PATTERN RECOGNITION

Lab 2, Bayesian Decision Theory
Random Number Generation

Dr. Hongjian Shi
Southern University of Science and Technology
Email: shihj@sustc.edu.cn
Office Hours: By appointment, 1102 Intelligence Park A7

Bayesian decision rule

- Bayesian *decision rule*
  - If P(ω1) > P(ω2) , decide ω1; else decide ω2
  - What is the possible error: minimum(P(ω1), P(ω2))
  - More information can be obtained: different fish have obvious lightness measurement *x* as a *continuous* variable, its distribution in terms of class ω1 is expressed as

    $p(x|\omega 1)$ – *class-conditional probability density* function

    -- probability density function for *x*, given class nature is ω1 is not a probability

    -- the distribution of x depends on it

    -- what is the physical meaning?

Bayesian formula

- *Bayes' formula*
  - Probability density of a pattern in ωj and having feature value *x*:
    $p(\omega_j, x) = P(\omega_j|x)p(x) = p(x|\omega_j)P(\omega_j)$
  - $P(\omega_j|x) = \frac{p(x|\omega_j)P(\omega_j)}{p(x)}$ - *Bayes' formula*
  - $p(x) = \sum_{j=1}^{2} p(x|\omega_j)P(\omega_j)$
  - $posterior = \frac{likelihood\ X\ prior}{evidence}$
  - Converting a prior probability to a posterior probability
  - $p(x|\omega_j)$ – *likelihood* of ωj with respect to x
  - p(x) is called the *evidence*

Rewriting Bayes' Decision Rule

- How can we make a decision after observing the value of x?

$$\text{Decide} \begin{cases} w_1 & \text{if } P(w_1|x) > P(w_2|x) \\ w_2 & \text{otherwise} \end{cases}$$

- Rewriting the rule gives

$$\text{Decide} \begin{cases} w_1 & \text{if } \frac{p(x|w_1)}{p(x|w_2)} > \frac{P(w_2)}{P(w_1)} \\ w_2 & \text{otherwise} \end{cases}$$

- Note that, at every x, $P(w_1|x) + P(w_2|x) = 1$.

Probability error

- Bayes' decision rule (two class classification):
  Decide $\omega 1$ if $P(\omega 1/x) > P(\omega 2/x)$; otherwise decide $\omega 2$

- Probability error from the Bayes' decision rule:
  $P(error/x) = P(\omega 1/x)$ if we decide $\omega 2$
  $= P(\omega 2/x)$ if we decide $\omega 1$.
  $\rightarrow P(error/x) = \min [P(\omega 1/x), P(\omega 2/x)]$

- $P(error) = \int_{-\infty}^{\infty} P(error, x)\, dx = \int_{-\infty}^{\infty} P(error/x)\, p(x) dx$

- How to minimize this error?

Random

- Truly Random
  - Exhibiting true randomness
- Pseudorandom
  - Appearance of randomness but having a specific repeatable pattern
- Quasi-random
  - Having a set of non-random numbers in a randomized order

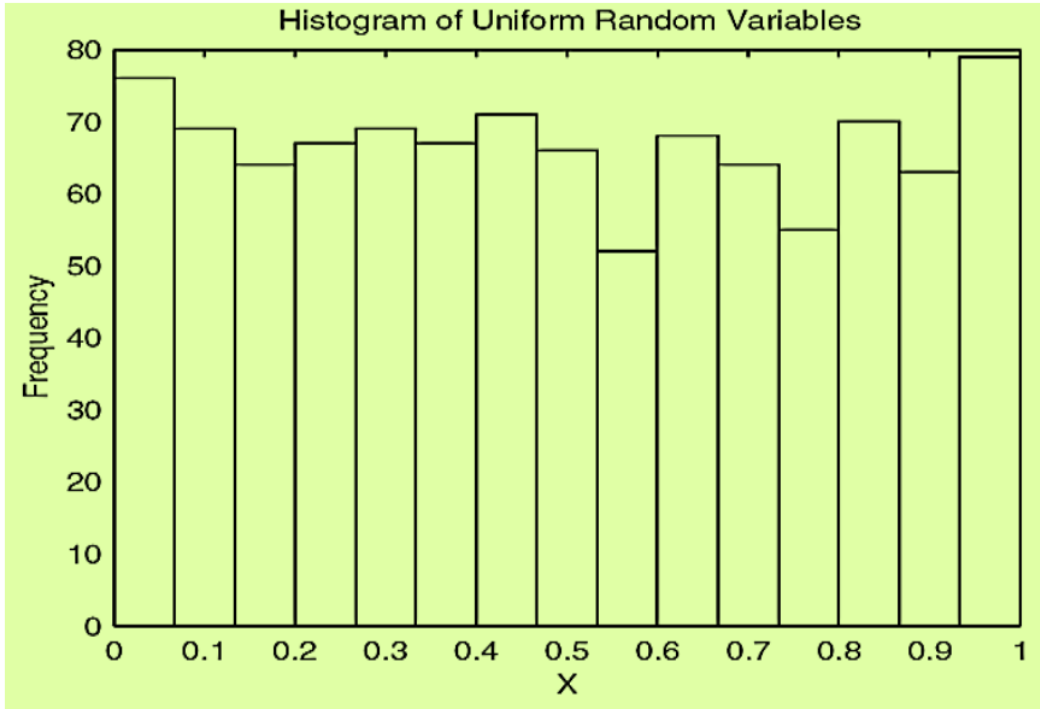
Seeds

- In your experimentation, you can generate 100 streams each with a different seed point in order to:
  - avoid duplicate streams of random numbers,
  - get a general idea of the behavior of the random number generator on your workstation.
- Seeds can be obtained from:
  - A. M. Law and W. D. Kelton, *Simulation Modeling & Analysis,* 2nd Edition, McGraw-Hill, NewYork, 1991.

```
Seeds = [ 1, ...
 1973272912, 281629770,  20006270,1280689831,2096730329,1933576050,...
  913566091, 246780520,1363774876, 604901985,1511192140,1259851944,...
  824064364, 150493284, 242708531,  75253171,1964472944,1202299975,...
  233217322,1911216000, 726370533, 403498145, 993232223,1103205531,...
  762430696,1922803170,1385516923,  76271663, 413682397, 726466604,...
  336157058,1432650381,1120463904, 595778810, 877722890,1046574445,...
   68911991,2088367019, 748545416, 622401386,2122378830, 640690903,...
 1774806513,2132545692,2079249579,  78130110, 852776735,1187867272,...
 1351423507,1645973084,1997049139, 922510944,2045512870, 898585771,...
  243649545,1004818771, 773686062, 403188473, 372279877,1901633463,...
  498067494,2087759558, 493157915, 597104727,1530940798,1814496276,...
  536444882,1663153658, 855503735,  67784357,1432404475, 619691088,...
  119025595, 880802310, 176192644,1116780070, 277854671,1366580350,...
 1142483975,2026948561,1053920743, 786262391,1792203830,1494667770,...
 1923011392,1433700034,1244184613,1147297105, 539712780,1545929719,...
  190641742,1645390429, 264907697, 620389253,1502074852, 927711160,...
  364849192,2049576050, 638580085, 547070247 ];
```

In Matlab

```
% Obtain a vector of uniform random variables in (0,1).
x = rand(1,1000);
% Do a histogram to plot.
% First get the height of the bars.
[N,X] = hist(x,15);
% Use the bar function to plot.
bar(X,N,1,'w')
title('Histogram of Uniform Random Variables')
xlabel('X')
ylabel('Frequency')
```

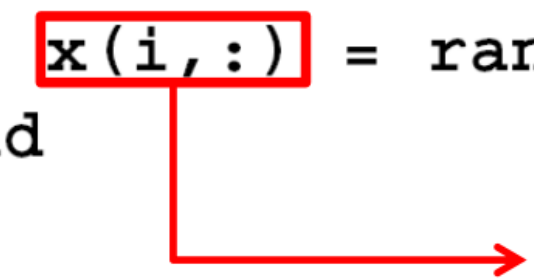


Notes:

- The function *rand* with no arguments returns a single instance of the random variable *U*.
- To get an array *mxn* of uniform variates, you can use the syntax **rand(m,n).**
- If you use **rand(n),** then you get an *nxn* matrix.

- The seed or the state of the generator is reset to the default when Matlab starts up, so the same sequencyes of random variables are generated whenever you start Matlab.
- If you call the function using **rand('state',0),** then MATLAB resets the generator to the initial state.
- If you want to specify another state, then use the syntax **rand('state',j)** to set the generator to the *j*-th state.
- You can obtain the current state using **S = rand('state'),** where **S** is a 35 element vector. To reset the state to this one, use **rand('state',S)**.

```
% Generate 3 random samples of size 5.
x = zeros(3,5);   % Allocate the memory.
for i = 1:3
    rand('state',i) % set the state
    x(i,:) = rand(1,5);
end
```



```
    0.9528    0.7041    0.9539    0.5982    0.8407
    0.8752    0.3179    0.2732    0.6765    0.0712
    0.5162    0.2252    0.1837    0.2163    0.4272
```

Lab Exercise 1 (50%)

Lab Exercise CH2-1:

- 2-1.1 Using MATLAB to redo p(x|ω1) and p(x|ω2) in Example 1.
- 2-1.2 Find P(ω1|x) and P(ω2|x) for all x and draw the figure as left using MATLAB.

Lab Exercise CH2-2

Two exclusive categories $\omega 1, \omega 2$:

P($\omega 1$) = 2/3, P($\omega 2$) = 1/3, P(x| $\omega 1$) => N(2, 0.5) (normal distribution),

P(x| $\omega 2$) => N(1.5, 0.2). Select the optimal decision using MATLAB.

Lab Exercise CH2-3

Feature space Ω = {$\omega 1, \omega 2$}, P($\omega 1$) = 2/3, P($\omega 2$) = 1/3, P(x| $\omega 1$) => N(2, 0.5),

P(x| $\omega 2$) => N(1.5, 0.2). $\lambda = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$.

Select the optimal decision using MATLAB.

N(μ, σ) = $\frac{1}{\sqrt{2\pi}\sigma} e^{-1/2((x-\mu)/\sigma)^2}$

Lab Exercise 1 (50%)

Random number generation

(a) Generate three streams (1D, 2D and 3D) of random numbers with 1,000 samples, you may use the Matlab command rand.

(b) Visualize the generated samples, you may use a scatter plot.

(c) Compute the histogram of the three streams, then normalize them to become a probability density function (pdf).

(d) Visualize the pdf's of the three streams. Are the samples uniformly distributed? Do the pdf's represent a standard uniform distributions? Comment.

OpenCV

- Languages for OpenCV – C, C++, Python, MATLAB, Java
- Operating systems – Windows, Linux, Mac OS, iOS, Android
- Cross platforms – OpenCL, CUDA, OpenGL, VTK, ITK etc. (compatibility is needed or overcome)
- Started from 2000 to now – including almost everything such as CUDA, VTK 3D, Computer Graphics, 2D and 3D calibration, audio and video etc.
- Web sites for questions – www.stackoverflow.com; www.google.com; or www.baidu.com

OpenCV Structure

- Lib indicates the corresponding operations, computing, analysis algorithm
- Closing with d indicates debug version, otherwise release version
  - Core – main module (basic structures, operations, drawing, clustering etc.)
  - Imgproc – image processing module
  - Video – video analysis and analytics module
  - Photo – image computing, restoration, noise reduction
  - Feature2d – 2D feature module
  - Calib3d – camera positioning and 3D reconstruction module
  - Objdetect – object detection module
  - Ml – machine learning module
  - Highgui – high level graphic interface module
  - Gpu – graphic processing unit module (for fast computing)

OpenCV header files (interface)

- #ifndef __OPENCV_ALL_HPP__
- #define __OPENCV_ALL_HPP__
- #include "opencv2/core/core_c.h"
- #include "opencv2/core/core.hpp"
- #include "opencv2/flann/miniflann.hpp"
- #include "opencv2/imgproc/imgproc_c.h"
- #include "opencv2/imgproc/imgproc.hpp"
- #include "opencv2/photo/photo.hpp"
- #include "opencv2/video/video.hpp"
- #include "opencv2/features2d/features2d.hpp"
- #include "opencv2/objdetect/objdetect.hpp"
- #include "opencv2/calib3d/calib3d.hpp"
- #include "opencv2/ml/ml.hpp"
- #include "opencv2/highgui/highgui_c.h"
- #include "opencv2/highgui/highgui.hpp"
- #include "opencv2/contrib/contrib.hpp"
- #endif

Or use using namespace cv;

Installation OpenCV 3.1.0 (VS 2013)

- Download opencv-3.1.0.exe (x64/x86) from the official web site
  http://sourceforge.net/projects/opencvlibrary/files/opencv-win/
  and extract it to Program Files\opencv310
- Set environment variable (if you have administrator right), otherwise copy all dll files into run time directory (Debug/Release)
  - Press right key on your computer, select
  - Advanced System Setting -> Environment variable -> Create
    - Fill in Path
    - D:\Program Files\opencv310\opencv\build\x64\vc12\bin
- Visual Studio configuration
  - Project->Properties, select Configuration Properties ->C/C++->General
    - Additional Include Directories: xxx\build\include
  - Project->Properties, select Configuration Properties ->Linker -> General
    - Additional Library Directories: xxxx\x64\vc12
  - Project->Properties, select Configuration Properties ->Linker -> Input
    - Copy all opencv libraries into Additional Dependencies such as opencv_core310d.lib

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
// 使用标准的cv库
using namespace cv;

int main( )
{
        // 读取源图像并转化为灰度图像
        Mat srcImage = cv::imread("..\\images\\flower.jpg");
        // 判断文件是否读入正确
        if( !srcImage.data )
            return 1;
        // 图像显示
         imshow("srcImage", srcImage);
        // 等待键盘键入
        waitKey(0);
        return 0;
}
```