Today

- Introduction to nonparametric techniques
- Basic Issues in Density Estimation
- Two Density Estimation Methods
  1. Parzen Windows
  2. Nearest Neighbors
Non-Parametric Methods

- Neither probability distribution nor discriminant function is known
  - Happens quite often
- All we have is labeled data
  - Estimate the probability distribution from the labeled data

a lot is known "easier"

little is known "harder"
In previous lectures we assumed that either

1. someone gives us the density $p(x)$
   - In pattern recognition applications this never happens

2. someone gives us $p(x|\theta)$
   - Does happen sometimes, but
     - we are likely to suspect whether the given $p(x|\theta)$ models the data well
     - Most parametric densities are unimodal (have a single local maximum), whereas many practical problems involve multi-modal densities
Nonparametric procedures can be used with arbitrary distributions and without any assumption about the forms of the underlying densities.

There are two types of nonparametric methods:

- Parzen windows
  - Estimate likelihood $p(x / c_j)$
- Nearest Neighbors
  - Bypass likelihood and go directly to posterior estimation $P(c_j / x)$
Nonparametric techniques attempt to estimate the underlying density functions from the training data. Idea: the more data in a region, the larger is the density function.

\[ Pr[X \in \mathbb{R}] = \int_{\mathbb{R}} f(x) dx \]

The average of \( f(x) \) over \( \mathbb{R} \) is equal to the probability of \( X \) being in the region. 

\[ p(x) \]

salmon length \( x \)
NonParametric Techniques: Introduction

\[ Pr[X \in \mathcal{R}] = \int_{\mathcal{R}} f(x)dx \]

- How can we approximate \( Pr[X \in \mathcal{R}_1] \) and \( Pr[X \in \mathcal{R}_2] \)?
  - \( Pr[X \in \mathcal{R}_1] \approx \frac{6}{20} \) and \( Pr[X \in \mathcal{R}_2] \approx \frac{6}{20} \)

- Should the density curves above \( \mathcal{R}_1 \) and \( \mathcal{R}_2 \) be equally high?
  - No, since is \( \mathcal{R}_1 \) smaller than \( \mathcal{R}_2 \)
  
  \[ Pr[X \in \mathcal{R}_1] = \int_{\mathcal{R}_1} f(x)dx \approx \int_{\mathcal{R}_2} f(x)dx = Pr[X \in \mathcal{R}_2] \]

- To get density, normalize by region size

\[ p(x) \]

\[ \mathcal{R}_1 \quad \mathcal{R}_2 \]

salmon length \( x \)
NonParametric Techniques: Introduction

- Assuming \( f(x) \) is basically flat inside \( \mathcal{R} \)

\[
\frac{\text{# of samples in } \mathcal{R}}{\text{total # of samples}} \approx Pr[X \in \mathcal{R}] = \int_{\mathcal{R}} f(y)dy \approx f(x) \times \text{Volume}(\mathcal{R})
\]

- Thus, density at a point \( x \) inside \( \mathcal{R} \) can be approximated

\[
f(x) \approx \frac{\text{# of samples in } \mathcal{R}}{\text{total # of samples}} \times \frac{1}{\text{Volume}(\mathcal{R})}
\]

- Now let’s derive this formula more formally
Binomial Random Variable

- Let us flip a coin $n$ times (each one is called “trial”)
  - Probability of head $\rho$, probability of tail is $1-\rho$
- Binomial random variable $K$ counts the number of heads in $n$ trials
  \[ P(K = k) = \binom{n}{k} \rho^k (1 - \rho)^{n-k} \]

where \( \binom{n}{k} = \frac{n!}{k!(n-k)!} \)

- Mean is \( E(K) = n\rho \)
- Variance is \( \text{var}(K) = n\rho(1 - \rho) \)
Density Estimation: Basic Issues

- From the definition of a density function, probability $\rho$ that a vector $\mathbf{x}$ will fall in region $\mathcal{R}$ is:

$$\rho = Pr[\mathbf{x} \in \mathcal{R}] = \int_{\mathcal{R}} p(\mathbf{x}') d\mathbf{x}'$$

- Suppose we have samples $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ drawn from the distribution $p(\mathbf{x})$. The probability that $k$ points fall in $\mathcal{R}$ is then given by binomial distribution:

$$Pr[K = k] = \binom{n}{k} \rho^k (1 - \rho)^{n-k}$$

- Suppose that $k$ points fall in $\mathcal{R}$, we can use MLE to estimate the value of $\rho$. The likelihood function is

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_n | \rho) = \binom{n}{k} \rho^k (1 - \rho)^{n-k}$$
Density Estimation: Basic Issues

\[ p(x_1, \ldots, x_n \mid \rho) = \binom{n}{k} \rho^k (1 - \rho)^{n-k} \]

- This likelihood function is maximized at \( \rho = \frac{k}{n} \)
- Thus the MLE is \( \hat{\rho} = \frac{k}{n} \)
- Assume that \( p(x) \) is continuous and that the region \( \mathcal{R} \) is so small that \( p(x) \) is approximately constant in \( \mathcal{R} \).

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

- \( x \) is in \( \mathcal{R} \) and \( V \) is the volume of \( \mathcal{R} \)
- Recall from the previous slide: \( \rho = \int_{\mathcal{R}} p(x') \, dx' \)
- Thus \( p(x) \) can be approximated: \( p(x) \approx \frac{k/n}{V} \)
Density Estimation: Basic Issues

- This is exactly what we had before:

  \[
  p(x) \approx \frac{k/n}{V} \quad x \text{ is inside some region } \mathcal{R}
  \]

  \[k = \text{number of samples inside } \mathcal{R}\]

  \[n = \text{total number of samples inside } \mathcal{R}\]

  \[V = \text{volume of } \mathcal{R}\]

- Our estimate will always be the average of true density over \( \mathcal{R} \):

  \[
  p(x) \approx \frac{k/n}{V} = \frac{\hat{\rho}}{V} \approx \frac{\int p(x')dx'}{V}
  \]

- Ideally, \( p(x) \) should be constant inside \( \mathcal{R} \)
Density Estimation: Histogram

\[ p(x) \approx \frac{k/n}{V} \]

If regions \( R_i \)'s do not overlap, we have a histogram
Density Estimation: Accuracy

- How accurate is density approximation \( p(x) \approx \frac{k/n}{V} \)?
- We have made two approximations

1. \( \hat{\rho} = \frac{k}{n} \)
   - as \( n \) increases, this estimate becomes more accurate

2. \( \int_{\mathbb{R}} p(x')dx' \approx p(x)V \)
   - as \( \mathbb{R} \) grows smaller, the estimate becomes more accurate
   - As we shrink \( \mathbb{R} \) we have to make sure it contains samples, otherwise our estimated \( p(x) = 0 \) for all \( x \) in \( \mathbb{R} \)

- Thus in theory, if we have an unlimited number of samples, to we get convergence as we simultaneously increase the number of samples \( n \), and shrink region \( \mathbb{R} \), but not too much so that \( \mathbb{R} \) still contains a lot of samples.
**Density Estimation: Accuracy**

\[ p(x) \approx \frac{k}{n} \frac{1}{V} \]

- In practice, the number of samples is always fixed.
- Thus the only available option to increase the accuracy is by decreasing the size of \( R \) (\( V \) gets smaller).
  - If \( V \) is too small, \( p(x) = 0 \) for most \( x \), because most regions will have no samples.
  - Thus have to find a compromise for \( V \)
    - not too small so that it has enough samples
    - but also not too large so that \( p(x) \) is approximately constant inside \( V \).
Density Estimation: Two Approaches

\[ p(x) \approx \frac{k/n}{V} \]

1. Parzen Windows:
   - Choose a fixed value for volume \( V \) and determine the corresponding \( k \) from the data

2. k-Nearest Neighbors
   - Choose a fixed value for \( k \) and determine the corresponding volume \( V \) from the data

   Under appropriate conditions and as number of samples goes to infinity, both methods can be shown to converge to the true \( p(x) \)
Parzen Windows

- In Parzen-window approach to estimate densities we fix the size and shape of region $\mathcal{R}$
- Let us assume that the region $\mathcal{R}$ is a $d$-dimensional hypercube with side length $h$ thus it’s volume is $h^d$
To estimate the density at point $x$, simply center the region $\mathcal{R}$ at $x$, count the number of samples in $\mathcal{R}$, and substitute everything in our formula

$$p(x) \approx \frac{k / n}{V}$$

$$p(x) \approx \frac{3/6}{10}$$
Parzen Windows

- We wish to have an analytic expression for our approximate density $R$.
- Let us define a window function

$$
\varphi(u) = \begin{cases} 
1 & |u_j| \leq \frac{1}{2} \quad j = 1, \ldots, d \\
0 & \text{otherwise}
\end{cases}
$$

1 dimension

$\varphi(u)$

1/2

$1$

$u$

2 dimensions

$u_1$

$u_2$

$\varphi$ is 1 inside

$\varphi$ is 0 outside
Recall we have samples $x_1, x_2, \ldots, x_n$. Then

$$\phi\left(\frac{x - x_i}{h}\right) = \begin{cases} 1 & \text{if } |x - x_i| \leq \frac{h}{2} \\ 0 & \text{otherwise} \end{cases}$$

$$j = 1, \ldots, d$$

If $x_i$ is inside the hypercube with width $h$ and centered at $x$.
How do we count the total number of sample points $x_1, x_2, \ldots, x_n$ which are inside the hypercube with side $h$ and centered at $x$?

$$k = \sum_{i=1}^{i=n} \varphi \left( \frac{x - x_i}{h} \right)$$

Recall $p(x) \approx \frac{k}{n}$

Thus we get the desired analytical expression for the estimate of density $p_\varphi(x)$

$$p_\varphi(x) = \frac{1}{n} \sum_{i=1}^{i=n} \frac{1}{h^d} \varphi \left( \frac{x - x_i}{h} \right)$$
Parzen Windows

\[ p_\varphi(x) = \frac{1}{n} \sum_{i=1}^{i=n} \frac{1}{h^d} \varphi \left( \frac{x - x_i}{h} \right) \]

- Let’s make sure \( p_\varphi(x) \) is in fact a density

- \( p_\varphi(x) \geq 0 \quad \forall x \)

\[ \int p_\varphi(x)dx = \int \frac{1}{n} \sum_{i=1}^{i=n} \frac{1}{h^d} \varphi \left( \frac{x - x_i}{h} \right)dx = \frac{1}{h^d n} \sum_{i=1}^{i=n} \int \varphi \left( \frac{x - x_i}{h} \right)dx \]

\[ = \frac{1}{n} \frac{1}{h^d} \sum_{i=1}^{i=n} h^d = 1 \]
Today

- Continue nonparametric techniques
  1. Finish Parzen Windows
  2. Start Nearest Neighbors (hopefully)
Parzen Windows

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

- To estimate the density at point \( x \), simply center the region \( \mathcal{R} \) at \( x \), count the number of samples in \( \mathcal{R} \), and substitute everything in our formula.

\[ x \text{ is inside some region } \mathcal{R} \]
\[ k = \text{number of samples inside } \mathcal{R} \]
\[ n = \text{total number of samples inside } \mathcal{R} \]
\[ V = \text{volume of } \mathcal{R} \]

\[ p(x) \approx \frac{3}{6} \]
\[ \frac{10}{10} \]
Parzen Windows

- Formula for Parzen window estimation

\[
p_\varphi(x) = \frac{1}{n} \sum_{i=1}^{i=n} \varphi\left( \frac{x - x_i}{h} \right) h^d = \frac{1}{n} \sum_{i=1}^{i=n} \frac{1}{h^d} \varphi\left( \frac{x - x_i}{h} \right)
\]
Parzen Windows: Example in 1D

\[ p_\phi(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h^d} \phi \left( \frac{x - x_i}{h} \right) \]

- Suppose we have 7 samples \( D = \{2, 3, 4, 8, 10, 11, 12\} \)

- Let window width \( h = 3 \), estimate density at \( x = 1 \)

\[ p_\phi(1) = \frac{1}{7} \sum_{i=1}^{7} \frac{1}{3} \phi \left( \frac{1 - x_i}{3} \right) = \frac{1}{21} \left[ \phi \left( \frac{1 - 2}{3} \right) + \phi \left( \frac{1 - 3}{3} \right) + \phi \left( \frac{1 - 4}{3} \right) + \ldots + \phi \left( \frac{1 - 12}{3} \right) \right] \]

\[
\begin{aligned}
&\left| \frac{1}{3} \right| \leq 1/2 \\
&\left| \frac{2}{3} \right| > 1/2 \\
&\left| -1 \right| > 1/2 \\
&\left| -\frac{11}{3} \right| > 1/2
\end{aligned}
\]

\[ p_\phi(1) = \frac{1}{7} \sum_{i=1}^{7} \frac{1}{3} \phi \left( \frac{1 - x_i}{3} \right) = \frac{1}{21} \left[ 1 + 0 + 0 + \ldots + 0 \right] = \frac{1}{21} \]
Parzen Windows: Sum of Functions

- Fix $x$, let $i$ vary and ask
  - For which samples $x_i$ is $\varphi\left(\frac{x - x_i}{h}\right) = 1$?

- Now fix $f$ and let $x$ vary and ask
  - For which $x$ is $\varphi\left(\frac{x - x_f}{h}\right) = 1$? For all $x$ in gray box

- Thus $\varphi\left(\frac{x - x_f}{h}\right) = 1$ is simply a function which is 1 inside square of width $h$ centered at $x_f$ and 0 otherwise!
Now let’s look at our density estimate $p_\phi(x)$ again:

$$p_\phi(x) = \frac{1}{n} \sum_{i=1}^{i=n} \frac{1}{h^d} \varphi\left(\frac{x - x_i}{h}\right) = \sum_{i=1}^{i=n} \frac{1}{nh^d} \varphi\left(\frac{x - x_i}{h}\right)$$

1 inside square centered at $x_i$
0 otherwise

Thus $p_\phi(x)$ is just a sum of $n$ “box like” functions each of height $\frac{1}{nh^d}$
Parzen Windows: Example in 1D

- Let’s come back to our example
  - 7 samples $D=\{2,3,4,8,10,11,12\}$, $h=3$

- To see what the function looks like, we need to generate 7 boxes and add them up
- The width is $h=3$ and the height, according to previous slide is

$$\frac{1}{nh^d} = \frac{1}{21}$$
In essence, window function $\varphi$ is used for interpolation: each sample $x_i$ contributes to the resulting density at $x$ if $x$ is close enough to $x_i$. 

$$p_\varphi(x)$$
Parzen Windows: Drawbacks of Hypercube $\phi$

- As long as sample point $x_i$ and $x$ are in the same hypercube, the contribution of $x_i$ to the density at $x$ is constant, regardless of how close $x_i$ is to $x$

\[
\varphi\left(\frac{x - x_1}{h}\right) = \varphi\left(\frac{x - x_2}{h}\right) = 1
\]

- The resulting density $p_\varphi(x)$ is not smooth, it has discontinuities
Parzen Windows: general $\varphi$

$$p_\varphi(x) = \frac{1}{n} \sum_{i=1}^{i=n} \frac{1}{h^d} \varphi\left(\frac{x - x_i}{h}\right)$$

- We can use a general window $\varphi$ as long as the resulting $p_\varphi(x)$ is a legitimate density, i.e.

1. $p_\varphi(u) \geq 0$
   - satisfied if $\varphi(u) \geq 0$

2. $\int p_\varphi(x)dx = 1$
   - satisfied if $\int \varphi(u)du = 1$

$$\int p_\varphi(x)dx = \frac{1}{nh^d} \sum_{i=1}^{i=n} \int \varphi\left(\frac{x - x_i}{h}\right)dx = \frac{1}{nh^d} \sum_{i=1}^{n} \int h^n \varphi(u)du = 1$$

*change coordinates to $u = \frac{x - x_i}{h}$, thus $du = \frac{dx}{h}$*
Parzen Windows: general \( \varphi \)

\[
p_{\varphi}(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h^d} \varphi \left( \frac{x - x_i}{h} \right)
\]

- Notice that with the general window \( \varphi \) we are no longer counting the number of samples inside \( \mathcal{R} \).
- We are counting the weighted average of potentially every single sample point (although only those within distance \( h \) have any significant weight).

- With infinite number of samples, and appropriate conditions, it can still be shown that

\[
p_{\varphi}^{n}(x) \rightarrow p(x)
\]
**Parzen Windows: Gaussian $\phi$**

$$p_\phi(x) = \frac{1}{n} \sum_{i=1}^{i=n} \frac{1}{h^d} \varphi \left( \frac{x - x_i}{h} \right)$$

- A popular choice for $\varphi$ is $N(0,1)$ density

$$\varphi(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}$$

- Solves both drawbacks of the “box” window
  - Points $x$ which are close to the sample point $x_i$ receive higher weight
  - Resulting density $p_\phi(x)$ is smooth
Let’s come back to our example

- 7 samples $D=\{2,3,4,8,10,11,12\}$, $h=1$

$$p_\varphi(x) = \frac{1}{7} \sum_{i=1}^{7} \varphi(x - x_i)$$

$p_\varphi(x)$ is the sum of 7 Gaussians, each centered at one of the sample points, and each scaled by $1/7$. 
Parzen Windows: Did We Solve the Problem?

- Let’s test if we solved the problem
  1. Draw samples from a known distribution
  2. Use our density approximation method and compare with the true density
- We will vary the number of samples $n$ and the window size $h$
- We will play with 2 distributions

$N(0, 1)$

triangle and uniform mixture
Parzen Windows: True Density $N(0,1)$

$h=1$

$n=1$

$h=0.5$

$n=10$

$h=0.1$
**Parzen Windows: True Density $N(0,1)$**

$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) = \frac{1}{n \cdot 0.5} \sum_{i=1}^{n} K\left(\frac{x - x_i}{0.5}\right) = \frac{1}{n \cdot 0.1} \sum_{i=1}^{n} K\left(\frac{x - x_i}{0.1}\right)$

$\text{for } h = 1, 0.5, 0.1$

$\text{and for } n = 100, \infty$

**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
Parzen Windows: True density is Mixture of Uniform and Triangle

$h=1$

$n=1$

$h=0.5$

$n=16$

$h=0.2$
Parzen Windows: True density is Mixture of Uniform and Triangle

$h=1$  $h=0.5$  $h=0.2$

$n=256$

$n=\infty$

**FIGURE 4.7.** 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 Windows: Effect of Window Width $h$

- By choosing $h$ we are guessing the region where density is approximately constant.

- Without knowing anything about the distribution, it is really hard to guess where the density is approximately constant.
Parzen Windows: Effect of Window Width $h$

- If $h$ is small, we superimpose $n$ sharp pulses centered at the data
  - Each sample point $x_i$ influences too small range of $x$
  - Smoothed too little: the result will look noisy and not smooth enough
- If $h$ is large, we superimpose broad slowly changing functions,
  - Each sample point $x_i$ influences too large range of $x$
  - Smoothed too much: the result looks oversmoothed or “out-of-focus”
- Finding the best $h$ is challenging, and indeed no single $h$ may work well
  - May need to adapt $h$ for different sample points
- However we can try to learn the best $h$ to use from the test data
In classifiers based on Parzen-window estimation:

- We estimate the densities for each category and classify a test point by the label corresponding to the maximum posterior.

- The decision region for a Parzen-window classifier depends upon the choice of window function as illustrated in the following figure.
Parzen Windows: Classification Example

- For small enough window size $h$, classification on training data is perfect.
- However, decision boundaries are complex, and this solution is not likely to generalize well to novel data.

- For larger window size $h$, classification on training data is not perfect.
- However, decision boundaries are simpler, and this solution is more likely to generalize well to novel data.
Parzen Windows: Summary

- **Advantages**
  - Can be applied to the data from any distribution
  - In theory can be shown to converge as the number of samples goes to infinity

- **Disadvantages**
  - Number of training data is limited in practice, and so choosing the appropriate window size $h$ is difficult
  - May need large number of samples for accurate estimates
  - Computationally heavy, to classify one point we have to compute a function which potentially depends on all samples
  - Window size $h$ is not trivial to choose
$k$-Nearest Neighbors

- Recall the generic expression for density estimation

$$p(x) \approx \frac{k/n}{V}$$

- In Parzen windows estimation, we fix $V$ and that determines $k$, the number of points inside $V$

- In k-nearest neighbor approach we fix $k$, and find $V$ that contains $k$ points inside
**k-Nearest Neighbors**

- kNN approach seems a good solution for the problem of the “best” window size
  - Let the cell volume be a function of the training data
  - Center a cell about $x$ and let it grows until it captures $k$ samples
  - $k$ are called the $k$ nearest-neighbors of $x$

- 2 possibilities can occur:
  - Density is high near $x$; therefore the cell will be small which provides a good resolution
  - Density is low; therefore the cell will grow large and stop until higher density regions are reached
**k-Nearest Neighbor**

- Of course, now we have a new question
  - How to choose \( k \)?
  - A good “rule of thumb“ is \( k = \sqrt{n} \)
    - Can prove convergence if \( n \) goes to infinity
    - Not too useful in practice, however

- Let’s look at 1-D example
  - we have one sample, i.e. \( n = 1 \)

\[
p(x) \approx \frac{k / n}{V} = \frac{1}{2|x - x_1|}
\]

- But the estimated \( p(x) \) is not even close to a density function:
\[
\int_{-\infty}^{\infty} \frac{1}{2|x - x_1|} \, dx = \infty \neq 1
\]
k-Nearest Neighbor: Gaussian and Uniform plus Triangle Mixture Estimation
Today

- Continue with Nonparametric Density Estimation
  - Finish Nearest Neighbor
**k-Nearest Neighbors**

\[ p(x) \approx \frac{k}{n/V} \]

- kNN approach seems a good solution for the problem of the “best” window size
  - Let the cell volume be a function of the training data
  - Center a cell about \( x \) and let it grows until it captures \( k \) samples
  - \( k \) are called the \( k \) nearest-neighbors of \( x \)
Thus straightforward density estimation \( p(x) \) does not work very well with kNN approach because the resulting density estimate

1. Is not even a density
2. Has a lot of discontinuities (looks very spiky, not differentiable)
3. Even for large regions with no observed samples the estimated density is far from zero (tails are too heavy)

Notice in the theory, if infinite number of samples is available, we could construct a series of estimates that converge to the true density using kNN estimation. However this theorem is not very useful in practice because the number of samples is always limited.
However we shouldn’t give up the nearest neighbor approach yet

Instead of approximating the density $p(x)$, we can use kNN method to approximate the posterior distribution $P(c_i|x)$
- We don’t even need $p(x)$ if we can get a good estimate on $P(c_i|x)$
**k-Nearest Neighbor**

- How would we estimate $P(c_i \mid x)$ from a set of $n$ labeled samples?
- Recall our estimate for density: $p(x) \approx \frac{k}{n} \frac{1}{V}$
- Let’s place a cell of volume $V$ around $x$ and capture $k$ samples
  - $k_i$ samples amongst $k$ labeled $c_i$, then:
    $$p(c_i, x) \approx \frac{k_i}{n} \frac{1}{V}$$
- Using conditional probability, let’s estimate posterior:
  $$p(c_i \mid x) = \frac{p(x, c_i)}{p(x)} \approx \frac{k_i}{n} \frac{1}{V} \frac{1}{\sum_{j=1}^{m} p(x, c_j)} \approx \frac{k_i}{n} \frac{k}{V \sum_{j=1}^{m} \frac{k_j}{n}} = \frac{k_i}{\sum_{j=1}^{m} k_j} = \frac{k_i}{k}$$
Thus our estimate of posterior is just the fraction of samples which belong to class $c_i$:

$$p(c_i \mid x) = \frac{k_i}{k}$$

This is a very simple and intuitive estimate.

Under the zero-one loss function (MAP classifier) just choose the class which has the largest number of samples in the cell.

Interpretation is: given an unlabeled example (that is $x$), find $k$ most similar labeled examples (closest neighbors among sample points) and assign the most frequent class among those neighbors to $x$. 

$k$-Nearest Neighbor
**k-Nearest Neighbor: Example**

- Back to fish sorting
  - Suppose we have 2 features, and collected sample points as in the picture
  - Let $k = 3$

- 2 sea bass, 1 salmon are the 3 nearest neighbors
- Thus classify as sea bass
kNN rule is certainly simple and intuitive, but does it work?
Pretend that we can get an unlimited number of samples
By definition, the best possible error rate is the Bayes rate $E^*$
Even for $k = 1$, the nearest-neighbor rule leads to an error rate greater than $E^*$
But as $n \to \infty$, it can be shown that nearest neighbor rule error rate is smaller than $2E^*$
If we have a lot of samples, the kNN rule will do very well!
FIGURE 4.13. In two dimensions, the nearest-neighbor algorithm leads to a partitioning of the input space into Voronoi cells, each labeled by the category of the training point it contains. In three dimensions, the cells are three-dimensional, and the decision boundary resembles the surface of a crystal. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.
Most parametric distributions would not work for this 2 class classification problem:

- Nearest neighbors will do reasonably well, provided we have a lot of samples
**kNN: How to Choose k?**

- In theory, when the infinite number of samples is available, the larger the $k$, the better is classification (error rate gets closer to the optimal Bayes error rate)
- But the caveat is that all $k$ neighbors have to be close to $x$
  - Possible when infinite # samples available
  - Impossible in practice since # samples is finite
kNN: How to Choose $k$?

- In practice
  1. $k$ should be large so that error rate is minimized
    - $k$ too small will lead to noisy decision boundaries
  2. $k$ should be small enough so that only nearby samples are included
    - $k$ too large will lead to over-smoothed boundaries
- Balancing 1 and 2 is not trivial
  - This is a recurrent issue, need to smooth data, but not too much
**kNN: How to Choose k?**

- For \( k = 1, \ldots, 7 \) point \( x \) gets classified correctly
  - red class
- For larger \( k \) classification of \( x \) is wrong
  - blue class
Basic \textit{kNN} algorithm stores all examples. Suppose we have \( n \) examples each of dimension \( k \)
- \( O(d) \) to compute distance to one example
- \( O(nd) \) to find one nearest neighbor
- \( O(knd) \) to find \( k \) closest examples examples
- Thus complexity is \( O(knd) \)

This is prohibitively expensive for large number of samples
But we need large number of samples for \textit{kNN} to work well!
Reducing Complexity: Editing 1NN

- If all voronoi neighbors have the same class, a sample is useless, we can remove it:

- Number of samples decreases
- We are guaranteed that the decision boundaries stay the same
Reducing Complexity: kNN prototypes

- Explore similarities between samples to represent data as search trees of prototypes

- Advantages: Complexity decreases

- Disadvantages:
  - finding good search tree is not trivial
  - will not necessarily find the closest neighbor, and thus not guaranteed that the decision boundaries stay the same
**kNN: Selection of Distance**

- So far we assumed we use Euclidian Distance to find the nearest neighbor:
  \[ D(a, b) = \sqrt{\sum_k (a_k - b_k)^2} \]

- However some features (dimensions) may be much more discriminative than other features (dimensions)

- Euclidean distance treats each feature as equally important
kNN: Extreme Example of Distance Selection

- decision boundaries for blue and green classes are in red
- These boundaries are really bad because
  - feature 1 is discriminative, but it’s scale is small
  - feature 2 gives no class information (noise) but its scale is large
Extreme Example
- feature 1 gives the correct class: 1 or 2
- feature 2 gives irrelevant number from 100 to 200

Suppose we have to find the class of \( x = [1 \ 100] \) and we have 2 samples \([1 \ 150]\) and \([2 \ 110]\)

\[
D([1\ 100], [1\ 150]) = \sqrt{(1-1)^2 + (100-150)^2} = 50 \\
D([1\ 100], [2\ 110]) = \sqrt{(1-2)^2 + (100-110)^2} = 10.5
\]

\( x = [1 \ 100] \) is misclassified!

The denser the samples, the less of the problem
- But we rarely have samples dense enough
Notice the 2 features are on different scales:
- feature 1 takes values between 1 or 2
- feature 2 takes values between 100 to 200

We could normalize each feature to be between of mean 0 and variance 1
If $X$ is a random variable of mean $\mu$ and variance $\sigma^2$, then $(X - \mu)/\sigma$ has mean 0 and variance 1
Thus for each feature vector $x_i$, compute its sample mean and variance, and let the new feature be $[x_i - \text{mean}(x_i)]/\sqrt{\text{var}(x_i)}$
Let’s do it in the previous example
The decision boundary (in red) is very good now!
kNN: Selection of Distance

- However in high dimensions if there are a lot of irrelevant features, normalization will not help

\[ D(a, b) = \sqrt{\sum_{k} (a_k - b_k)^2} = \sqrt{\sum_{i} (a_i - b_i)^2 + \sum_{j} (a_j - b_j)^2} \]

- If the number of discriminative features is smaller than the number of noisy features, Euclidean distance is dominated by noise.
kNN: Feature Weighting

- Scale each feature by its importance for classification

\[ D(a, b) = \sqrt{\sum_k w_k (a_k - b_k)^2} \]

- Can learn the weights \( w_k \) from the training data
  - Increase/decrease weights until classification improves
**kNN Summary**

- **Advantages**
  - Can be applied to the data from any distribution
  - Very simple and intuitive
  - Good classification if the number of samples is large enough

- **Disadvantages**
  - Choosing best $k$ may be difficult
  - Computationally heavy, but improvements possible
  - Need large number of samples for accuracy
    - Can never fix this without assuming parametric distribution