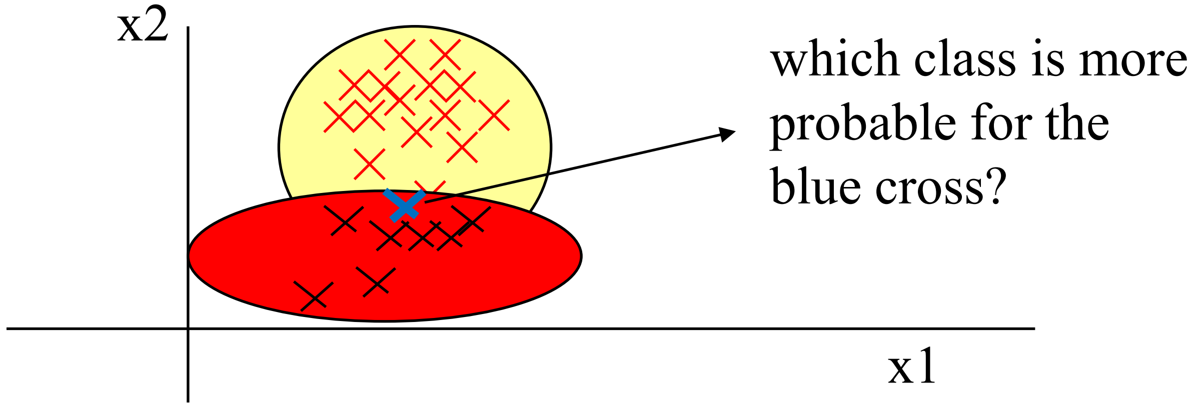
Lecture 2
Bayesian Classifiers

Statistics of features

- ➔ **For a given reference class, rather than providing a set of exemplary vectors, we have a statistical description of features**
  - ⇨ E.g. the red class has (x1, x2) distributed in the shaded area, with 90% of the points in the highest semicircle



- ➔ **How we do train a classifier? By generating exemplary points? Can we do more?**

Training the classifier

- ➔ **Learning about p($\underline{x}/\omega_i$) observing examples of each class ω_i**
- ➔ **Learning about class occurrence probability p(ω_i)**

- ➔ **If we know these probability functions, we can classify each vector $\underline{x}$ as belonging to:**
  - ⇨ ω_i : $Pr(\omega_i / \underline{x}) = p(x/ \omega_i)p(x)/p(\omega_i)$ is maximized

An intuitive example
before formal definition

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

From data to probability

Features: x1=outlook, x2=temperature, x3=humidity, x4=wind

Class: ω1=Play, ω2=No Play

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

	$p(x2/\omega1)$	$p(x2/\omega2)$
Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

p(ω1)= *P*(Play=*Yes*) = 9/14

p(ω2)= *P*(Play=*No*) = 5/14

From probability to Classification

Test Phase

- **Given a new instance, predict its label**

  $\mathbf{x}'$=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)

- **Look up tables achieved in the learning phrase**

P(Outlook=*Sunny*|Play=*Yes*) = 2/9	P(Outlook=S*unny*|Play=*No*) = 3/5
P(Temperature=*Cool*|Play=*Yes*) = 3/9	P(Temperature=*Cool*|Play==*No*) = 1/5
P(Huminity=*High*|Play=*Yes*) = 3/9	P(Huminity=*High*|Play=*No*) = 4/5
P(Wind=Strong|Play=Yes) = 3/9	P(Wind=*Strong*|Play=*No*) = 3/5
P(Play=*Yes*) = 9/14	P(Play=*No*) = 5/14

- **Decision making with the Bayesian rule**

$$P(Yes|\mathbf{x}') \approx [P(Sunny|Yes)P(Cool|Yes)P(High|Yes)P(Strong|Yes)]P(Play=Yes) = 0.0053$$
$$P(No|\mathbf{x}') \approx [P(Sunny|No)\, P(Cool|No)P(High|No)P(Strong|No)]P(Play=No) = 0.0206$$

Given the fact P(*Yes*|$\mathbf{x}'$) < P(*No*|$\mathbf{x}'$), we label $\mathbf{x}'$ to be "*No*".

Problems to be solved

- **How to define class belonging criterion?**
  - Just simple a maximum a posteriori (MAP)?
  - What about non uniform error risks?
  - Is it possible to map the a posteriori pr maximization into a region boundary in the feature space?
- **How to estimate discrete and continous feature probability?**
  - Example of continuous features: lenght of a flower petal to recognize the specie

Formal definition of a Bayesian Classifier

assuming probabilities are known

Classifier Definition

➔ **Statistical nature of feature vectors**

$$\underline{x} = \left[x_1, x_2, \ldots, x_l\right]^T$$

➔ **Class separation could be not deterministic**

➔ **Assign the pattern represented by feature vector $\underline{x}$ to the most probable of the available classes**

$$\omega_1, \omega_2, \ldots, \omega_M$$

That is $\underline{x} \rightarrow \omega_i : P(\omega_i | \underline{x})$ **maximum**

Maximum likelihood

➔Computation of a-posteriori probabilities

- ⇨ Assume known
  - →a-priori probabilities of class occurrences

$$P(\omega_1), P(\omega_2) \ldots, P(\omega_M)$$

  - → feature statistics for a given class

$$p(\underline{x} | \omega_i), i = 1,2 \ldots M$$

This is also known as the likelihood of

$$\underline{x} \ \ w.r. \ \ to \ \ \omega_i.$$

- The Bayes rule (M=2)

$$p(\underline{x})P(\omega_i|\underline{x}) = p(\underline{x}|\omega_i)P(\omega_i) \Rightarrow$$

$$P(\omega_i / \underline{x}) = \frac{p(\underline{x}|\omega_i)P(\omega_i)}{p(\underline{x})}$$

where

$$p(\underline{x}) = \sum_{i=1}^{2} p(\underline{x}|\omega_i)P(\omega_i)$$

Bayes classification rule (M=2)

Given $\underline{x}$ classify it according to the rule

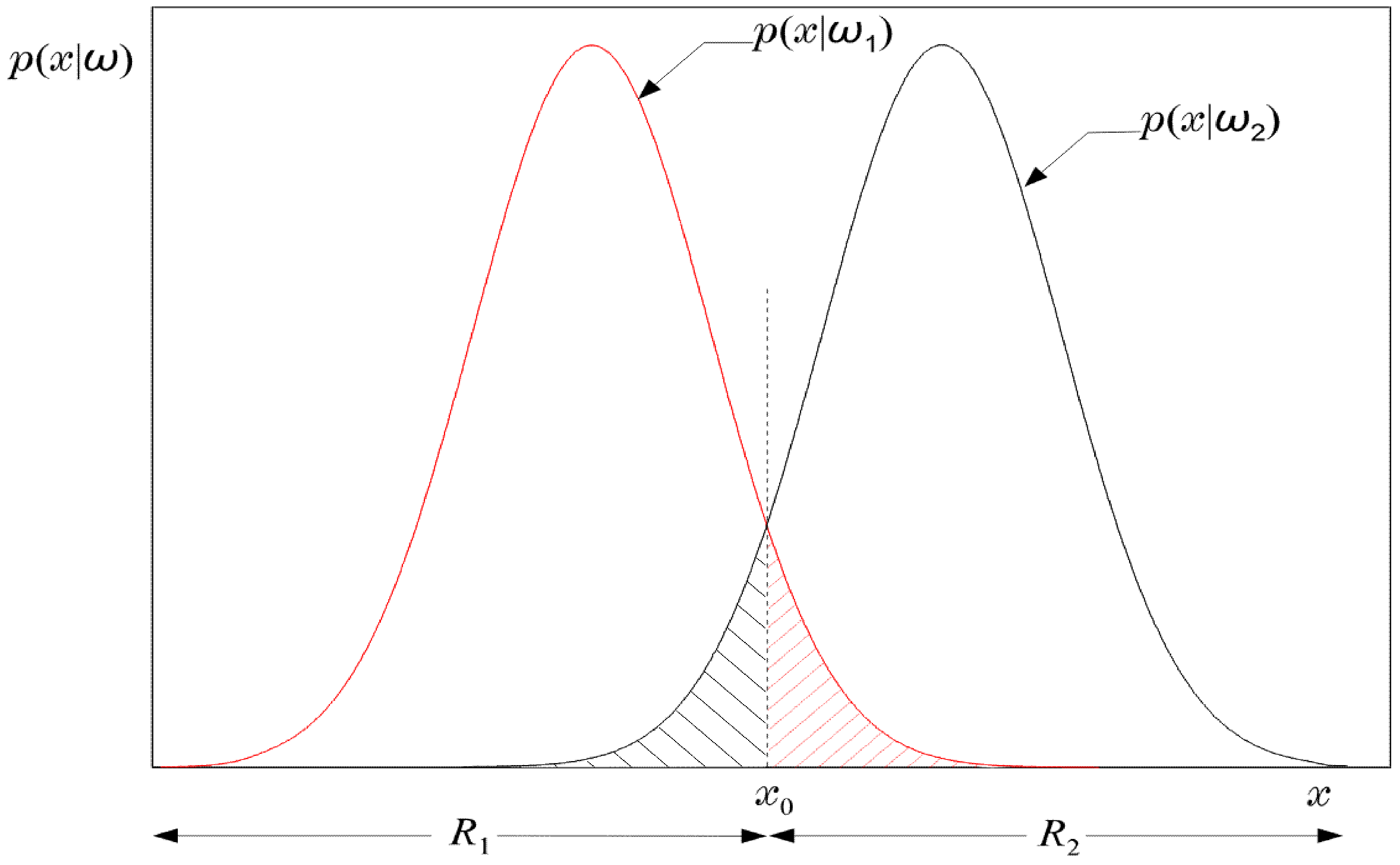
$$\boxed{\begin{array}{l} If\ P(\omega_1|\underline{x}) > P(\omega_2|\underline{x})\ \ \underline{x} \rightarrow \omega_1 \\ If\ P(\omega_2|\underline{x}) > P(\omega_1|\underline{x})\ \ \underline{x} \rightarrow \omega_2 \end{array}}$$

Equivalently: classify $\underline{x}$ according to the rule

$$\boxed{p(\underline{x}|\omega_1)P(\omega_1)(><)p(\underline{x}|\omega_2)P(\omega_2)}$$

For equiprobable classes the test becomes

$$\boxed{p(\underline{x}|\omega_1)(><)P(\underline{x}|\omega_2)}$$



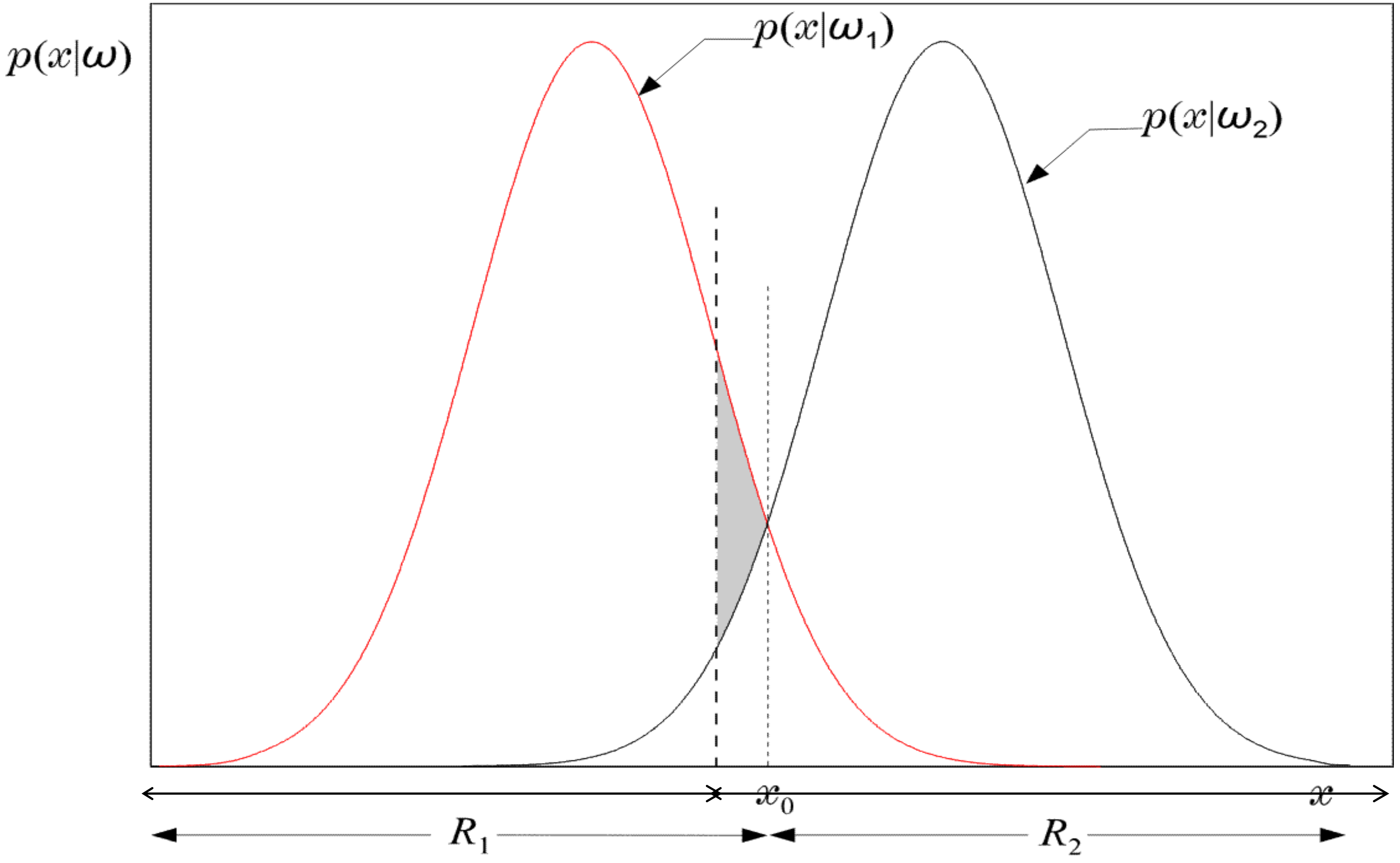
$$R_1(\rightarrow \omega_1) \ and \ R_2(\rightarrow \omega_2)$$

- ➔ **Equivalently in words: Divide space in two regions**

$$\boxed{\begin{array}{l}\text{If } \underline{x} \in R_1 \Rightarrow \underline{x} \text{ in } \omega_1 \\ \text{If } \underline{x} \in R_2 \Rightarrow \underline{x} \text{ in } \omega_2\end{array}}$$

- ➔ **Probability of error**
  - ⇨ Total shaded area
  - ⇨ $P_e = \int_{-\infty}^{x_0} p(x|\omega_2)dx + \int_{x_0}^{+\infty} p(x|\omega_1)dx$

- ➔ **Bayesian classifier is OPTIMAL with respect to minimising the classification error probability!!!!**



⇨ Indeed: Moving the threshold the total shaded area INCREASES by the extra “grey” area.

Multi-class generalization (M>2)

➔ **Given $\underline{x}$ classify it to ω_i if:**

$$\boxed{P(\omega_i|\underline{x}) > P(\omega_j|\underline{x}) \quad \forall j \neq i}$$

⇨ Such a choice also minimizes the classification error probability

➔ **Minimizing the average risk**

⇨ For each wrong decision, a penalty term is assigned since some decisions are more sensitive than others

⇨ For M=2

→ Define the loss matrix

$$L = \begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{pmatrix}$$

→ λ_{12} penalty term for deciding class ω_2 , although the pattern belongs to ω_1 , etc.

⇨ Risk with respect to ω_1

$$r_1 = \lambda_{11} \int_{R_1} p(\underline{x}|\omega_1) d\underline{x} + \boxed{\lambda_{12} \int_{R_2} p(\underline{x}|\omega_1) d\underline{x}}$$

⇨ Risk with respect to ω_2

$$r_2 \boxed{= \lambda_{21}\int_{R_1} p(\underline{x}|\omega_2)\,d\underline{x}} + \lambda_{22}\int_{R_2} p(\underline{x}|\omega_2)\,d\underline{x}$$

$\Rightarrow$ Probabilities of wrong decisions, weighted by the penalty terms

⇨ Average risk

$$r = r_1 P(\omega_1) + r_2 P(\omega_2)$$

➔**Choose R_1 and R_2 so that risk is minimized**

➔**Then assign $\underline{x}$ to ω_i if**

$$\lambda_1 \equiv \lambda_{11} p(\underline{x}|\omega_1)P(\omega_1) + \lambda_{21} p(\underline{x}|\omega_2)P(\omega_2) \quad <$$

$$\lambda_2 \equiv \lambda_{12} p(\underline{x}|\omega_1)P(\omega_1) + \lambda_{22} p(\underline{x}|\omega_2)P(\omega_2)$$

➔**Equivalently:**

assign $\underline{x}$ in $\omega_1(\omega_2)$ if

$$\boxed{\lambda_{12} \equiv \frac{p(\underline{x}|\omega_1)}{p(\underline{x}|\omega_2)} > (<) \frac{P(\omega_2)}{P(\omega_1)} \frac{\lambda_{21} - \lambda_{22}}{\lambda_{12} - \lambda_{11}}}$$

λ_{12} **likelihood ratio**

If $$P(\omega_1) = P(\omega_2) = \frac{1}{2} \text{ and } \lambda_{11} = \lambda_{22} = 0$$

$$\boxed{\begin{gathered} \underline{x} \rightarrow \omega_1 \text{ if } P(\underline{x}|\omega_1) > P(\underline{x}|\omega_2)\frac{\lambda_{21}}{\lambda_{12}} \\ \underline{x} \rightarrow \omega_2 \text{ if } P(\underline{x}|\omega_2) > P(\underline{x}|\omega_1)\frac{\lambda_{12}}{\lambda_{21}} \end{gathered}}$$

$$\text{if } \lambda_{21} = \lambda_{12} \Rightarrow \text{Minimum classification error probability}$$

➔An example:

- $p(x|\omega_1) = \dfrac{1}{\sqrt{\pi}} \exp(-x^2)$
- $p(x|\omega_2) = \dfrac{1}{\sqrt{\pi}} \exp(-(x-1)^2)$
- $P(\omega_1) = P(\omega_2) = \dfrac{1}{2}$
- $L = \begin{pmatrix} 0 & 0.5 \\ 1.0 & 0 \end{pmatrix}$

⇨Then the threshold value is:

$$x_0 \text{ for minimum } P_e :$$

$$x_0 : \exp(-x^2) = \exp(-(x-1)^2) \Rightarrow$$

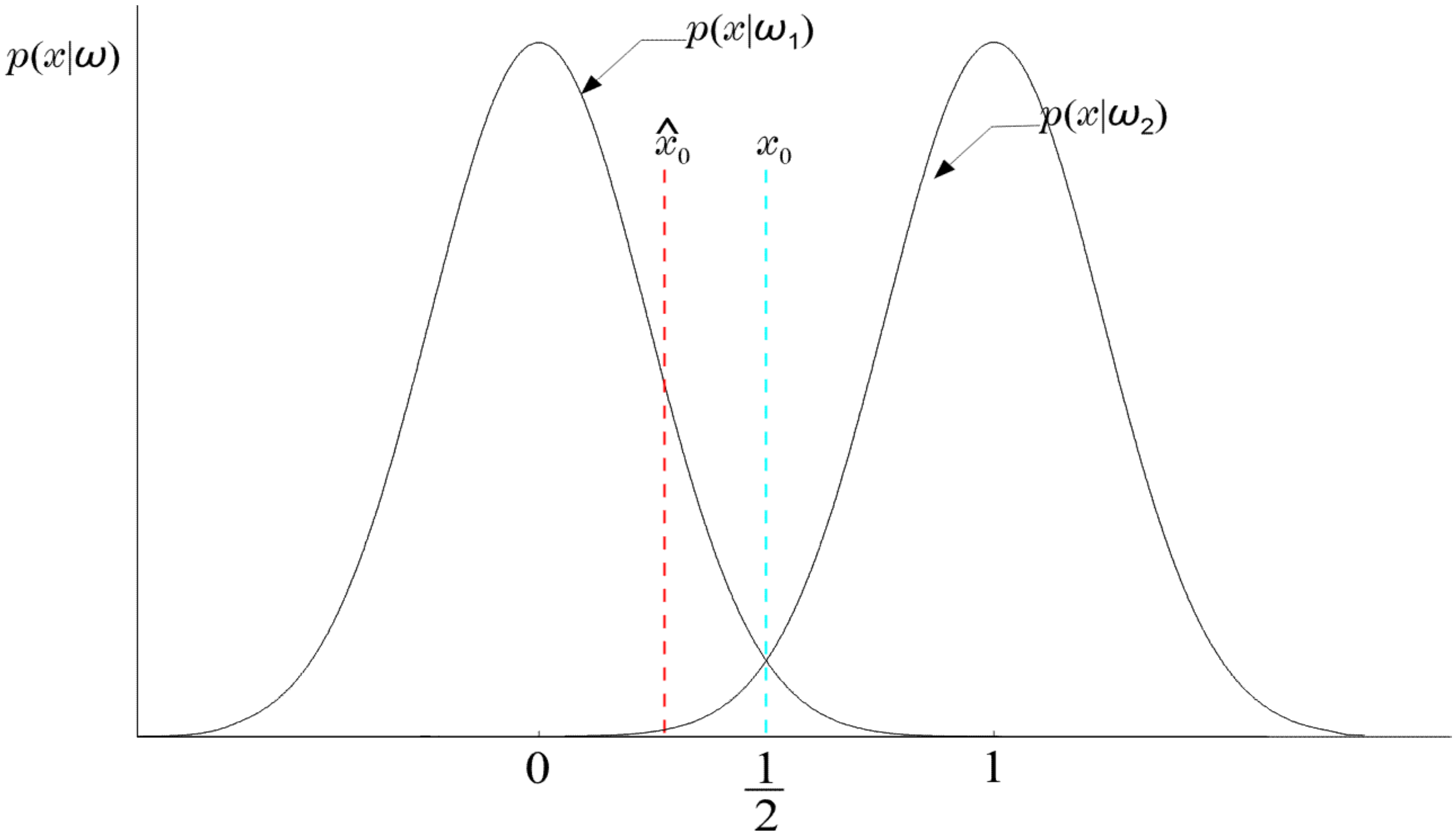
$$x_0 = \frac{1}{2}$$

⇨Threshold $\hat{x}_0$ for minimum risk

$$\hat{x}_0 : \exp(-x^2) = 2\exp(-(x-1)^2) \Rightarrow$$

$$\hat{x}_0 = \frac{(1-\lambda n2)}{2} < \frac{1}{2}$$

Thus $\hat{x}_0$ moves to the left of $\frac{1}{2} = x_0$ (WHY?)



Decision Surface

➔If R_i, R_j are contiguous: $g(\underline{x}) \equiv P(\omega_i|\underline{x}) - P(\omega_j|\underline{x}) = 0$

$$+ \quad R_i : P(\omega_i|\underline{x}) > P(\omega_j|\underline{x})$$

$$g(\underline{x}) = 0$$

$$- \quad R_j : P(\omega_j|\underline{x}) > P(\omega_i|\underline{x})$$

is the surface separating the regions. On one side is positive (+), on the other is negative (-).

It is known as Decision Surface

Which $p(x/\omega)$?

- ➔ **Two usual cases:**
  - ⇨ Each feature component x_i is a discrete variable, with p(x_i/ω) impulsive function
  - ⇨ Each feature component x_i is a continous variable, with p(x_i/ω) gaussian function
- ➔ **How to define $p(\underline{x}/\omega)$ starting from p(x_i/ω)?**
  - ⇨ Naive assumption: $p(\underline{x}/\omega)=p(x_1/\omega)\, p(x_2/\omega)\ldots p(x_n/\omega)$
  - ⇨ use multivariate gaussian distribution

Classifiers for Normal Distributions

➔ **Multivariate Gaussian pdf**

$$p(\underline{x}|\omega_i) = \frac{1}{(2\pi)^{\frac{\lambda}{2}}|\Sigma_i|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\underline{x}-\underline{\mu}_i)^{\mathrm{T}}\Sigma_i^{-1}(\underline{x}-\underline{\mu}_i)\right)$$

$$\underline{\mu}_i = E[\underline{x}] \lambda\times\lambda \text{ matrix in } \omega_i$$

$$\Sigma_i = E\left[(\underline{x}-\underline{\mu}_i)(\underline{x}-\underline{\mu}_i)^{\mathrm{T}}\right]$$

called covariance matrix

➔ $\ln(\cdot)$ is monotonic. Define:

⇨ $$g_i(\underline{x}) = \ln(p(\underline{x}|\omega_i)P(\omega_i)) =$$

$$\ln p(\underline{x}|\omega_i) + \ln P(\omega_i)$$

⇨ $$g_i(\underline{x}) = -\frac{1}{2}(\underline{x} - \underline{\mu}_i)^T \Sigma_i^{-1}(\underline{x} - \underline{\mu}_i) + \ln P(\omega_i) + C_i$$

$$C_i = -(\frac{\lambda}{2})\ln 2\pi - (\frac{1}{2})\ln|\Sigma_i|$$

⇨ Example: $$\Sigma_i = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}$$

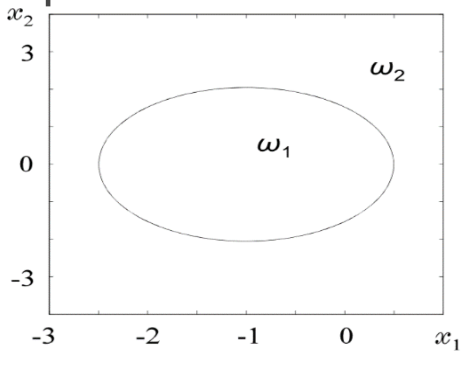
$$\Rightarrow \quad g_i(\underline{x}) = -\frac{1}{2\sigma^2}(x_1^2 + x_2^2) + \frac{1}{\sigma^2}(\mu_{i1}x_1 + \mu_{i2}x_2) - \frac{1}{2\sigma^2}(\mu_{i1}^2 + \mu_{i2}^2) + \ln(P\omega_i) + C_i$$

That is, $g_i(x)$ is quadratic and the surfaces

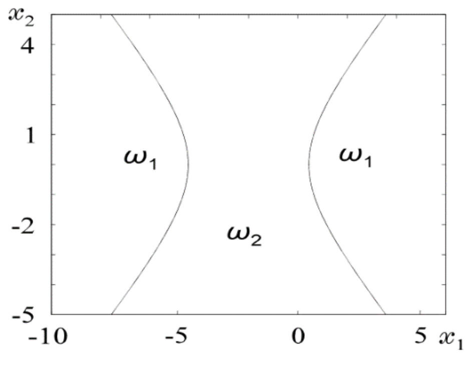
$$g_i(\underline{x}) - g_j(\underline{x}) = 0$$

quadrics, ellipsoids, parabolas, hyperbolas, pairs of lines.

For example:



(a)



(b)

➔Decision Hyperplanes

⇨Quadratic terms: $\underline{x}^T \Sigma_i^{-1} \underline{x}$

If ALL $\Sigma_i = \Sigma$ (the same) the quadratic terms are not of interest. They are not involved in comparisons. Then, equivalently, we can write:

$$\boxed{\begin{aligned} g_i(\underline{x}) &= \underline{w}_i^T \underline{x} + w_{io} \\ \underline{w}_i &= \Sigma^{-1} \underline{\mu}_i \\ w_{i0} &= \ln P(\omega_i) - \frac{1}{2} \underline{\mu}^T{}_i \Sigma^{-1} \underline{\mu}_i \end{aligned}}$$

Discriminant functions are LINEAR

⇨ Let in addition:

→ $\Sigma = \sigma^2 I$. Then

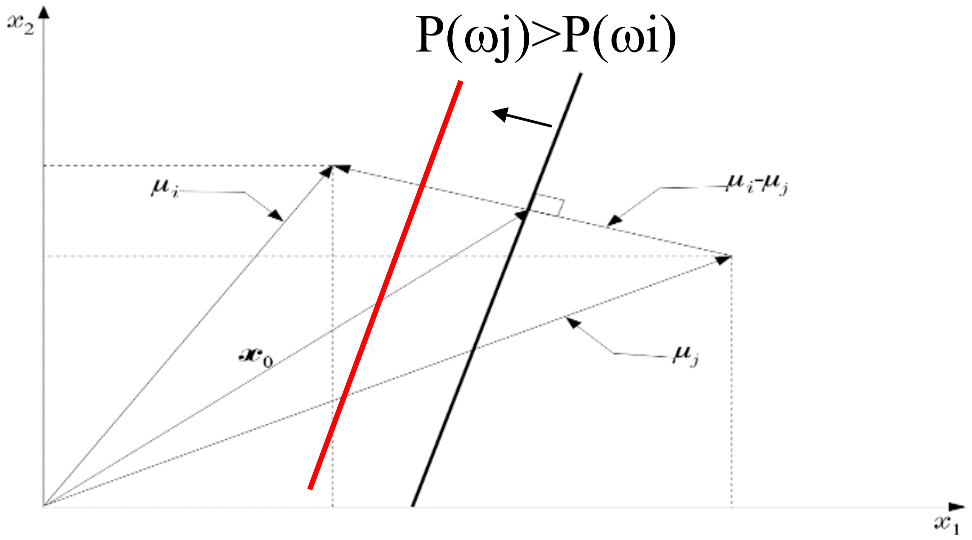
$$g_i(\underline{x}) = \frac{1}{\sigma^2}\underline{\mu}_i^T \underline{x} + w_{i0}$$

→ $g_{ij}(\underline{x}) = g_i(\underline{x}) - g_j(\underline{x}) = 0$

$$= \underline{w}^T(\underline{x} - \underline{x}_o)$$

→ $\underline{w} = \underline{\mu}_i - \underline{\mu}_j$,

→ $$\underline{x}_o = \frac{1}{2}(\underline{\mu}_i + \underline{\mu}_j) - \sigma^2 \ln\frac{P(\omega_i)}{P(\omega_j)} \frac{\underline{\mu}_i - \underline{\mu}_j}{\left\|\underline{\mu}_i - \underline{\mu}_j\right\|^2}$$



⇨Nondiagonal: $\Sigma \neq \sigma^2 I$

→ $$g_{ij}(\underline{x}) = \underline{w}^T(\underline{x} - \underline{x}_0) = 0$$

→ $$\underline{w} = \Sigma^{-1}(\underline{\mu}_i - \underline{\mu}_j)$$

→ $$\underline{x}_0 = \frac{1}{2}(\underline{\mu}_i + \underline{\mu}_j) - \lambda n\left(\frac{P(\omega_i)}{P(\omega_j)}\right)\frac{\underline{\mu}_i - \underline{\mu}_j}{\left\|\underline{\mu}_i - \underline{\mu}_j\right\|^2_{\Sigma^{-1}}}$$

$$\left\|\underline{x}\right\|_{\Sigma^{-1}} \equiv (\underline{x}^T \Sigma^{-1} \underline{x})^{\frac{1}{2}}$$

⇨Decision hyperplane $\begin{cases} \text{not normal to } \underline{\mu}_i - \underline{\mu}_j \\ \text{normal to } \Sigma^{-1}(\underline{\mu}_i - \underline{\mu}_j) \end{cases}$

Minimum Distance Classifiers

⇨ $P(\omega_i) = \frac{1}{M}$ equiprobable

⇨ $g_i(\underline{x}) = -\frac{1}{2}(\underline{x} - \underline{\mu}_i)^T \Sigma^{-1} (\underline{x} - \underline{\mu}_i)$

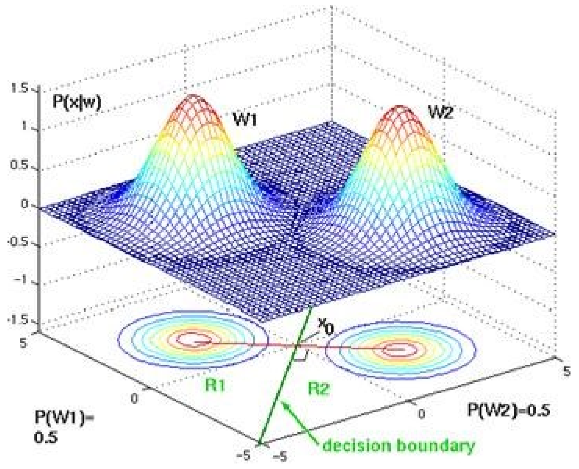
⇨ $\Sigma = \sigma^2 I : \text{ Assign } \underline{x} \rightarrow \omega_i :$

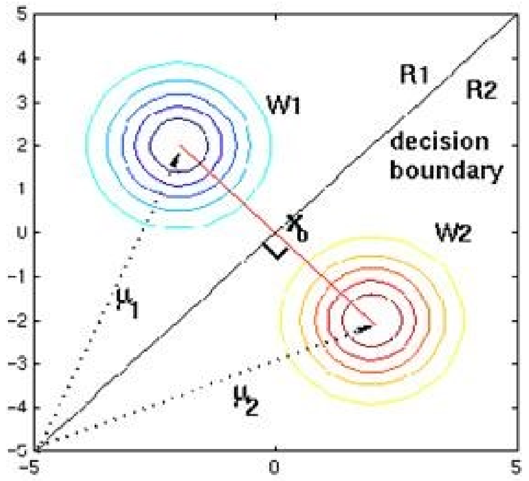
Euclidean Distance: $d_E \equiv \left\| \underline{x} - \underline{\mu}_i \right\|$ smaller

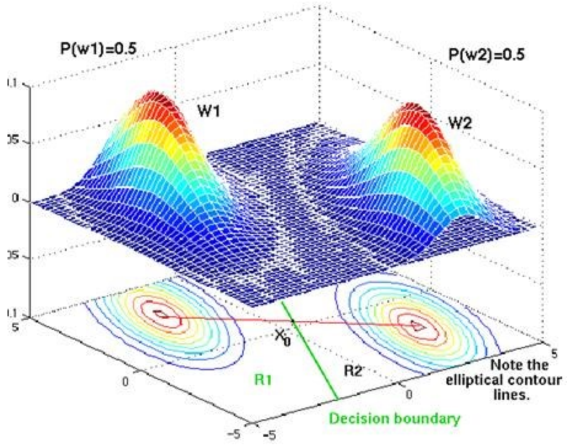
⇨ $\Sigma \neq \sigma^2 I : \text{ Assign } \underline{x} \rightarrow \omega_i :$

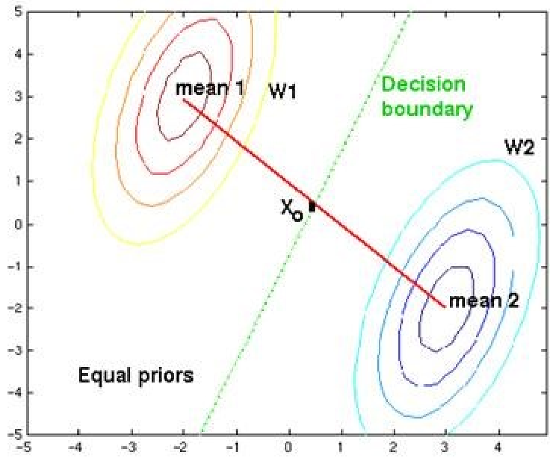
Mahalanobis Distance: $d_m = ((\underline{x} - \underline{\mu}_i)^T \Sigma^{-1} (\underline{x} - \underline{\mu}_i))^{\frac{1}{2}}$ smaller

Examples of decision boundaries

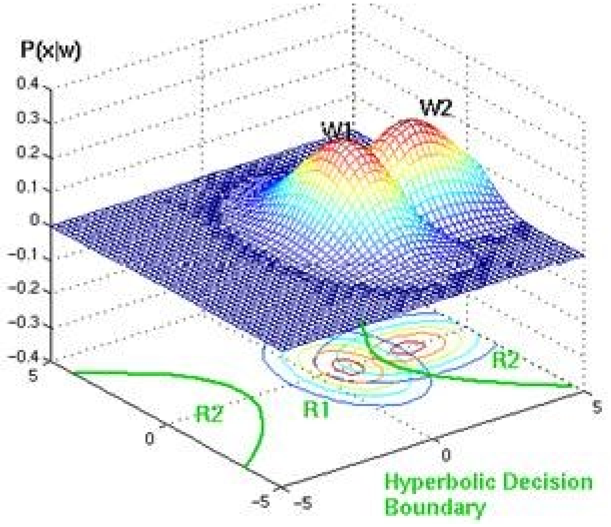


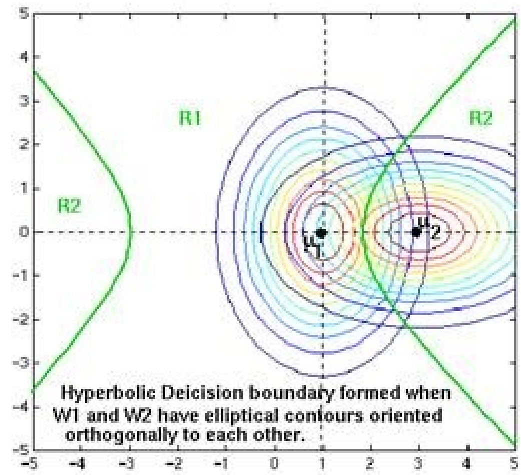


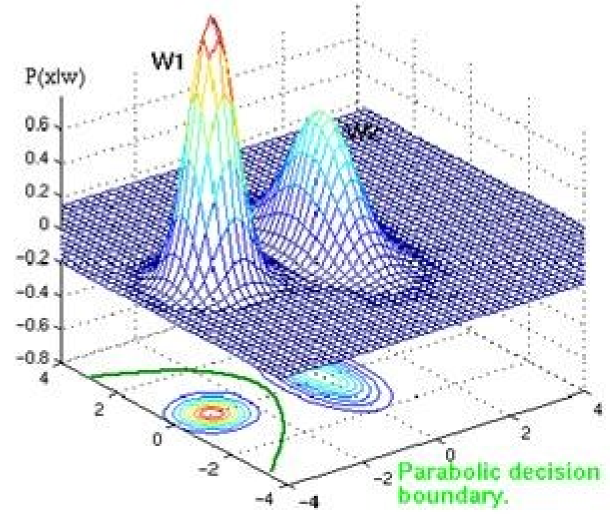


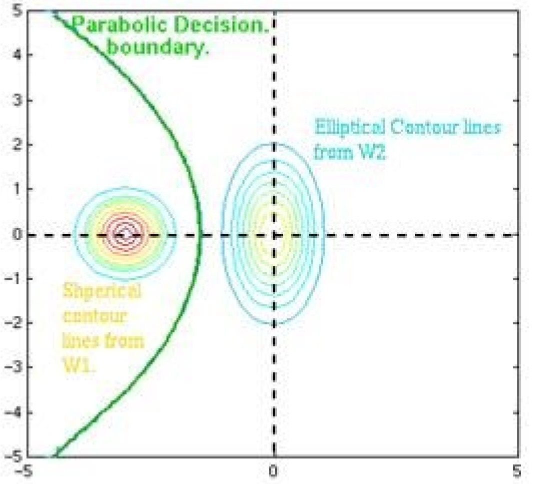


Examples of decision boundaries









Python Exercise 1

- ➔ **Generate 500 random samples of a random vector [x1, x2], whose p(x) is a multivariate gaussian distribution with mean [0, 0] and covariance matrix:**
  - ⇨ $\sigma 1^2 = \sigma 2^2 = 1$, $\sigma 12 = 0$
  - ⇨ $\sigma 1^2 = \sigma 2^2 = 0.2$, $\sigma 12 = 0$
  - ⇨ $\sigma 1^2 = 0.2$; $\sigma 2^2 = 1$, $\sigma 12 = 0$
  - ⇨ $\sigma 1^2 = \sigma 2^2 = 1$, $\sigma 12 = 0.5$
  - ⇨ $\sigma 1^2 = 0.3$, $\sigma 2^2 = 2$, $\sigma 12 = -0.5$
- ➔ **Plot the features in a bi-dimensional space.**
- ➔ **How do results change if we extract x1 and x2 independently?**

Python Exercise 2

- **Consider a 3-dimensional space of feaures and two possible classes; assuming p(x/ω_i) mutlivariate gaussian with m_1=[0, 0, 0] and m_2=[0.5, 0.5, 0.5] and P(ω_1)=P(ω_2).**
  - Classify the point x=[0.1 0.5, 0.1] when ∑=[0.8 0.01 0.01; 0.01 0.2 0.01; 0.01 0.01 0.2] with bayesian rules and euclidean distance. Comment the results.
  - Classify using the naive bayer classifier.
  - Repeat previous classification with ∑=[0.8 0 0; 0 0.2 0; 0 0 0.2]

Python Exercise 3

- ➔ **Consider a 2-dimensional space of feaures and two possible classes; assuming p(x/ω_i) mutlivariate gaussian with m_1=[0, 0] and m_2=[1, 1] and P(ω_1)=P(ω_2).**
  - ⇨ Classify the point x=[0.3 0.5] when ∑=[0.5, 0; 0, 1] with bayesian rules and euclidean distance. Comment the results.
  - ⇨ Find the decision surface for this classifier and how this surface changes when P(ω_1)=0.2, P(ω_2)=0.8.

But.. how do we know $p(x/\omega)$ in real applications?

Estimation of Pr density functions

- **What if I do not know feature distribution?**
  - I can estimate it!
- **Two cases:**
  - Parametric estimate: I know the shape, but I do not know the parameters
    - E.g. Gaussian distribution, with unknown mean value and standard deviation
  - Non parametric estimate: I do not know the shape, but I have some feature samples

Parametric Estimate

➔ **Maximum Likelihood** assuming to know N cases of feature vectors

⇨ Let $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N$ known and independent

⇨ Let $p(\underline{x})$ known within an unknown vector parameter $\underline{\theta}$: $p(\underline{x}) \equiv p(\underline{x};\underline{\theta})$

$$X = \{\underline{x}_1, \underline{x}_2, \ldots \underline{x}_N\}$$

$$p(X;\underline{\theta}) \equiv p(\underline{x}_1, \underline{x}_2, \ldots \underline{x}_N;\underline{\theta})$$

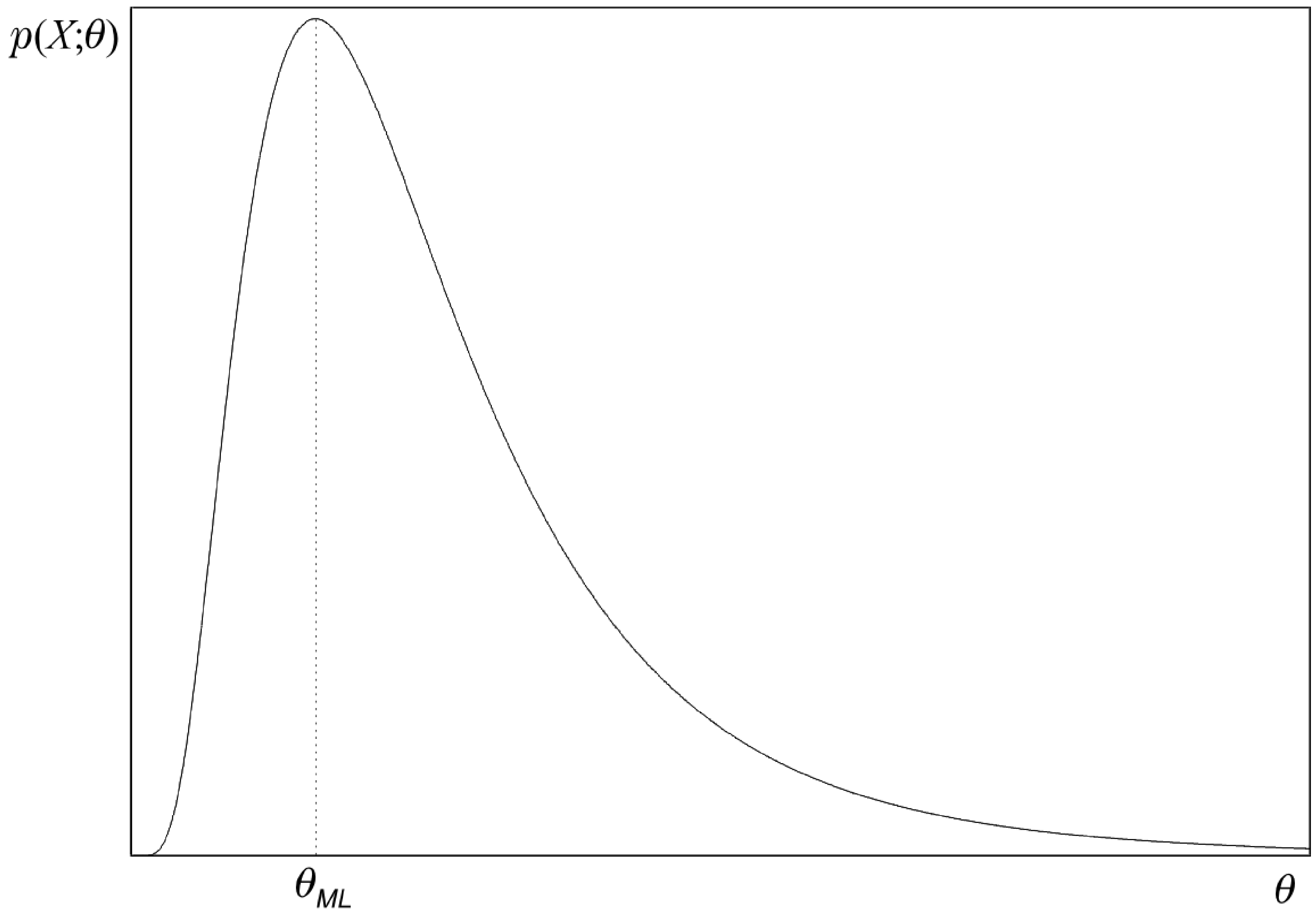
$$= \prod_{k=1}^{N} p(\underline{x}_k;\underline{\theta})$$

which is known as the Likelihood of $\underline{\theta}$ w.r. to X

$$\Rightarrow \boxed{\hat{\underline{\theta}}_{\mathrm{ML}} : \arg\max_{\underline{\theta}} \prod_{k=1}^{N} p(\underline{x}_k ; \underline{\theta})}$$

$$\Rightarrow L(\underline{\theta}) \equiv \ln p(X;\underline{\theta}) = \sum_{k=1}^{N} \ln p(\underline{x}_k ; \underline{\theta})$$

$$\Rightarrow \hat{\underline{\theta}}_{ML} : \frac{\partial L(\underline{\theta})}{\partial(\underline{\theta})} = \sum_{k=1}^{N} \frac{1}{p(\underline{x}_{k\,;}\,\underline{\theta})} \frac{\partial p(\underline{x}_{k\,;}\,\underline{\theta})}{\partial(\underline{\theta})} = \underline{0}$$


$p(X;\theta)$
θ_{ML}
θ

If, indeed, there is a $\underline{\theta}_0$ such that

$p(\underline{x}) = p(\underline{x};\underline{\theta}_0)$, then

$$\lim_{N\to\infty} E[\underline{\theta}_{ML}] = \underline{\theta}_0$$

$$\lim_{N\to\infty} E\left\|\hat{\underline{\theta}}_{ML} - \underline{\theta}_0\right\|^2 = 0$$

Asymptotically unbiased and consistent

➔Example:

$$p(\underline{x}):\ N(\underline{\mu},\Sigma):\ \underline{\mu}\text{ unknown},\quad \underline{x}_1,\underline{x}_2,\ldots,\underline{x}_N\quad p(\underline{x}_k)\equiv p(\underline{x}_k;\underline{\mu})$$

$$L(\underline{\mu})=\ln\prod_{k=1}^{N}p(\underline{x}_k;\underline{\mu})=C-\frac{1}{2}\sum_{k=1}^{N}(\underline{x}_k-\underline{\mu})^T\Sigma^{-1}(\underline{x}_k-\underline{\mu})$$

$$p(\underline{x}_k;\underline{\mu})=\frac{1}{(2\pi)^{\frac{l}{2}}|\Sigma|^{\frac{1}{2}}}\exp(-\frac{1}{2}(\underline{x}_k-\underline{\mu})^T\Sigma^{-1}(\underline{x}_k-\underline{\mu}))$$

$$\frac{\partial L(\underline{\mu})}{\partial(\underline{\mu})}\equiv\begin{bmatrix}\frac{\partial L}{\partial\mu_1}\\ .\\ .\\ .\\ \frac{\partial L}{\partial\mu_l}\end{bmatrix}=\sum_{k=1}^{N}\Sigma^{-1}(\underline{x}_k-\underline{\mu})=\underline{0}\Rightarrow\underline{\mu}_{ML}=\frac{1}{N}\sum_{k=1}^{N}\underline{x}_k$$

Remember : if $A=A^T\Rightarrow\dfrac{\partial(\underline{\alpha}^T A\underline{\alpha})}{\partial\underline{\alpha}}=2A\underline{\alpha}$

Exercise

- ➔Find the mean and the variance of a mono-dimensional gaussian density function for which N observations are available by applying the maximum likelihood criterion.
- ➔Find the maximum likelihood estimate of the probability of extracting 1 in a dice, starting from N observations of extractions.

Parametric Estimate

➔ Maximum A posteriori Probability Estimation

- ⇨ In ML method, $\underline{\theta}$ was considered as a <u>deterministic parameter</u>
- ⇨ Here we shall look at $\underline{\theta}$ as a <u>random vector</u> described by a pdf $p(\underline{\theta})$, assumed to be known
- ⇨ Given $X = \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N\}$

  Compute the maximum of $p(\underline{\theta} | X)$
- ⇨ From Bayes theorem

$$p(\underline{\theta})\, p(X | \underline{\theta}) = p(X)\, p(\underline{\theta} | X) \text{ or}$$

$$p(\underline{\theta} | X) = \frac{p(\underline{\theta})\, p(X | \underline{\theta})}{p(X)}$$

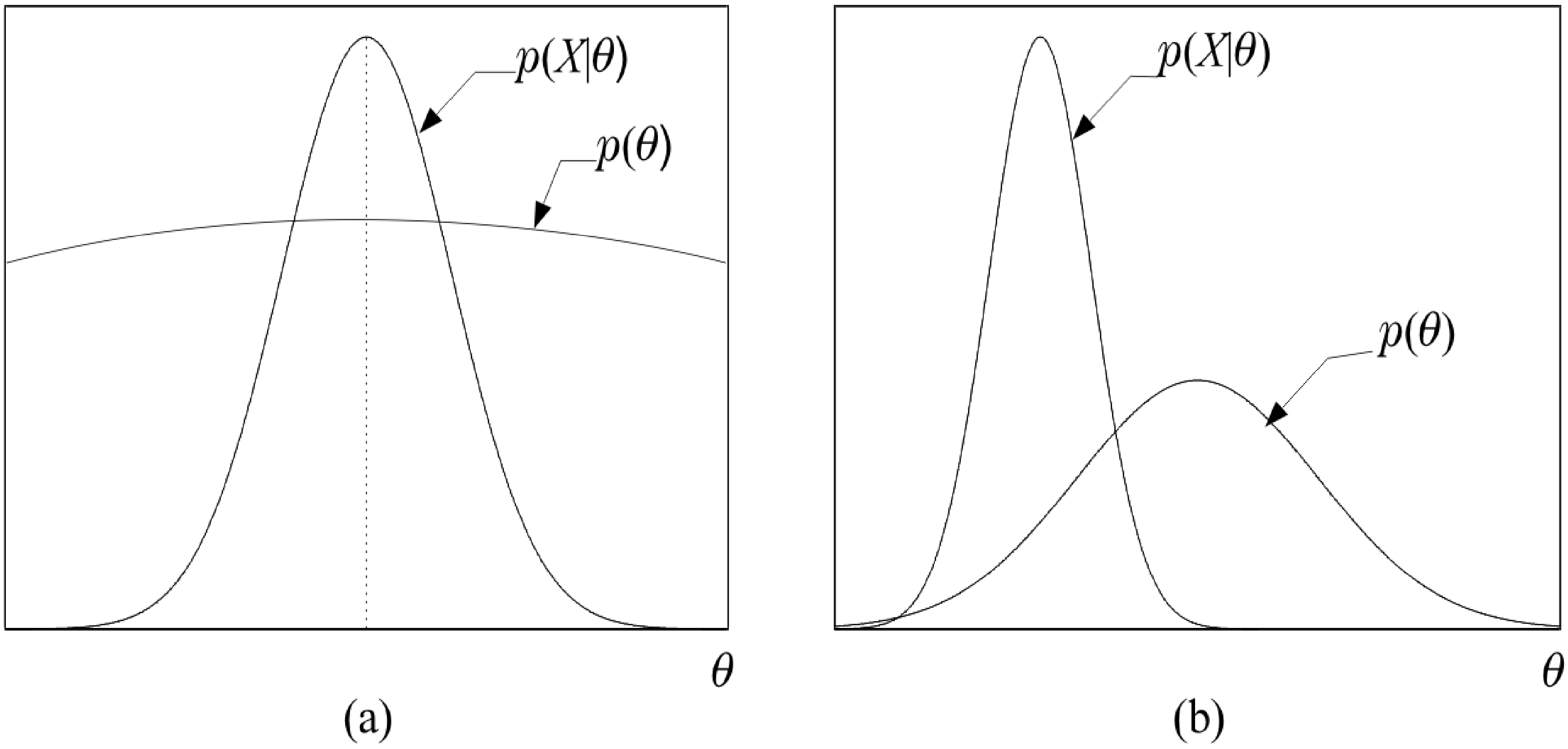
⇨The method:

$$\hat{\underline{\theta}}_{MAP} = \arg\max_{\underline{\theta}} p(\underline{\theta}|X) \text{ or}$$

$$\boxed{\hat{\underline{\theta}}_{MAP} : \frac{\partial}{\partial \underline{\theta}}(P(\underline{\theta})p(X|\underline{\theta}))}$$

If $p(\underline{\theta})$ is uniform or broad enough $\hat{\underline{\theta}}_{MAP} \cong \underline{\theta}_{ML}$

In other words, maximizing p(θ/X)=P(θ)p(X/ θ) is equivalent to maximizing p(X/ θ)


$p(X|\theta)$
$p(\theta)$
θ
(a)
$p(X|\theta)$
$p(\theta)$
θ
(b)

➔ Example:

$$p(\underline{x}):\ N(\underline{\mu},\Sigma),\quad \underline{\mu}\ \text{ unknown},\quad X=\{\underline{x}_1,\ldots,\underline{x}_N\}$$

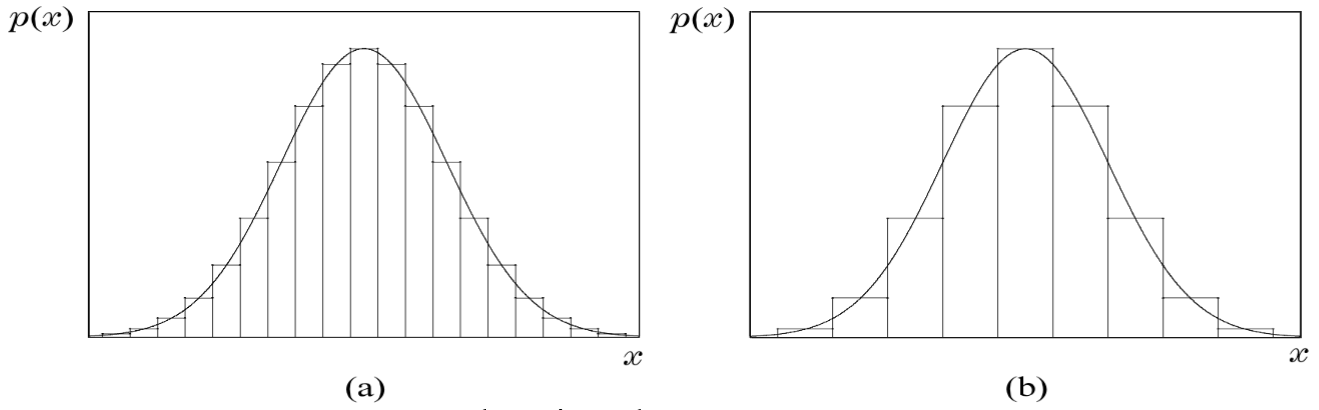
$$p(\underline{\mu})=\frac{1}{(2\pi)^{\frac{l}{2}}\sigma_\mu^l}\exp\left(-\frac{\left\|\underline{\mu}-\underline{\mu}_0\right\|^2}{2\sigma_\mu^2}\right)$$

$$\underline{\theta}_{MAP}:\ \frac{\partial}{\partial\underline{\mu}}\ln\left(\prod_{k=1}^{N}p(\underline{x}_k|\underline{\mu})\,p(\underline{\mu})\right)=\underline{0}\ \text{ or }\ \sum_{k=1}^{N}\frac{1}{\sigma^2}(\underline{x}_k-\underline{\mu})-\frac{1}{\sigma_\mu^2}(\underline{\mu}-\underline{\mu}_0)=\underline{0}\Rightarrow$$

$$\underline{\hat{\mu}}_{MAP}=\frac{\underline{\mu}_0+\frac{\sigma_\mu^2}{\sigma^2}\sum_{k=1}^{N}\underline{x}_k}{1+\frac{\sigma_\mu^2}{\sigma^2}N}\quad For\ \ \frac{\sigma_\mu^2}{\sigma^2}>>1,\text{ or for N}\to\infty$$

$$\underline{\hat{\mu}}_{\mathrm{MAP}}\cong\underline{\hat{\mu}}_{ML}=\frac{1}{N}\sum_{k=1}^{N}\underline{x}_k$$

Non Parametric Estimation



⇨ $$P \approx \frac{k_N}{N} \quad \begin{array}{l} \nearrow\ k_N \text{ in } h \\ \searrow\ N \text{ total} \end{array}$$

Discretization of probability in each bin

⇨ $$\hat{p}(x) \equiv \hat{p}(\hat{x}) = \frac{1}{h}\frac{k_N}{N}, |x - \hat{x}| \leq \frac{h}{2}$$

From bin mass function to continuous p(x) function

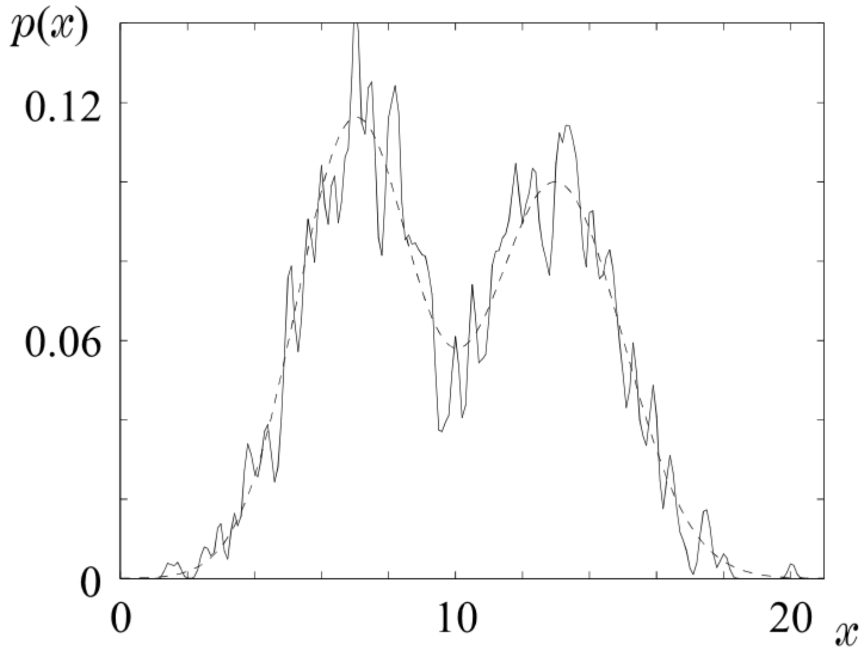
⇨ If $p(x)$ continuous , $\hat{p}(x) \to p(x)$ as $N \to \infty$, if

$$h_N \to 0, \quad k_N \to \infty, \quad \frac{k_N}{N} \to 0$$

⇨Variance

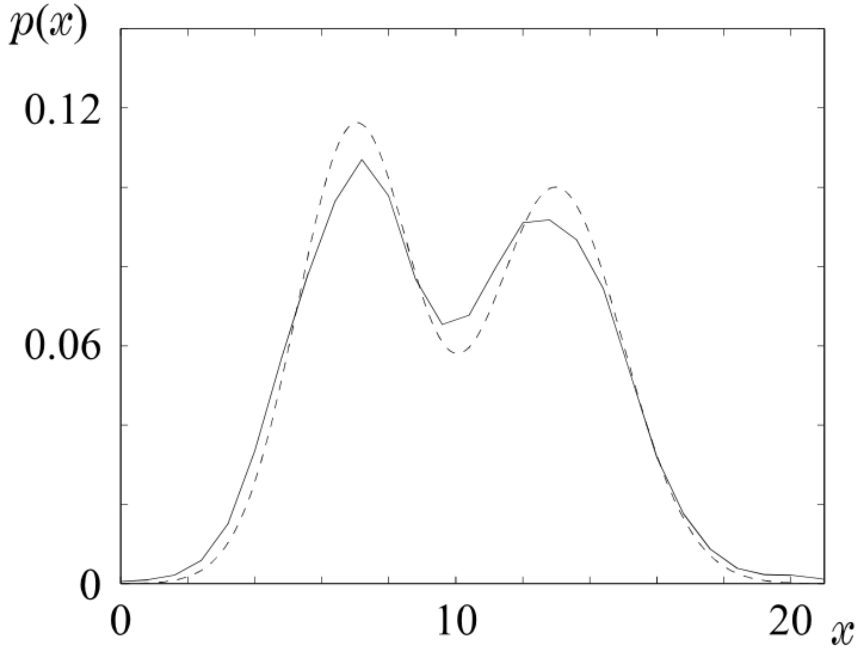
→The smaller the *h* the higher the variance

h=0.1, N=1000



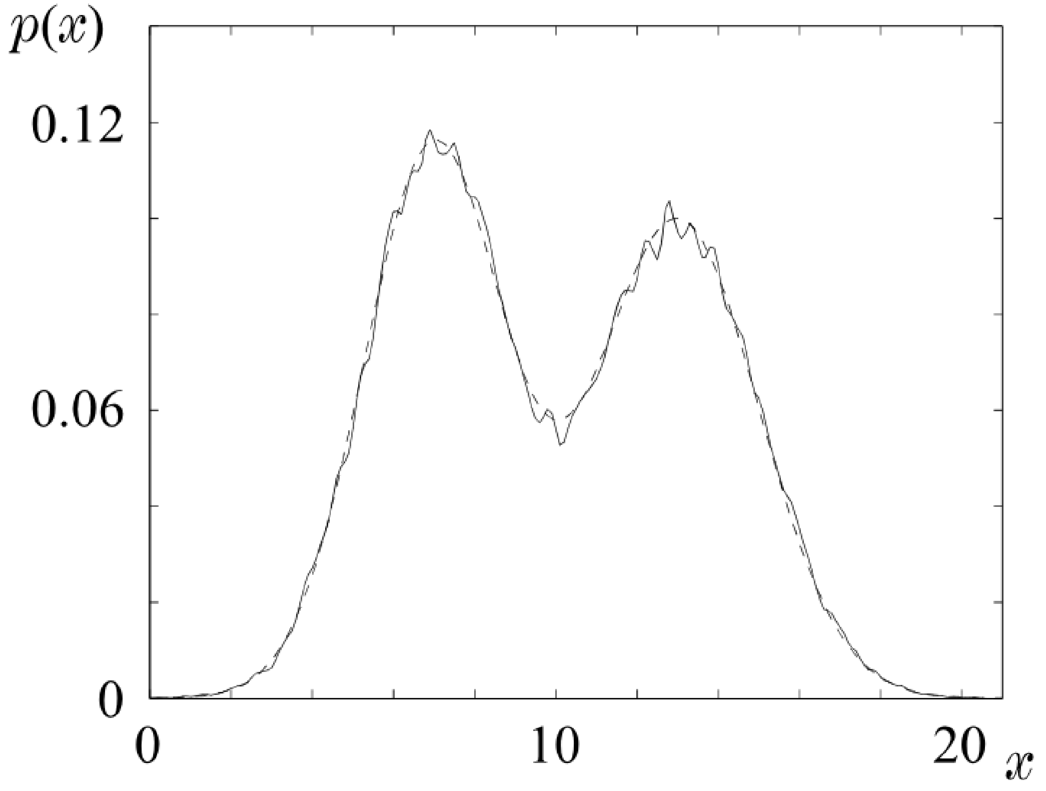
(a)

h=0.8, N=1000



(b)

h=0.1, N=10000

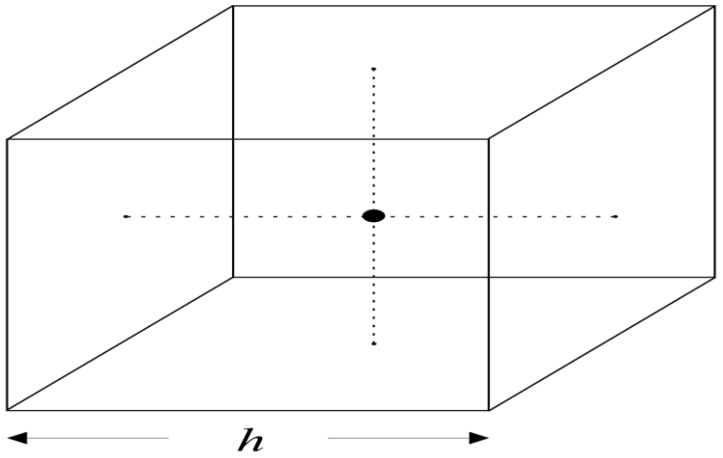


(b)

The higher the N the better the accuracy

Parzen Windows

- ⇨ Is the previous method generalizable in multi-dimensional spaces?
  - → YES!
- ⇨ Divide the multidimensional space in hypercubes
  - → The number of points to be observed increases with the dimension of the space



If we assume independent features, p(x)=p(x1)p(x2)..p(xn) and..
we do not need working in multidimensional space

Naive-Bayes Classifiers

⇨ Let $\underline{x} \in \Re^{\lambda}$ and the goal is to estimate $p(\underline{x} \mid \omega_i)$ $i = 1, 2, \ldots, M$.

For a "good" estimate of the pdf one would need, say, N^{ℓ} points.

⇨ Assume $x_1, x_2, \ldots, x_{\ell}$ mutually independent. Then:

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

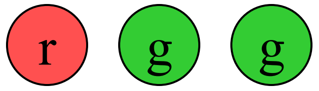
⇨ In this case, one would require, roughly, N points for each pdf. Thus, a number of points of the order $N \cdot \ell$ would suffice.

⇨ It turns out that the Naïve – Bayes classifier works reasonably well even in cases that violate the independence assumption.

Training

- **Based on the estimation of the distribution of a random variable**
  - Discrete: Frequency counting
  - Continuous: Estimate mean and deviation of a Gaussian variable
  - Very effective for large data sets, despite of its simplicity!
- ***Empirically:* use training data and apply ML estimates!**
  - E.g. For each value x, look at the the *empirical rate* of that value:

$$\hat{P}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

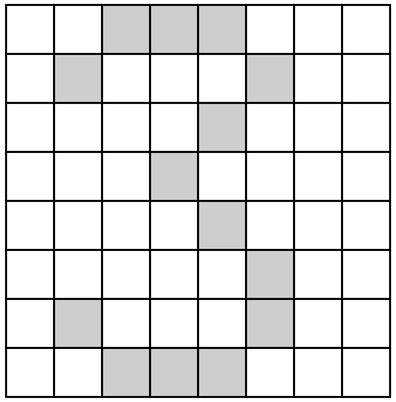


$$\hat{P}(\text{r}) = 1/3$$

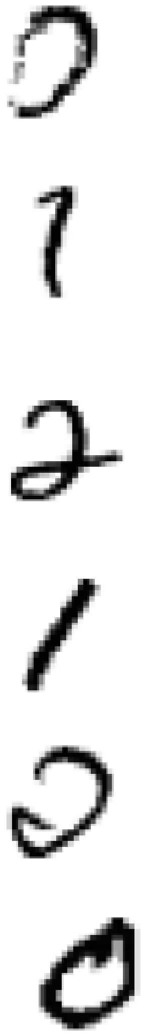
- ***Class Elicitation:* ask a human!**
  - Usually need domain experts, and sophisticated ways of eliciting probabilities (e.g. betting games)
  - Trouble calibrating

A Digit Recognizer

- ➔**Input: pixel grids**



- ➔**Output: a digit 0-9**



Naïve Bayes for Digits

➔ **Simple version:**

- ⇨ One feature F_{ij} for each grid position <i,j>
- ⇨ Feature values are on / off based on whether intensity is more or less than 0.5
- ⇨ Input looks like:

$$\rightarrow \langle F_{0,0} = 0 \;\; F_{0,1} = 0 \;\; F_{0,2} = 1 \;\; F_{0,3} = 1 \;\; F_{0,4} = 0 \;\; \ldots F_{15,15} = 0 \rangle$$

➔ **Naïve Bayes model:**

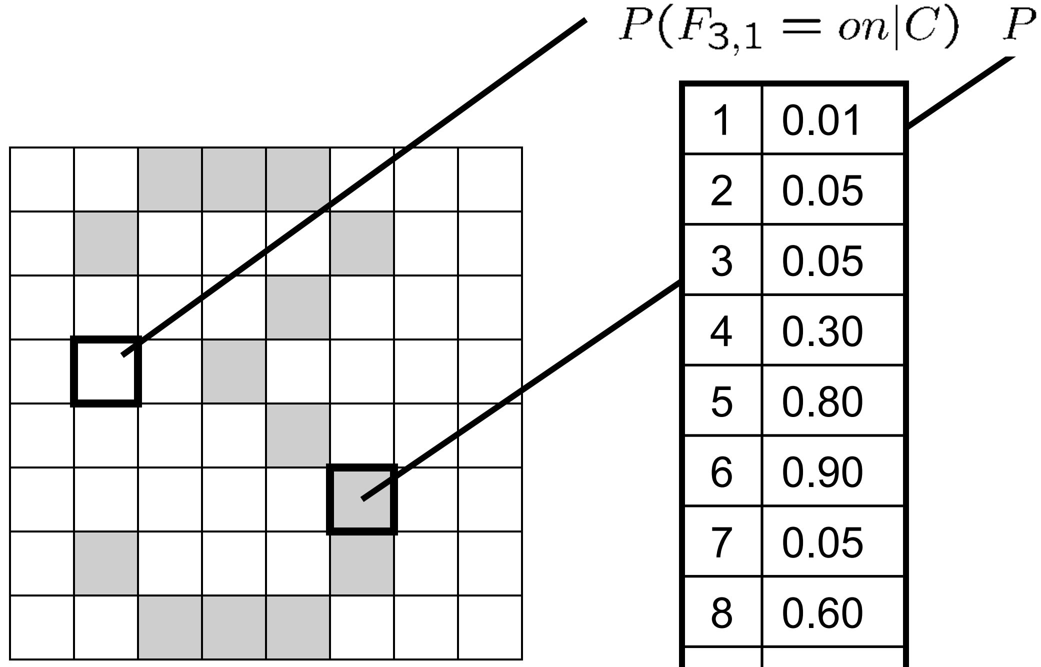
$$P(C, F_{0,0} \ldots F_{15,15}) = P(C) \prod_{i,j} P(F_{i,j} | C)$$

➔ **What do we need to learn?**

Training Digit Recognition

$P(C)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|C)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|C)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

A Spam Filter

- **Naïve Bayes spam filter**
- **Data:**
  - Collection of emails, labeled spam or ham
  - Note: someone has to hand label all this data!
  - Split into training, held-out, test sets
- **Classifiers**
  - Learn on the training set
  - (Tune it on a held-out set)
  - Test it on new emails

✗

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

✗

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES
FOR ONLY $99

✓

Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Naïve Bayes for Spam

➔ **Naïve Bayes:**

- ⇨ Predict unknown cause (spam vs. ham)
- ⇨ Independent evidence from observed variables (e.g. the words)

➔ **Features:**

- ⇨ list of common words found in spam/ham emails
  - ➔ e.g. review, award, date, prize..

➔ **Model:**

$$P(C, W_1 \ldots W_n) = P(C) \prod_i P(W_i | C)$$

Training Spam Filtering

- ➔ **Model:** $P(C, W_1 \ldots W_n) = P(C) \prod_i P(W_i|C)$
- ➔ **What are the parameters?**

$P(C)$

ham	0.5
spam	0.5

$P(W|\text{spam})$

order	0.0156
address	0.0153
Money	0.0115
language	0.0093
business	0.0093
work	0.0080
…	

$P(W|\text{ham})$

language	0.0210
university	0.0133
english	0.0119
follow	0.0110
research	0.0108
Information	0.0092
...	

- ➔ **Where do these tables come from?**
  - ⇨ We will see later on our python trainer!

Generalization and Overfitting

- ➔ **Relative frequency parameters will overfit the training data!**
  - ⇨ Unlikely that every occurrence of "minute" is 100% spam
  - ⇨ Unlikely that every occurrence of "seriously" is 100% ham
  - ⇨ What about all the words that don't occur in the training set?
  - ⇨ In general, we can't go around giving unseen events zero probability
- ➔ **As an extreme case, imagine using the entire email as the only feature**
  - ⇨ Would get the training data perfect (if deterministic labeling)
  - ⇨ Wouldn't *generalize* at all
- ➔ **To generalize better: we need to smooth or regularize the estimates**

Estimation: Smoothing

- ➔**Problems with maximum likelihood estimates:**
  - ⇨If I flip a coin once, and it's heads, what's the estimate for P(heads)?
  - ⇨What if I flip 10 times with 8 heads?
  - ⇨What if I flip 10M times with 8M heads?

- ➔**Can we empirically work on MAP rather than ML?**
  - ⇨Basic idea:
    - →We have some prior expectation about parameters (here, the probability of heads)
    - →Given little evidence, we should skew towards our prior
    - →Given a lot of evidence, we should listen to the data

Estimation: Smoothing

- ➔ **Relative frequencies are the maximum likelihood estimates**
  - ⇨ (from a binomial model of counting x in n trials)

$$\theta_{ML} = \arg\max_{\theta} P(\mathbf{X}|\theta) \quad \Rightarrow \quad \hat{P}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

- ➔ **In Bayesian statistics, we think of the parameters as just another random variable, with its own distribution**

$$\begin{aligned}\theta_{MAP} &= \arg\max_{\theta} P(\theta|\mathbf{X}) \\ &= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X}) \\ &= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)\end{aligned} \quad \Rightarrow \quad ????$$

Estimation: Laplace Smoothing

➔ Laplace's estimate:

- ⇨ Pretend you saw every outcome once more than you actually did

$$P_{LAP}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]}$$

$$= \frac{c(x)+1}{N+|X|}$$

$$P_{ML}(X) = \left\langle \frac{2}{3}, \frac{1}{3} \right\rangle$$

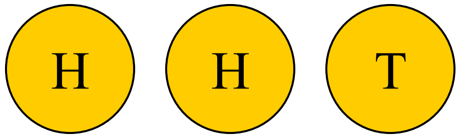
$$P_{LAP}(X) = \left\langle \frac{3}{5}, \frac{2}{5} \right\rangle$$

- ⇨ Can derive this as a MAP estimate with *Dirichlet priors*

Estimation: Laplace Smoothing

- **Laplace's estimate (extended):**
  - Pretend you saw every outcome k extra times

$$P_{LAP,k}(x) = \frac{c(x)+k}{N+k|X|}$$



$$P_{LAP,0}(X) = \left\langle \frac{2}{3}, \frac{1}{3} \right\rangle$$

$$P_{LAP,1}(X) = \left\langle \frac{3}{5}, \frac{2}{5} \right\rangle$$

$$P_{LAP,100}(X) = \left\langle \frac{102}{203}, \frac{101}{203} \right\rangle$$

  - What's Laplace with k = 0?
  - k is the strength of the prior

- **Laplace for conditionals:**
  - S

$$P_{LAP,k}(x|y) = \frac{c(x,y)+k}{c(y)+k|X|}$$

Bayes classifiers in Python

Bayes Classifiers in Python

scikit-learn

Three possibile probability estimators:

- ➔ **Gaussian:** assumes that features follow a normal distribution.
- ➔ **Multinomial:** it is used for discrete counts (or events with non-binary outputs). You can think of it as "number of times outcome number x_i is observed over the n trials".
- ➔ **Bernoulli:** The binomial model is useful if your feature vectors are binary (i.e. zeros and ones). It is a multivariate Bernoulli distribution.

```
#Create a Gaussian Classifier
model = GaussianNB()
# Train the model using the training sets
model.fit(x, y)
#Predict Output
predicted= model.predict(test_features)
```

Text Classifiers

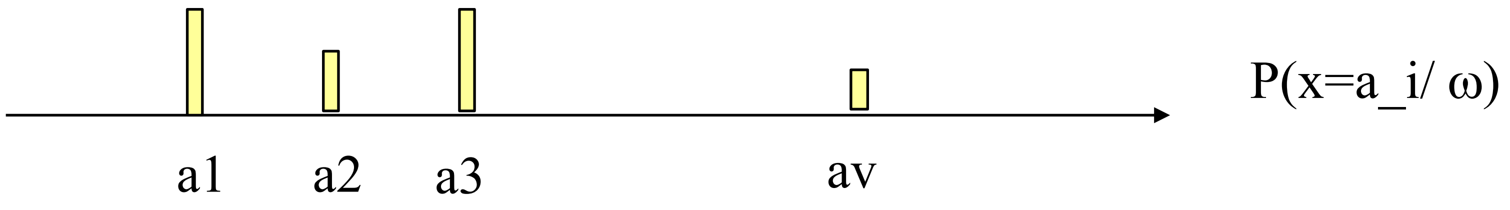
- ➔ **Suppose to consider a document as a sequence of n words**
  - ⇨ Doc1=w1, w2, … wN
- ➔ **Each text category (SPAM/No SPAM, financial/politics/news) is characterized by difference occurrence probabilities of words in a dictionary D=[a1, a2, … aV]**
- ➔ **Two feature models:**
  - ⇨ [x1, x2, … xV] =[1 1 0 0 .. 1] says if words a1, a2 … aV are in the document
  - ⇨ [x1, x2, … xN]=[w1, w2, …wN] sequence of n words included in the document

Two event models for training

- ➔ **Bernoulli:**
  - ⇨ x_i is {0, 1} : word a_i is in the document or not
    - →P(x_i=1/ω)=#(doc with word a_i)/#tot_doc
    - →P(x_i=0/ω)=1-#(doc with word a_i)/#tot_doc

P(a1/ ω)
a1=yes
a1=no
…
P(av/ ω)
av=yes
av=no

- ➔ **Multinomial**
  - ⇨ x_i is a word whose occurrence probability in the dictionary is:
    - →P(x_i / ω)=#(occurrences of words a_i)/#tot_words
  - ⇨ It has been demonstrated that this approach works better for text classification!



A SPAM Filter

- ➔ **Spam filtering is an example of document classification task**
  - ⇨ Goal: target email as spam or non-spam (a.k.a. ham) mail.
- ➔ **We work on a publicly available mail corpus.**
  - ⇨ Ling-spam corpus (others are available)
  - ⇨ The extracted subset on which we will be working can be downloaded from the link in:
    - → https://appliedmachinelearning.wordpress.com/2017/01/23/email-spam-filter-python-scikit-learn/

Project Steps

- **Preparing the text data. (already done in ling-spam)**
  - Split data into <u>train</u> and <u>test</u> sets
  - Text cleaning (e.g. remove *the, as, of..*)
  - Lemmatization (grouping together verbs/plurals as single words)
- **Creating word dictionary.**
  - Reading test files line by line and counting word occurrences
- **Feature extraction process**
  - Select most relevant words for discriminating
- **Training the classifier with a multinomial model**
  - Estimate the probability of word occurrences in spam and no spam email by using multinomial distributions

Implementation example

```
def make_Dictionary(train_dir):
    emails = [os.path.join(train_dir,f) for f in os.listdir(train_dir)]
    all_words = []
    for mail in emails:
        with open(mail) as m:
            for i,line in enumerate(m):
                if i == 2:
                    words = line.split()
                    all_words += words

    dictionary = Counter(all_words)

    list_to_remove = dictionary.keys()
    for item in list_to_remove:
        if item.isalpha() == False:
            del dictionary[item]
        elif len(item) == 1:
            del dictionary[item]
    dic1 = dictionary.most_common(100)
    return dic1
```

Implementation example

```
def extract_features(mail_dir):
    files = [os.path.join(mail_dir,fi) for fi in os.listdir(mail_dir)]
    features_matrix = np.zeros((len(files),100))
    docID = 0;
    for fil in files:
      with open(fil) as fi:
        for i,line in enumerate(fi):
          if i == 2:
            words = line.split()
            for word in words:
              wordID = 0
              for i,d in enumerate(dictionary):
                if d[0] == word:
                  wordID = i
                  features_matrix[docID,wordID] = words.count(word)
        docID = docID + 1
    return features_matrix
```

Implementation example

```
# Create a dictionary of words with its frequency
train_dir = '.\\myscripts\\train-mails'
dictionary = make_Dictionary(train_dir)

# Prepare feature vectors per training mail and its labels
train_labels = np.zeros(702)
train_labels[351:701] = 1
train_matrix = extract_features(train_dir)
print train_matrix[1:10, 1:10]
print train_matrix[351:361, 1:10]

# Naive bayes classifier and its variants
model = MultinomialNB()
model.fit(train_matrix,train_labels)

# Test the unseen mails for Spam
test_dir = '.\\myscripts\\test-mails'
test_matrix = extract_features(test_dir)
test_labels = np.zeros(260)
test_labels[130:260] = 1
result = model.predict(test_matrix)
print confusion_matrix(test_labels,result)
```

Python Exercise 4

- **Give a look to the project spam.py**
  - Can you print the most common 10 words found in the archive?
  - Can you print the occurrences of the most common 10 words in some exemplary 10 spam and 10 ham emails?
  - How classification results are affected by reducing the dimension of the monitored words to 100?
    - Can you quantify the complexity reduction on the new classifier?
  - Can you extend the implementation to the bernoulli model?

Python Execise 5

- ➔**Import the digits data from sklearn**
  - ⇨try to train a classifier able to recognize digits 1 and 0 starting from 3 sample pictures only.
  - ⇨Why a zero occurrence of a specific features can be a problem for the classifier?

Python Exercise 6

- ➔ **Try to implement a model for the play tennis predictor that we discussed in the introductory part of this module.**
- ➔ **Classify the vector (Sunny, Cool, High, Strong) with the model.**
  - ⇨ How to deal with binary, caterogical and continous variables at the same time?
    - →Is it easier to use the sklearn pre-defined models or to define a custom-model?
    - →which models can be combined from sklearn?
  - ⇨ How to design features for multinomial or bernoulli models?

Other ‘Naive’ Classifiers

Nearest Neighbor Rule

⇨ Choose k out of the N training vectors, identify the k nearest ones to $\underline{x}$

⇨ Out of these k identify k_i that belong to class ω_i

$$\text{Assign } \underline{x} \rightarrow \omega_i : k_i > k_j \quad \forall i \neq j$$

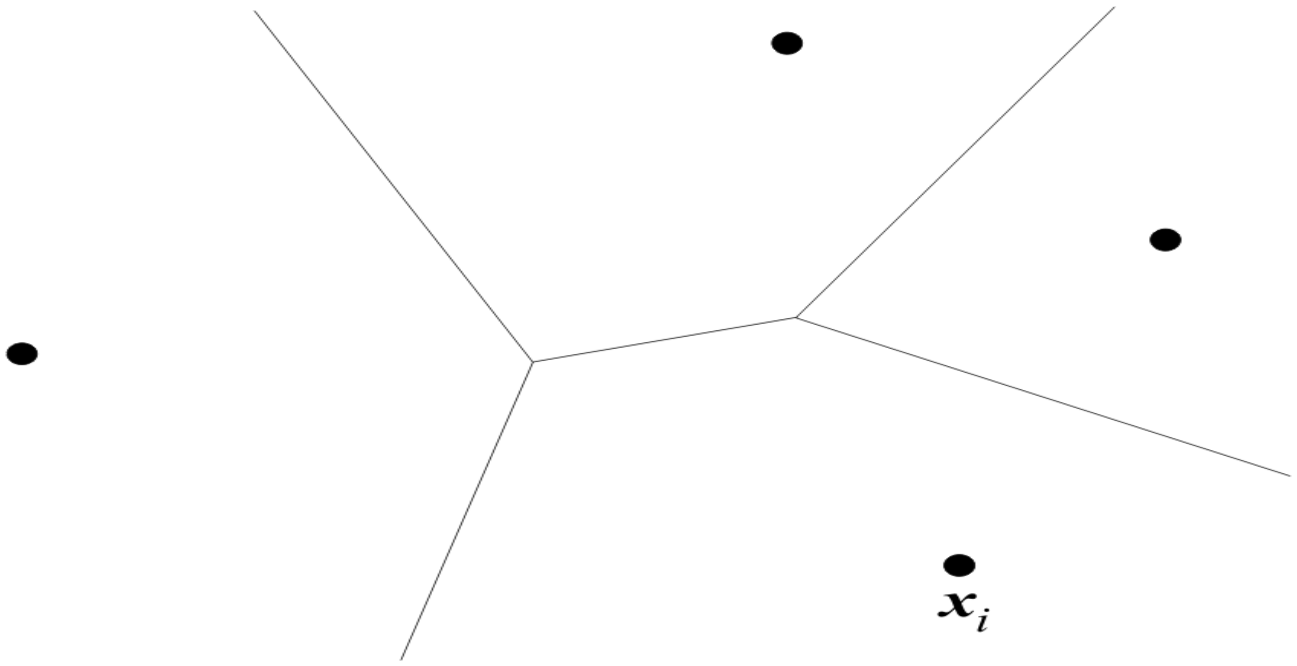
⇨ The simplest version

$\boxed{k=1 \text{ !!!}}$

⇨ For large N this is not bad. It can be shown that: if P_B is the optimal Bayesian error probability, then:

$$P_B \leq P_{NN} \leq P_B\left(2 - \frac{M}{M-1} P_B\right) \leq 2P_B$$

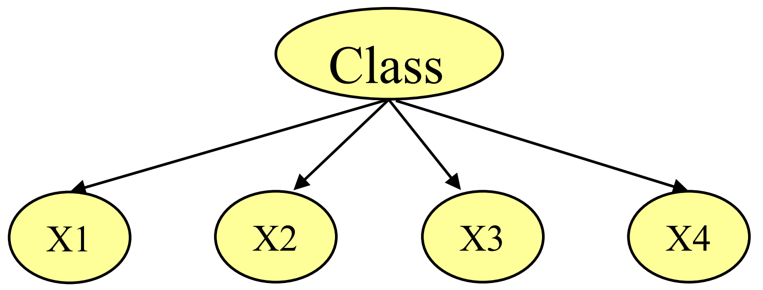
➔Voronoi tesselation

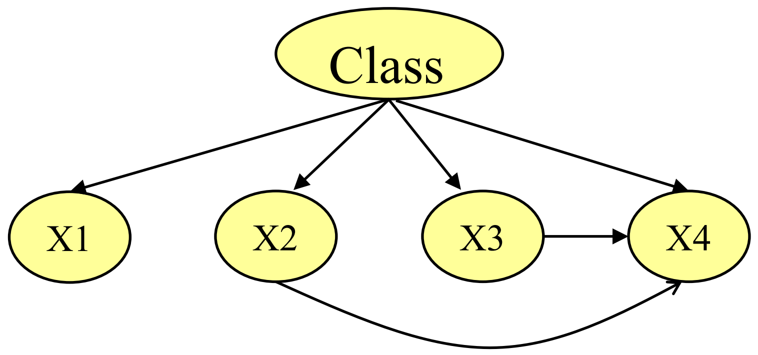


$$R_i = \left\{ \underline{x} : d(\underline{x}, \underline{x}_i) < d(\underline{x}, \underline{x}_j)\ i \neq j \right\}$$

From Naive Bayes to Bayes Networks

- ➔ **Naive Bayes could be a strong simplification in some cases**
  - ⇨ All the features are assumed to be independent
- ➔ **From joint multivariate distribution ot naive assumptions, we can consider intermediate cases**
  - ⇨ Consider only a few dependencies between features





- ➔ **Max P(c/x1, ..x4)=max P(c, x1, .., x4)=**
  - ⇨ Naive: P(c)p(x1)p(x2)p(x3)p(x4)
  - ⇨ Network: P(c)p(x1) p(x2)p(x2, x3, x4)=P(c)p(x1)p(x4/x3, x2)p(x3)p(x2)

Bayesian Networks

➔Bayes Probability Chain Rule

$$p(x_1, x_2, \ldots, x_\lambda) = p(x_\lambda \mid x_{\lambda-1}, \ldots, x_1) \cdot p(x_{\lambda-1} \mid x_{\lambda-2}, \ldots, x_1) \cdot \ldots$$

$$\ldots \cdot p(x_2 \mid x_1) \cdot p(x_1)$$

⇨ Assume now that the conditional dependence for each x_i is limited to a subset of the features appearing in each of the product terms. That is:

$$p(x_1, x_2, \ldots, x_\lambda) = p(x_1) \cdot \prod_{i=2}^{\lambda} p(x_i \mid A_i)$$

where

$$A_i \subseteq \{x_{i-1}, x_{i-2}, \ldots, x_1\}$$

⇨ For example, if ℓ=6, then we could assume:

$$p(x_6 \mid x_5, \ldots, x_1) = p(x_6 \mid x_5, x_4)$$

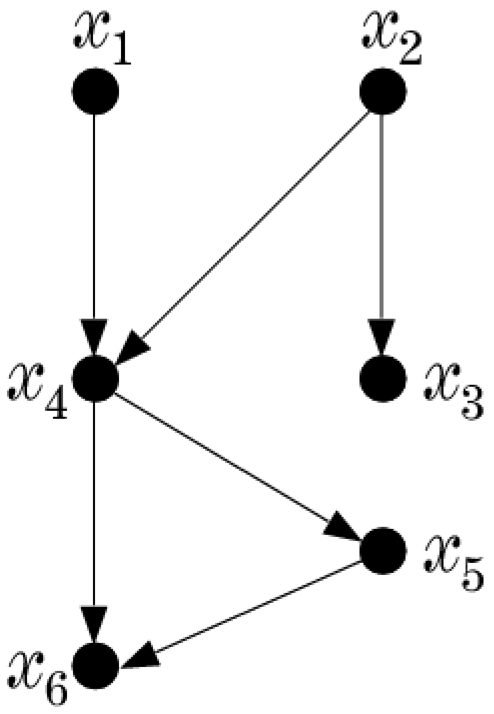
Then:

$$A_6 = \{x_5, x_4\} \subseteq \{x_5, \ldots, x_1\}$$

⇨ The above is a generalization of the Naïve – Bayes. For the Naïve – Bayes the assumption is:

$$A_i = \emptyset, \text{ for } i=1, 2, \ldots, \ell$$

⇨ A graphical way to portray <u>conditional dependencies</u> is given below



According to this figure we have that:

- x_6 is conditionally dependent on x_4, x_5.
- x_5 on x_4
- x_4 on x_1, x_2
- x_3 on x_2
- x_1, x_2 are conditionally **independent** on other variables.

For this case:

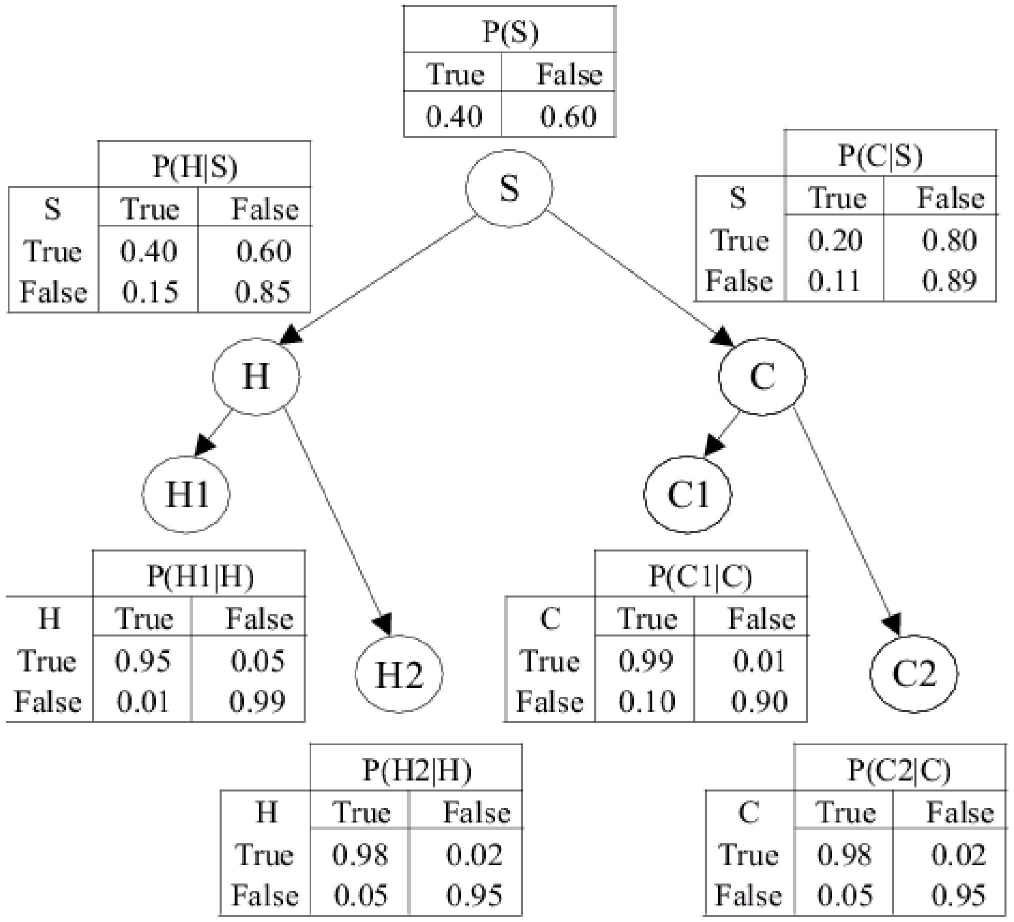
$$p(x_1, x_2, \ldots, x_6) = p(x_6 \mid x_5, x_4) \cdot p(x_5 \mid x_4) \cdot p(x_4 \mid x_1, x_2) \cdot p(x_3 \mid x_2) \cdot p(x_2) \cdot p(x_1)$$

➔ Bayesian Networks

- ⇨ **Definition:** A Bayesian Network is a directed acyclic graph (DAG) where the nodes correspond to random variables. Each node is associated with a set of conditional probabilities (densities), $p(x_i|A_i)$, where x_i is the variable associated with the node and A_i is the set of its parents in the graph.

- ⇨ A Bayesian Network is specified by:
  - → The marginal probabilities of its root nodes.
  - → The conditional probabilities of the non-root nodes, given their parents, for ALL possible combinations.

⇨ The figure below is an example of a Bayesian Network corresponding to a paradigm from the medical applications field.



P(S)	
True	False
0.40	0.60

	P(H|S)	
S	True	False
True	0.40	0.60
False	0.15	0.85

	P(C|S)	
S	True	False
True	0.20	0.80
False	0.11	0.89

	P(H1|H)	
H	True	False
True	0.95	0.05
False	0.01	0.99

	P(C1|C)	
C	True	False
True	0.99	0.01
False	0.10	0.90

	P(H2|H)	
H	True	False
True	0.98	0.02
False	0.05	0.95

	P(C2|C)	
C	True	False
True	0.98	0.02
False	0.05	0.95

This Bayesian network models conditional dependencies for an example concerning smokers (S), tendencies to develop cancer (C) and heart disease (H), together with variables corresponding to heart (H1, H2) and cancer (C1, C2) medical tests.

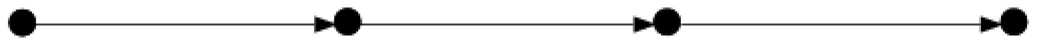
- ⇨ Once a DAG has been constructed, the joint probability can be obtained by multiplying the marginal (root nodes) and the conditional (non-root nodes) probabilities.

- ⇨ **Training**: Once a topology is given, probabilities are estimated via the training data set. There are also methods that learn the topology.

- ⇨ **Probability Inference**: This is the most common task that Bayesian networks help us to solve efficiently. Given the values of some of the variables in the graph, known as evidence, the goal is to compute the conditional probabilities for some of the other variables, given the evidence.

➔Example: Consider the Bayesian network of the figure:

$P(x1)=0.60$	$P(y1 \mid x1)=0.40$	$P(z1 \mid y1)=0.25$	$P(w1 \mid z1)=0.45$
	$P(y1 \mid x0)=0.30$	$P(z1 \mid y0)=0.60$	$P(w1 \mid z0)=0.30$



$P(x0)=0.40$	$P(y0 \mid x1)=0.60$	$P(z0 \mid y1)=0.75$	$P(w0 \mid z1)=0.55$
	$P(y0 \mid x0)=0.70$	$P(z0 \mid y0)=0.40$	$P(w0 \mid z0)=0.70$
	$P(y1)=0.36$	$P(z1)=0.47$	$P(w1)=0.37$
	$P(y0)=0.64$	$P(z0)=0.53$	$P(w0)=0.63$

a) If x is measured to be $x=1$ ($x1$), compute $P(w=0|x=1)$ [$P(w0|x1)$].

b) If w is measured to be $w=1$ ($w1$) compute $P(x=0|w=1)$ [$P(x0|w1)$].

- ⇨ For a), a set of calculations are required that propagate from node x to node w. It turns out that $P(w0|x1) = 0.63$.

- ⇨ For b), the propagation is reversed in direction. It turns out that $P(x0|w1) = 0.4$.

- ⇨ In general, the required inference information is computed via a combined process of “message passing” among the nodes of the DAG.

➔Complexity:

- ⇨ For singly connected graphs, message passing algorithms amount to a complexity linear in the number of nodes.