frac{y_j - x_{ij}}{\sigma_j} \right) \quad (35)
\]
Results: Stability
Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.

Figure 2. Segmentation results at different scales where \( /CM /C8 /CU /CV \) and \( /CM /C8 /CQ /CV \) are the PDFs of the figure and ground respectively, and \( /DC /CX \) is a sampled pixel in \( /CB /BP /CU /DC /CX /CV \).

3. The Iterative Sampling - Expectation algorithm

We assume that the pixels in an image were generated by two processes — the figure and ground processes. Then the figure-ground discrimination problem involves assigning each pixel to the process that generated it. If we knew the probability density functions of the two processes, then we could assign each pixel to the process with the maximum likelihood. Likewise, if we knew the assignment of each pixel, then we could estimate the probability density functions of the two processes. This chicken-and-egg problem suggests an iterative framework for computing the segmentation. Unlike the traditional EM algorithm which assumes mixed Gaussian models, we employ the kernel density estimation to approximate the color density distribution of each process. A set of pixels are sampled from the image for kernel density estimation. Thus, the maximization step in the EM algorithm is replaced with the sampling step. This gives the basic structure of an iterative sampling-expectation (SE) algorithm:

1. Start with a Gaussian spatial distribution for all pixels in the image. We select the scale of the initial Gaussian distribution which maximizes the normalized KL-divergence given in Eq. (3). Fig. 2 demonstrates that we can obtain the correct segmentation at the right scale.

2. S step: uniformly sample a set of pixels from the image for kernel density estimation.

3. E step: re-assign pixels to the two processes based on maximum likelihood estimation.

4. Repeat steps 2 and 3 until the segmentation becomes stable.

Since the assignments of pixels are soft, we cannot use the kernel density estimation in Eq. (1) directly. Instead we design a weighted kernel density estimation and make use of the samples from both the foreground and background. Given a set of samples \( /CB /BP /CU /DC /CX /CV \) from the whole image, we estimate the probabilities of a pixel belonging to the foreground and background by first calculating the following two values

\[
\]

\[
\]

where \( /C6 \) is the number of samples (we use \( /BI /B1 \) of the pixels in the image) and \( /CS \) is the dimension of the feature space. Through normalization we get the soft assignments of each pixel to the foreground and background:

\[
\]

\[
\]
Results
Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.

The above equations update the assignments of a pixel by integrating evidence from the samples in its neighborhood. To obtain the final segmentation, we assign a pixel to the foreground if

\[
\frac{C}{BQ} > \frac{C}{CM} \frac{C}{C8} \frac{C}{CV},
\]

otherwise to the background. In this way, we let the figure and ground processes "compete" to explain the pixel data.

4. Experiments

We did experiments to test the sensitivity of the SE algorithm to the initial distribution. We shift the center of the initial Gaussian distribution off the center of the image and compare the results with the one obtained by locating the distribution at the image center. Fig. 3 indicates that the average assignment error at each pixel is less than

\[
\frac{B}{BQ} \frac{B}{BM} \frac{B}{BH},
\]

when the shift is less than 10 pixels or 12% of the figure size.

To test how sensitive the SE algorithm is to the initial sampling, we ran the SE algorithm over the same image several times. Fig. 4 illustrates that the results are very stable. In comparison, we ran the traditional EM algorithm over the same image with three mixtures Gaussians for the figure and three for the background. When initializing the cluster centers at different places, we obtained very different segmentation results as shown in Fig. 4.

We further compared the SE algorithm with a more sophisticated EM algorithm proposed by Carson et al. [4]. In [4], the Minimum Description Length principle is used to select the number of mixture models. Fig. 5 demonstrates that the EM algorithm tends to merge the figure with part of the background, while our algorithm gives better segmentation results.

5. Conclusion

In this paper, we present a novel segmentation method for figure-ground discrimination. The use of kernel density estimation for color distribution enables automatic selection of weights of different cues based on the bandwidth calculation from the image itself. The size and shape of the figure are determined adaptively by the competition between the figure and background using the iterative sampling-expectation algorithm. Consequently, the combination of kernel density estimation for color distribution and the iterative sampling-expectation algorithm have resulted in encouraging segmentation results.

References


Results

Source: Zhao and Davis. Iterative Figure-Ground Discrimination. ICPR 2004.

The above equations update the assignments of a pixel by integrating evidence from the samples in its neighborhood. To obtain the final segmentation, we assign a pixel to the foreground if

\[
\frac{C_M}{C_8} \frac{C_R}{C_V}
\]

otherwise to the background. In this way, we let the figure and ground processes "compete" to explain the pixel data.

4. Experiments

We did experiments to test the sensitivity of the SE algorithm to the initial distribution. We shift the center of the initial Gaussian distribution off the center of the image and compare the results with the one obtained by locating the distribution at the image center. Fig. 3 indicates that the average assignment error at each pixel is less than

\[
\frac{B_C}{B_M} \frac{B_R}{B_H}
\]

when the shift is less than 10 pixels or 12% of the figure size.

To test how sensitive the SE algorithm is to the initial sampling, we ran the SE algorithm over the same image several times. Fig. 4 illustrates that the results are very stable. In comparison, we ran the traditional EM algorithm over the same image with three mixtured Gaussians for the figure and three for the background. When initializing the cluster centers at different places, we obtained very different segmentation results as shown in Fig. 4.

We further compared the SE algorithm with a more sophisticated EM algorithm proposed by Carson et al. [4]. In [4], the Minimum Description Length principle is used to select the number of mixture models. Fig. 5 demonstrates that the EM algorithm tends to merge the figure with part of the background, while our algorithm gives better segmentation results.

5. Conclusion

In this paper, we present a novel segmentation method for figure-ground discrimination. The use of kernel density estimation for color distribution enables automatic selection of weights of different cues based on the bandwidth calculation from the image itself. The size and shape of the figure are determined adaptively by the competition between the figure and background using the iterative sampling-expectation algorithm. Consequently, the combination of kernel density estimation for color distribution and the iterative sampling-expectation algorithm have resulted in encouraging segmentation results.

References