Probability Based Classification

The First Real Dataset: Iris

This is the most used data set in the pattern recognition community.

Labeled data, $M = 3$ (number of classes)

Iris Setosa

Iris Versicolor

Iris Virginica

$l = 4$ (number of features or dimensions)

sepal length in cm

sepal width in cm

petal length in cm

petal width in cm

Feature Vectors

The samples (N = 150, 50 for each class):

Iris sestosa				Iris versicolor				Iris virginica			
Sepal Leng.	Sepal Width	Petal Leng.	Petal Width	Sepal Leng.	Sepal Width	Petal Leng.	Petal Width	Sepal Leng.	Sepal Width	Petal Leng.	Petal Width
5.1	3.5	1.4	0.2	7.0	3.2	4.7	1.4	6.3	3.3	6.0	2.5
4.9	3.0	1.4	0.2	6.4	3.2	4.5	1.5	5.8	2.7	5.1	1.9
4.7	3.2	1.3	0.2	6.9	3.1	4.9	1.5	7.1	3.0	5.9	2.1
4.6	3.1	1.5	0.2	5.5	2.3	4.0	1.3	6.3	2.9	5.6	1.8
5.0	3.6	1.4	0.2	6.5	2.8	4.6	1.5	6.5	3.0	5.8	2.2

and more …

Typically, each sample is represented by a **feature vector**, where each element is a feature.

A feature vector is usually denoted by the symbol $\boldsymbol{x}$, which is a column vector. Example: $[\ 5.1 \quad 3.5 \quad 1.4 \quad 0.2\ ]^{\mathrm{T}}$

Each feature vector is a point in the **feature space**, whose number of dimensions is the number of features.

Classification

Now, let us try to classify using numerical features.
Consider only the training data shown here:

Iris sestosa				Iris versicolor				Iris virginica			
Sepal Leng.	Sepal Width	Petal Leng.	Petal Width	Sepal Leng.	Sepal Width	Petal Leng.	Petal Width	Sepal Leng.	Sepal Width	Petal Leng.	Petal Width
5.1	3.5	1.4	0.2	7.0	3.2	4.7	1.4	6.3	3.3	6.0	2.5
4.9	3.0	1.4	0.2	6.4	3.2	4.5	1.5	5.8	2.7	5.1	1.9
4.7	3.2	1.3	0.2	6.9	3.1	4.9	1.5	7.1	3.0	5.9	2.1
4.6	3.1	1.5	0.2	5.5	2.3	4.0	1.3	6.3	2.9	5.6	1.8
5.0	3.6	1.4	0.2	6.5	2.8	4.6	1.5	6.5	3.0	5.8	2.2
5.4	3.9	1.7	0.4	5.7	2.8	4.5	1.3	7.6	3.0	6.6	2.1
4.6	3.4	1.4	0.3	6.3	3.3	4.7	1.6	4.9	2.5	4.5	1.7
5.0	3.4	1.5	0.2	4.9	2.4	3.3	1.0	7.3	2.9	6.3	1.8
4.4	2.9	1.4	0.2	6.6	2.9	4.6	1.3	6.7	2.5	5.8	1.8
4.9	3.1	1.5	0.1	5.2	2.7	3.9	1.4	7.2	3.6	6.1	2.5

try to classify these feature vectors:

$[\ 6.9 \quad 3.1 \quad 5.4 \quad 2.1\ ]^T$

$[\ 6.7 \quad 3.0 \quad 5.0 \quad 1.7\ ]^T$

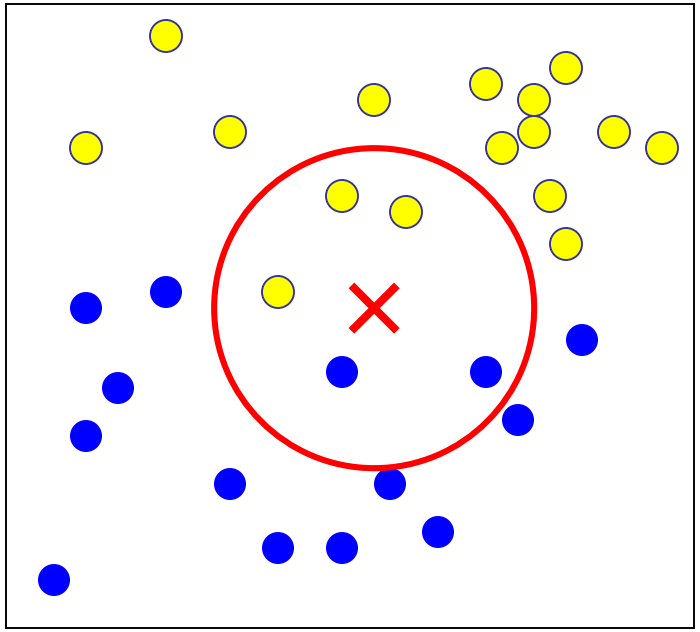
$[\ 5.0 \quad 3.5 \quad 1.6 \quad 0.6\ ]^T$

And consider how you make the decisions.

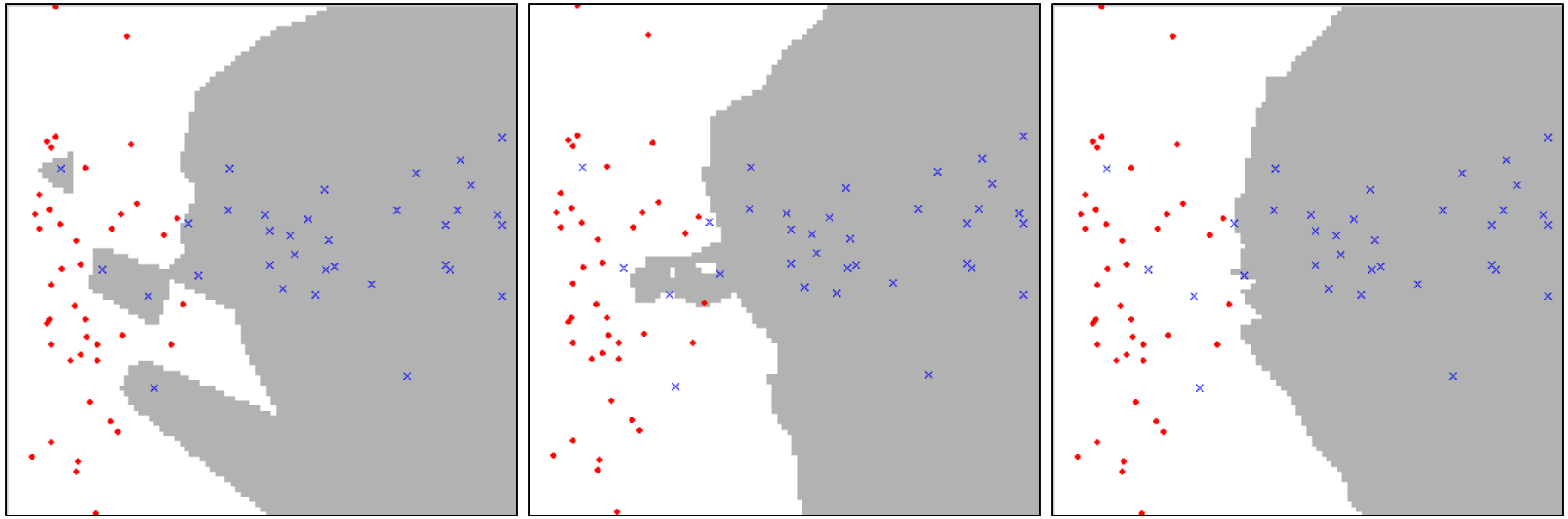
kNN Rule of Classification

Our first classifier: ***k*-nearest-neighbor** (kNN)

- For a given $\boldsymbol{x}$ to be classified, choose the k training samples that are closest to $\boldsymbol{x}$.
- Within these k samples, count each k_i, the number of samples in class ω_i .
- Assign $\boldsymbol{x}$ to the class with the largest k_i.
- We need a distance metric between samples.
- In general, k is NOT a multiple of M.
- Pro: Conceptually simple
- Con: Computationally expensive



kNN Classification - Example



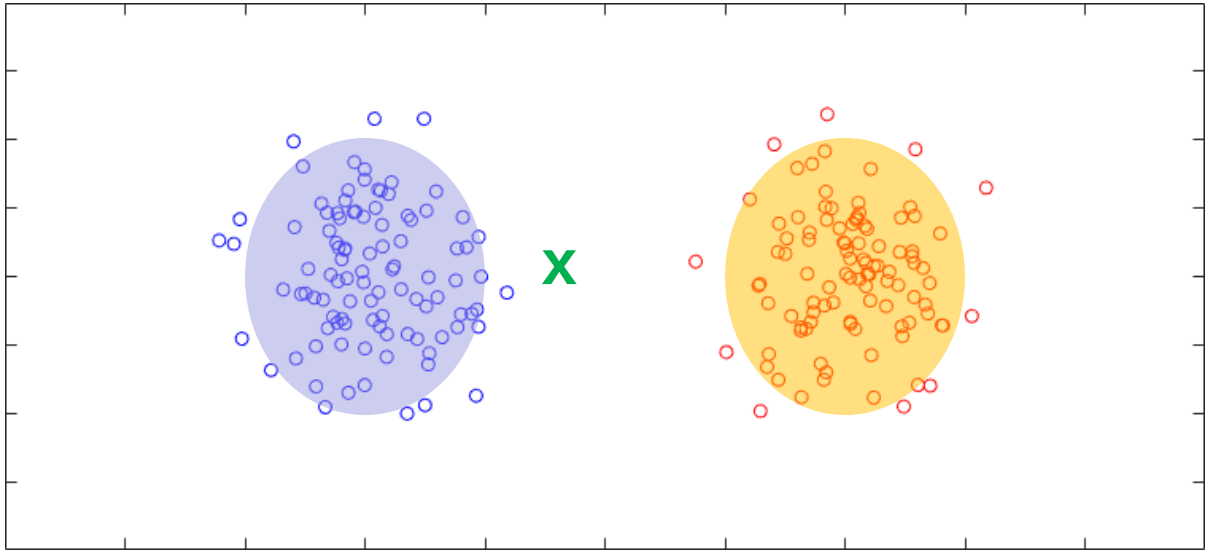
$k = 1$ $k = 3$ $k = 9$

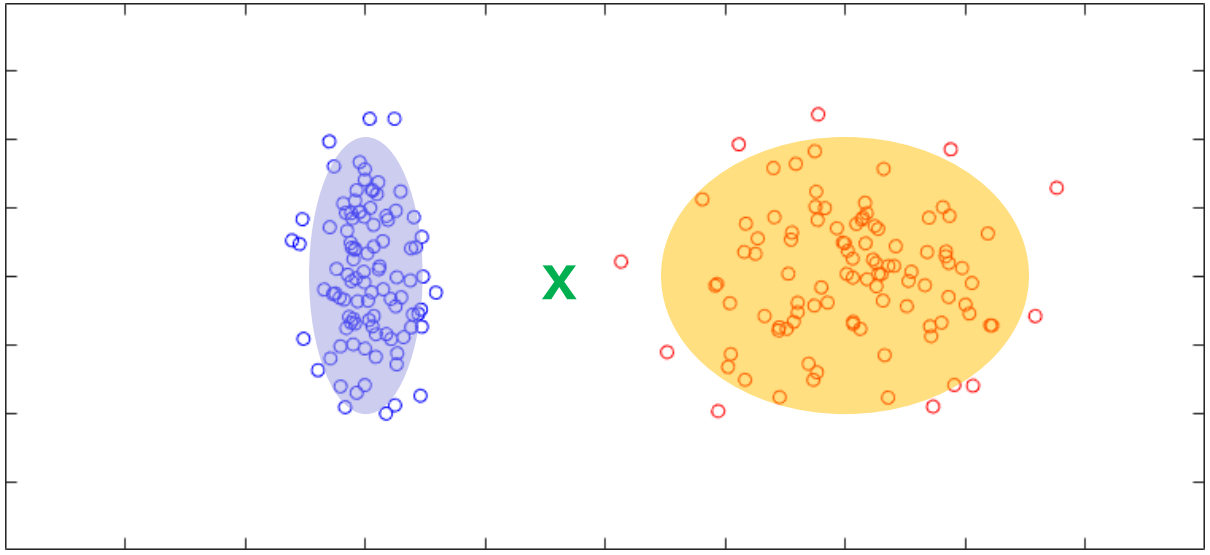
The red and blue dots are the training samples of the two classes.

The colors (white/gray) represent the **decision regions** of the two classes.

Classification and Data Distribution

How will you classify the given point **x**?





- The concept of ***Bayesian classifiers***: Classify a pattern to the most probable class.
- Internal representation: probabilities and probability distributions (in the feature space) of the classes
  - Example of a generative classifier
- Task of learning: Building the internal representation from the training data and domain knowledge, if available.

Bayesian Classifiers: Notations and Terminologies

$\boldsymbol{x}$: feature vector

ω_1, ω_2, ... : classes

Assumed known, or estimated from training data:

$P(\omega_1)$, $P(\omega_2)$, ... : ***a priori probabilities*** (sum to one)

$p(\boldsymbol{x}|\omega_1)$, $p(\boldsymbol{x}|\omega_2)$, ... : class-conditional probabilities

also called: ***likelihood function*** or

pdf (probability density function)

Each has a total area of one.

Used for classification:

$P(\omega_1|\boldsymbol{x})$, $P(\omega_2|\boldsymbol{x})$, ... : ***a posteriori probabilities***

Bayesian Decision Theory

We are only concerned with the task of classification here.
The goal: To find the most probable class for a given $\boldsymbol{x}$.

Assumed known:

$P(\omega_1)$, $P(\omega_2)$, ... and $p(\boldsymbol{x}|\omega_1)$, $p(\boldsymbol{x}|\omega_2)$, ...

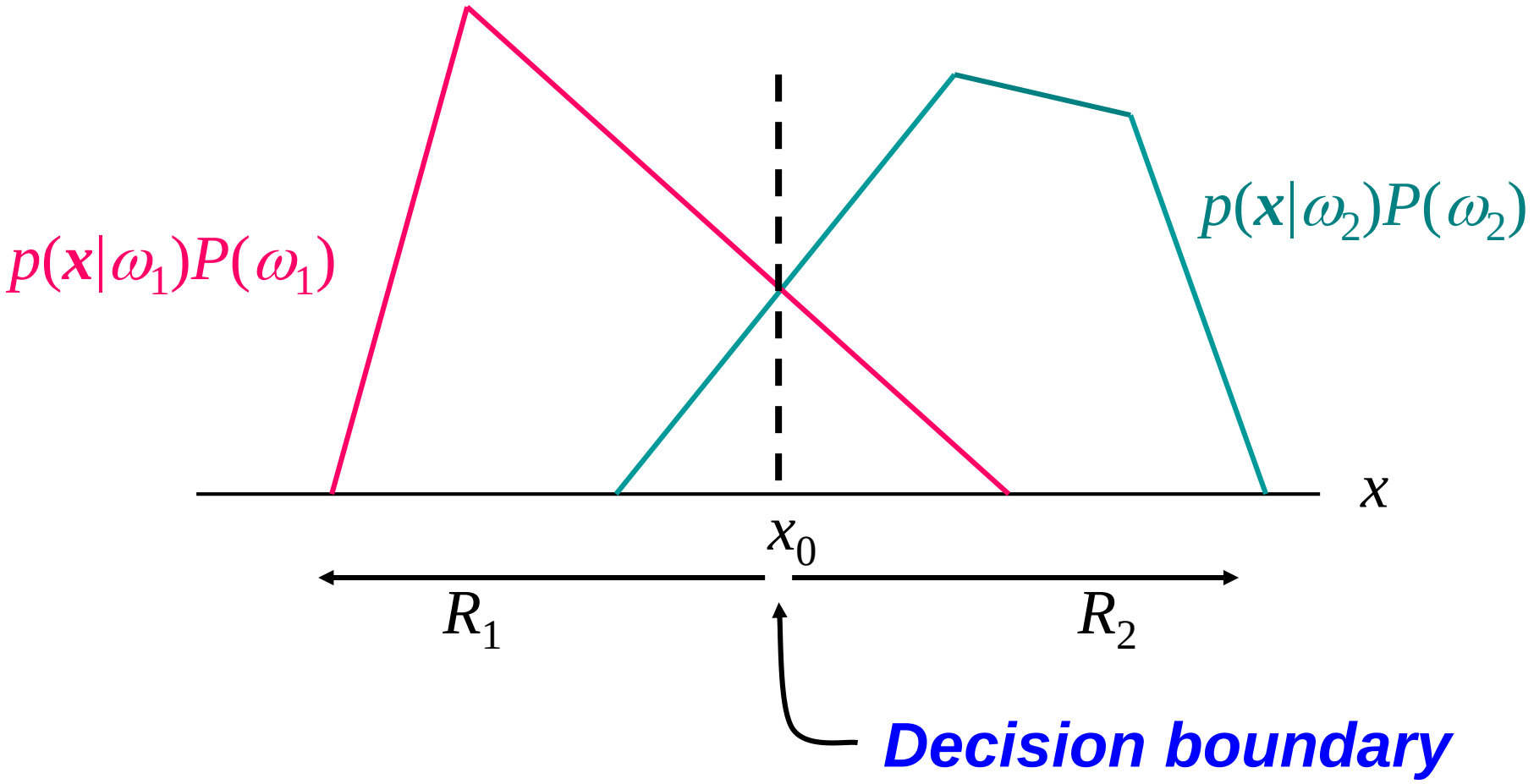
Method: Assign $\boldsymbol{x}$ to the class ω_i that gives the largest $P(\omega_i|\boldsymbol{x})$.

Bayes Rule:

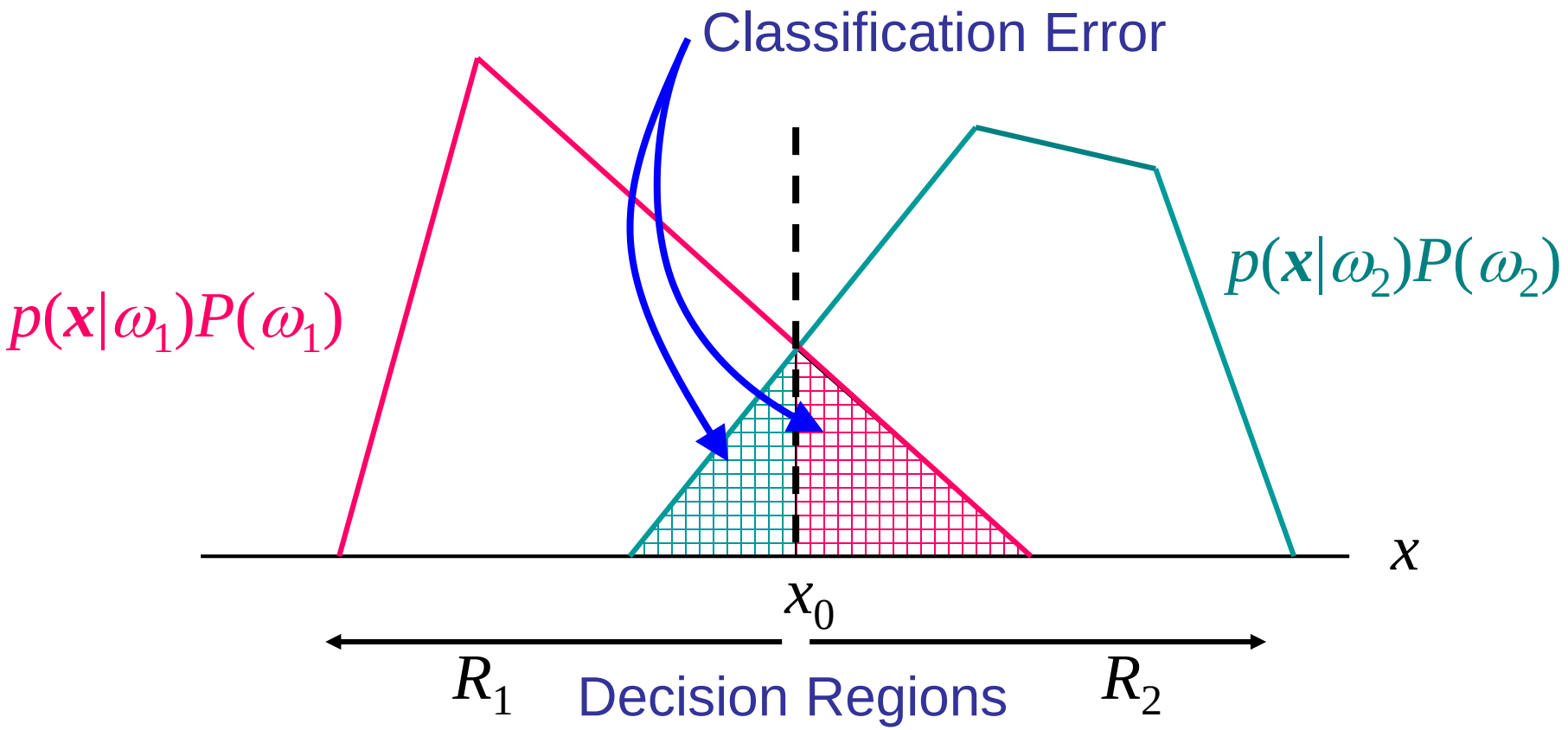
$$p(\boldsymbol{x} \mid \omega_i)P(\omega_i) = P(\omega_i \mid \boldsymbol{x})\,p(\boldsymbol{x})$$

Since $p(\boldsymbol{x})$ Method is independent of ω_i , the decision can be made by comparing $P(\omega_i|\boldsymbol{x})P(\omega_i)$.

One-dimensional View

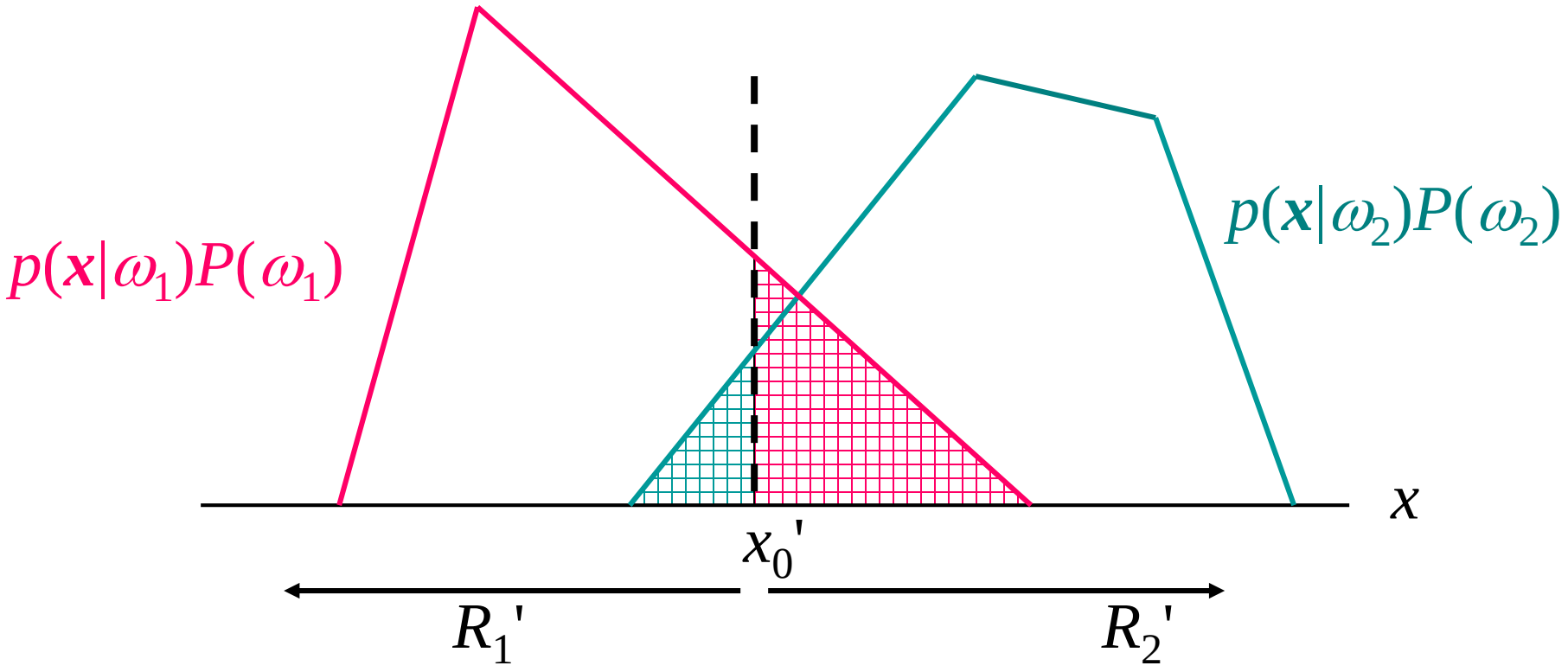


Probability of Classification Error



$$P_e = P(\omega_2)\int_{R_1} p(\boldsymbol{x}\,|\,\omega_2)d\boldsymbol{x} + P(\omega_1)\int_{R_2} p(\boldsymbol{x}\,|\,\omega_1)d\boldsymbol{x}$$

Optimality of Bayesian Classifier



$$P_e' = P(\omega_2)\int_{R_1'} p(\boldsymbol{x}\,|\,\omega_2)d\boldsymbol{x} + P(\omega_1)\int_{R_2'} p(\boldsymbol{x}\,|\,\omega_1)d\boldsymbol{x}$$

Not All Errors are Equal

Example: Traffic light recognition

What is the risk (expected loss)

(1) if a green light is misclassified as a red light?

(2) if a red light is misclassified as a green light?

Some classification errors are more costly than others.

Example: Medical screening tests

Usually the probability for the "positive" class is very low.

For example: $P(\omega_1)$=0.99 (negative) and $P(\omega_2)$=0.01 (positive).

P_e is only 0.01 if we just call every test result negative. But …

We don't want such trivial solutions.

Minimum-Risk Classifier

λ_{ki} : the ***loss*** if a sample actually in ω_k is classified as ω_i.

The ***loss matrix*** L consists of All the λ_{ki}.

Two-class cases:

The risk of classifying $\boldsymbol{x}$ as ω_1 is

$$r_1 = \lambda_{11} p(\boldsymbol{x} \mid \omega_1) P(\omega_1) + \lambda_{21} p(\boldsymbol{x} \mid \omega_2) P(\omega_2)$$

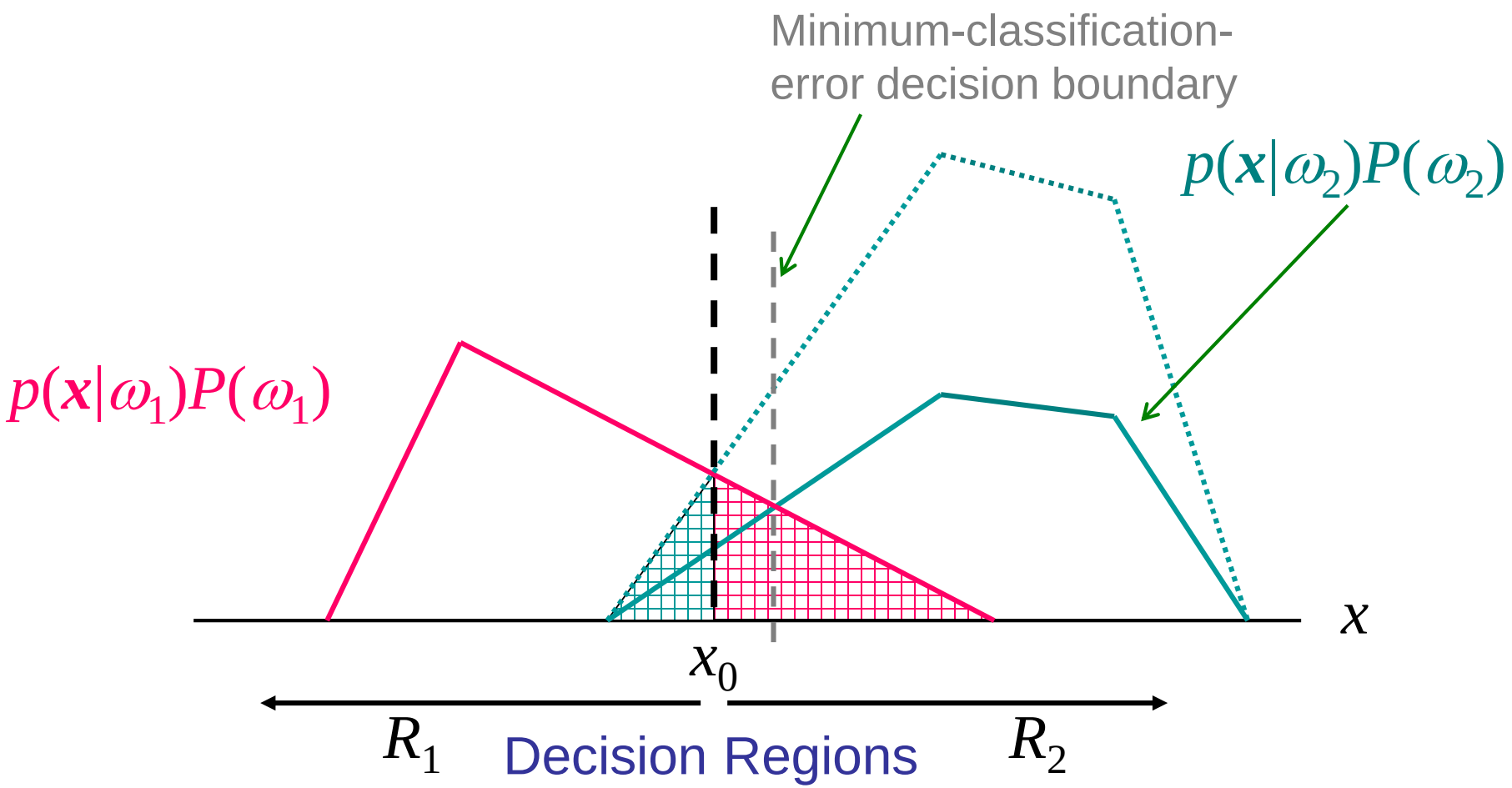
The risk of classifying $\mathbf{x}$ as ω_2 is

$$r_2 = \lambda_{12} p(\boldsymbol{x} \mid \omega_1) P(\omega_1) + \lambda_{22} p(\boldsymbol{x} \mid \omega_2) P(\omega_2)$$

A ***minimum-risk classifier*** classifies $\boldsymbol{x}$ to ω_1 if $r_1 < r_2$, and vice versa.

Minimum-Risk Classifier

Example: $\lambda_{11} = \lambda_{22} = 0$, $\lambda_{12} = 1$, $\lambda_{21} = 2$



Representing PDF

To classify a given $\boldsymbol{x}$, we need to know the **probability density function** (pdf) of each class at $\boldsymbol{x}$, that is, $p(\boldsymbol{x}|\omega_i)$.

Now, how do we represent the pdf?

Parametric methods:

- Assume a known functional form for pdf that can be defined with a few parameters.
- We will have detailed discussion of the most commonly used function form: a Gaussian (normal distribution).

Nonparametric methods:

- There is no assumed functional form; the value of the pdf at each $\boldsymbol{x}$ has to be estimated locally.

Normal Distributions

1-d Gaussian pdf:

$$p(x \mid \omega_i) = \frac{1}{\sqrt{2\pi}\ \sigma_i} \exp\left(-\frac{1}{2\sigma_i^2}(x-\mu_i)^2 \right)$$

Multi-dimensional Gaussian pdf:

$$p(\boldsymbol{x} \mid \omega_i) = \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^l |\Sigma_i|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_i) \right)$$

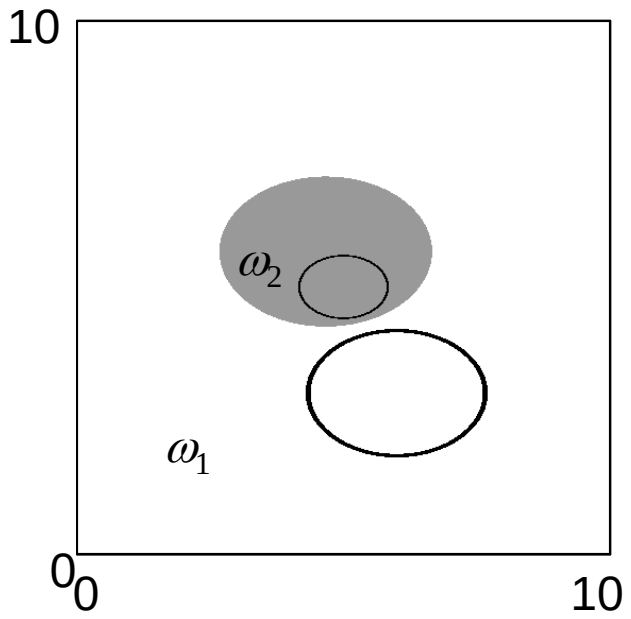
mean: $\boldsymbol{\mu}_i = E[\boldsymbol{x}]$

covariance matrix: $\Sigma_i = E\left[(\boldsymbol{x}-\boldsymbol{\mu}_i)(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \right]$

for $\boldsymbol{x} \in \omega_i$

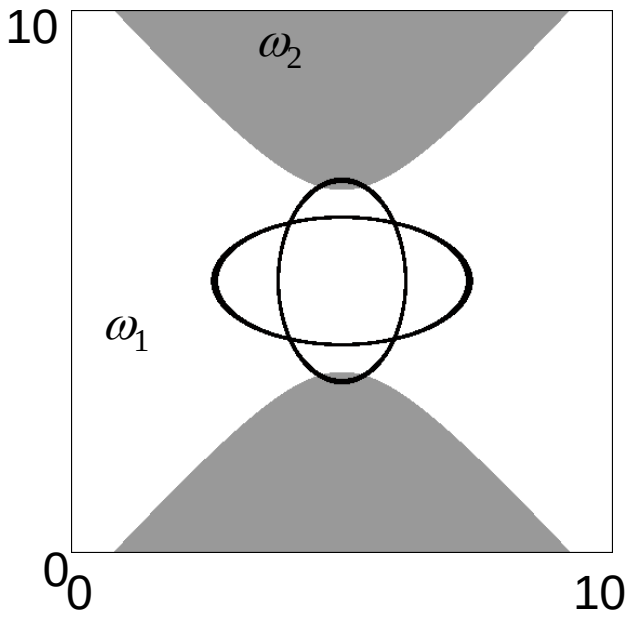
Decision Boundaries for Normal Distributions

Decision boundaries between classes with normal distribution pdfs are quadratic (linear if equal covariance).



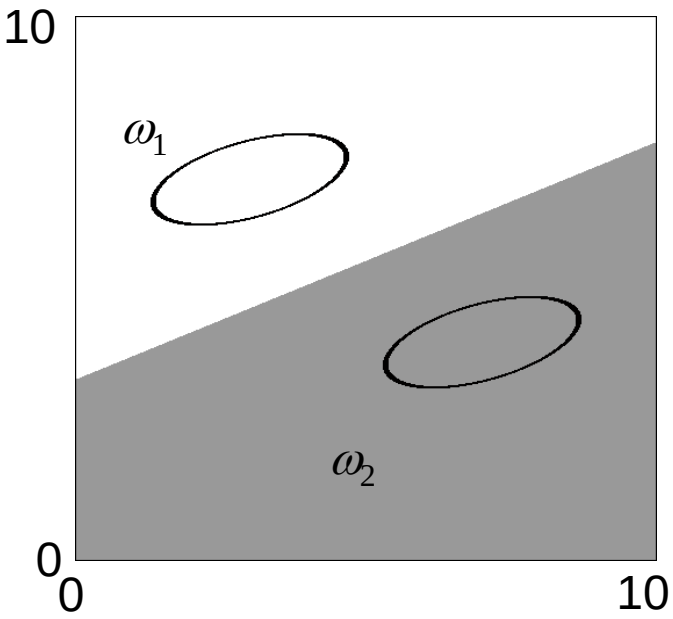
$$\boldsymbol{\mu}_1 = \begin{bmatrix} 6 \\ 3 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \quad P(\omega_1) = 0.5$$

$$\boldsymbol{\mu}_2 = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix} \quad P(\omega_2) = 0.5$$



$$\boldsymbol{\mu}_1 = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} \quad P(\omega_1) = 0.7$$

$$\boldsymbol{\mu}_2 = \begin{bmatrix} 5 \\ 5 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix} \quad P(\omega_2) = 0.3$$



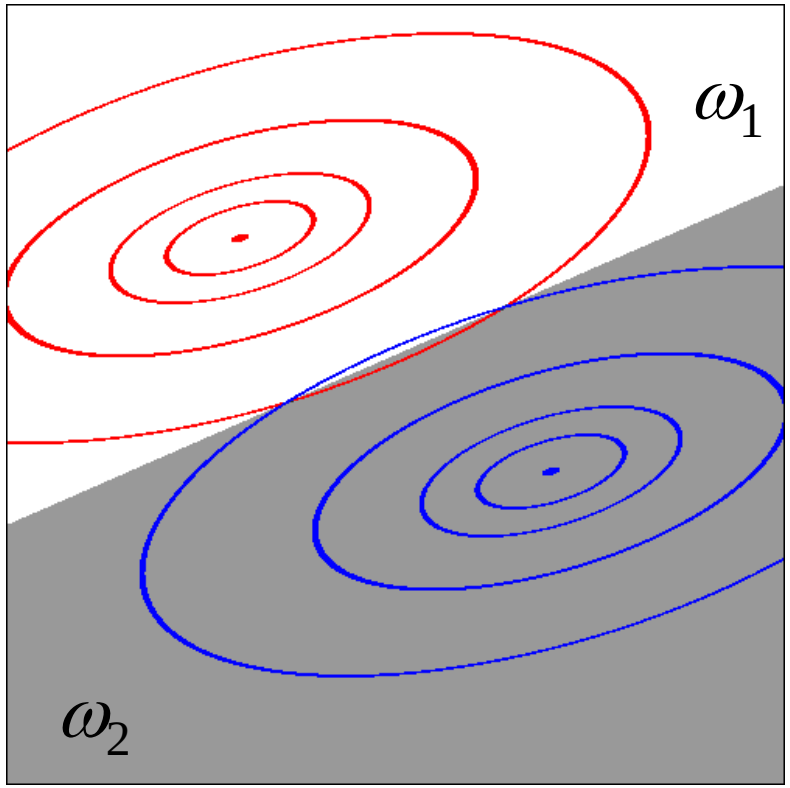
$$\boldsymbol{\mu}_1 = \begin{bmatrix} 3 \\ 7 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 4 & 1 \\ 1 & 1 \end{bmatrix} \quad P(\omega_1) = 0.5$$

$$\boldsymbol{\mu}_2 = \begin{bmatrix} 7 \\ 4 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 4 & 1 \\ 1 & 1 \end{bmatrix} \quad P(\omega_2) = 0.5$$

Mahalanobis Distance

$$d_m = \left[(\boldsymbol{x} - \boldsymbol{\mu}_i)^T \, \Sigma_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i) \right]^{1/2}$$

The **Mahalanobis distance** is a measure of how far $\boldsymbol{x}$ is to a normal distribution. It can be used for classification if we assume equal a priori probabilities for all classes.



d_m shown:
(from inside out)

- –ln(0.8)
- –ln(0.5)
- –ln(0.1)
- –ln(0.001)

Maximum Likelihood Estimation

- This is the most common method for estimating the parameters for a pdf with a parametric form; we commonly use a "vector" $\boldsymbol{\theta}$ to represent all the (unknown) parameters of the pdf.
- The pdf, which is depending on $\boldsymbol{\theta}$, is usually expressed as $p(\boldsymbol{x};\boldsymbol{\theta})$ or $p(\boldsymbol{x}|\omega_i;\boldsymbol{\theta})$.
- We estimate $\boldsymbol{\theta}$ by maximizing the overall likelihood of observing the samples (assuming the samples are statistically independent):

$$\hat{\boldsymbol{\theta}}_{ML} = \arg\max_{\boldsymbol{\theta}} \prod_{k=1}^{N} p(\boldsymbol{x}_k;\boldsymbol{\theta})$$

- We commonly solve for $\boldsymbol{\theta}$ by setting the derivative of the **loglikelihood** with respect to $\boldsymbol{\theta}$ to zero:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \ln\left[\prod_{k=1}^{N} p(\boldsymbol{x}_k;\boldsymbol{\theta})\right] = 0$$

ML for Gaussian

Now, what happens if we assume a Gaussian distribution? The parameters in $\boldsymbol{\theta}$ now include the mean, $\boldsymbol{\mu}$, and the covariance matrix, $\boldsymbol{\Sigma}$.

Results:

$$\hat{\boldsymbol{\mu}}_{ML} = \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{x}_k \qquad \hat{\Sigma}_{ML} = \frac{1}{N}\sum_{k=1}^{N} (\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})^T$$

We will skip the proof here.

The ML estimation of Σ is biased:

$$\hat{\Sigma}_{ML} < E\left[(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})^T\right] = \frac{1}{N-1}\sum_{k=1}^{N} (\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_k - \hat{\boldsymbol{\mu}})^T$$

An Easier ML Example

Consider the following 1-D distribution (assuming all $x>0$) with a scalar θ:

$$p(x) = \theta \exp(-x\theta)$$

What is the ML estimation of θ given the samples $\{x_1, x_2, \ldots, x_N\}$?

Loglikelihood: $$L(\theta) = \ln\left[\prod_{k=1}^{N} p(\boldsymbol{x}_k;\theta)\right] = \sum_{k=1}^{N}\left(\ln\theta - x_k\theta\right) = N\ln\theta - \theta\sum_{k=1}^{N} x_k$$

$$\frac{\partial L(\theta)}{\partial\theta} = 0 = \frac{N}{\theta} - \sum_{k=1}^{N} x_k \quad \Rightarrow \quad \theta = \frac{N}{\sum_{k=1}^{N} x_k}$$

Maximum A Posteriori Estimation

- Maximum likelihood estimation of pdf parameters does not consider any knowledge about the parameter values.
- **Maximum a posteriori (MAP) estimation** is a modification of maximum likelihood estimation when the (a priori) probability distribution of the parameters, $P(\boldsymbol{\theta})$, is given.
- The goal is to find the parameters $\boldsymbol{\theta}$ to maximize

$$P(\boldsymbol{\theta} \mid \mathbf{x}) \propto p(\mathbf{x} \mid \boldsymbol{\theta}) P(\boldsymbol{\theta})$$

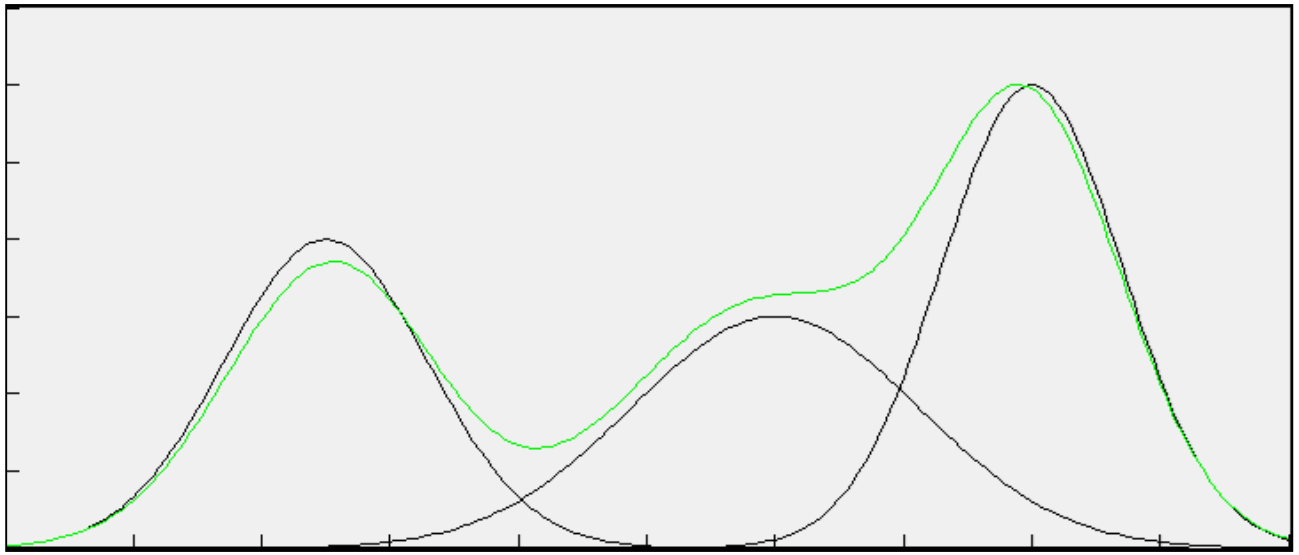
- When some information about the parameter values is available, this helps the estimation focus more on "reasonable" results.
- When no such information is available, $P(\boldsymbol{\theta})$ is assumed to be uniform, and MAP reduces to ML estimation.

Mixture Models

- Motivation: How can we extend parametric pdf models to represent more general distributions?
- Idea: Represent a distribution (no known functional form) with a linear combination of other distributions of known functional forms:

$$p(\boldsymbol{x}) = \sum_{j=1}^{J} p(\boldsymbol{x} \mid j) P_j$$

- Example: Modeling a 1-D distribution with mixture of Gaussians:



Mixture Model

Strength of the mixture model:

- With enough components and an appropriate component model such as Gaussian, it is possible to approximate any continuous distribution.

Difficulties of the mixture model:

- We do not know the mixture component label of $\boldsymbol{x}$.
- We do not know the weights (P_j) of the components.
- We do not know the number of components.

The standard algorithm to build a mixture model is the **EM (expectation-maximization) algorithm**. Since it's more difficult to understand, we will leave the detail until we talk about clustering.

Nonparametric pdf Estimation

- Assume no functional form of the pdf.
- Instead, the estimation is based on local data distribution:

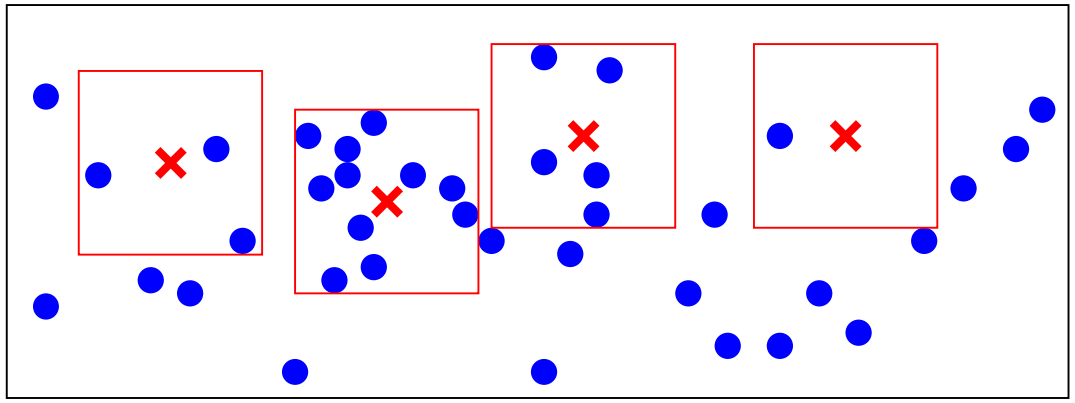
$$p(\boldsymbol{x}) \approx \frac{k}{N V(\boldsymbol{x})}$$

  - $V(\boldsymbol{x})$: a volume around $\boldsymbol{x}$
  - k : # samples inside $V(\boldsymbol{x})$
  - N : # total samples
- $p(\boldsymbol{x})$ and k only applies to a given class if we want to estimate the class-conditional pdf.
- Two approaches:
  - Fix $V(\boldsymbol{x})$: **Parzen window**
  - Fix k: **k-nearest-neighbor**

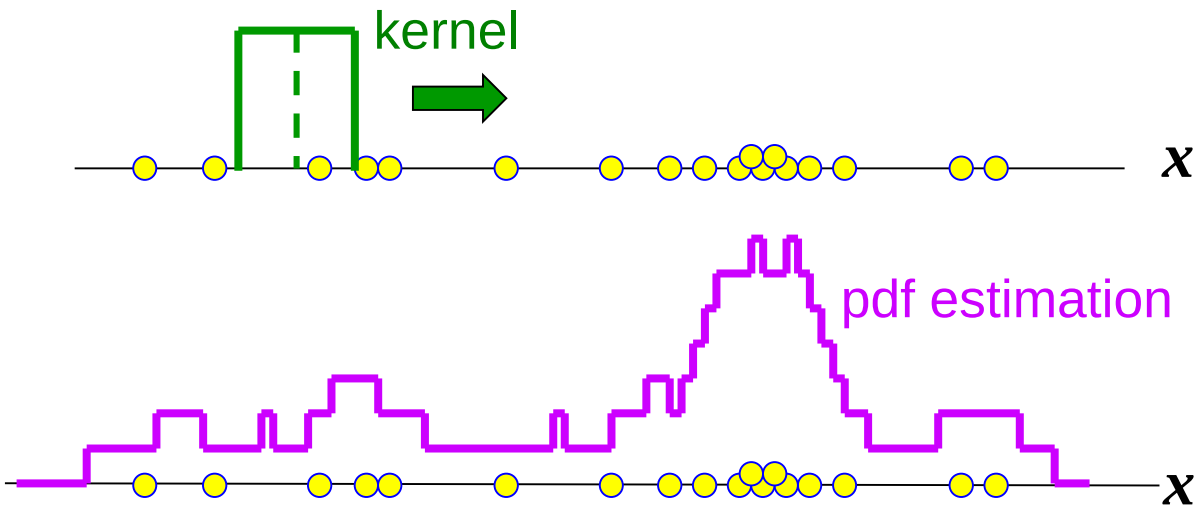
Parzen Window

To estimate $p(\boldsymbol{x})$ using $k/NV(\boldsymbol{x})$:

Fix $V(\boldsymbol{x})$ and count k within $V(\boldsymbol{x})$



An 1-D example with a box kernel:



Kernel function: $p(\boldsymbol{x}) = \frac{1}{h^l}\left[\frac{1}{N}\sum_{i=1}^{N}\phi\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h}\right)\right]$ with $\phi(\boldsymbol{x}) \geq 0$ and $\int_x \phi(\boldsymbol{x})d\boldsymbol{x} = 1$

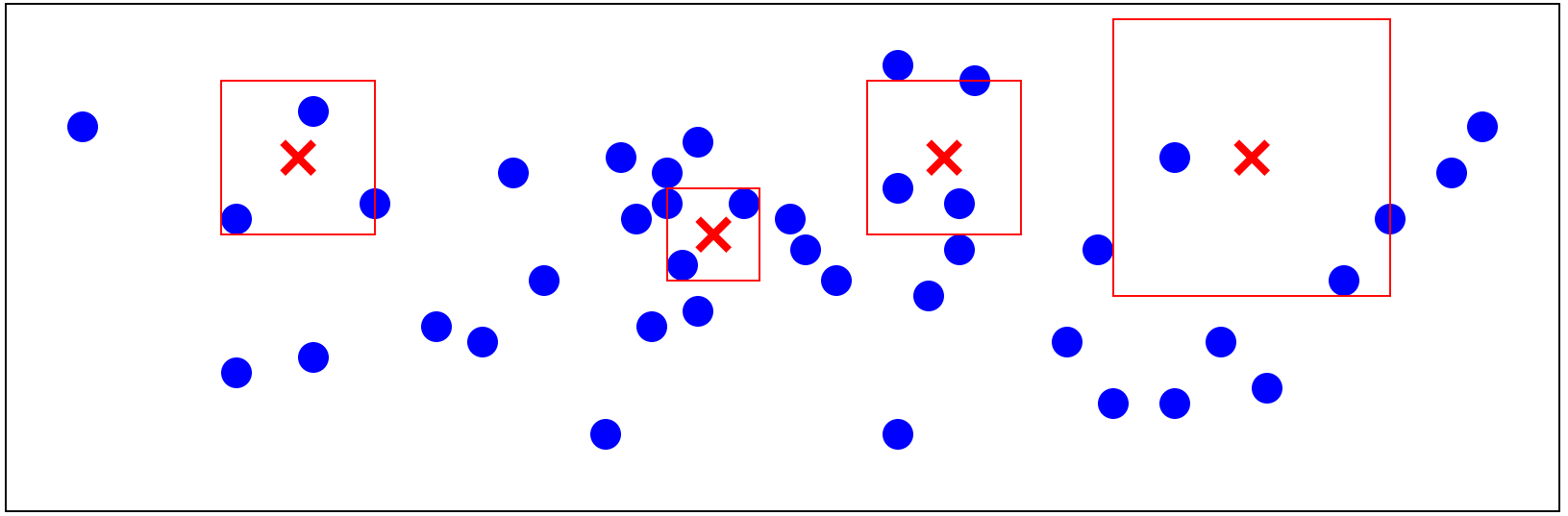
The most common kernel functions are Gaussians and hypercubes (box functions).

k-Nearest Neighbor (kNN)

To estimate $p(\boldsymbol{x})$ using $k/NV(\boldsymbol{x})$:

Fix k and find the smallest $V(\boldsymbol{x})$ that contains k samples.

Example: 3NN with hypercubes for $V(\boldsymbol{x})$.



Curse of Dimensionality

- The question: How many samples do we need for reliable pdf estimation or kNN classification?
- Each feature is a dimension of the feature space. The more features we have, the more information we retain for the classification task (which seems better).
- Problem of high dimensionality: The number of samples needed grow exponentially with the dimension.

Naïve-Bayes Classifier

- One way to deal with the curse of dimensionality.
- Assume that the features are independent.
- A l-D problem becomes l 1-D problems.
- The pdf becomes the

$$p(\boldsymbol{x} \mid \omega_i) = \prod_{j=1}^{l} p(x_j \mid \omega_i)$$

- Each one-dimensional pdf is estimated separately.
- The exponential growth (w.r.t. number of features) of samples needed is reduced to linear growth.
- This has been found to work surprisingly well for many complex problems, even with the oversimplified assumption.