ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ

Αναγνώρισης Προτύπων και Μηχανικής Μάθησης

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ #2

Διδάσκων Καθηγητής: Δαλιάνης Σωτήριος

ΕΚΠ. ΕΤΟΣ 2022-2023

1 Σκοπός της Άσκησης

Η άσκηση αυτή αποσκοπεί στο να εξοικειώσει τους φοιτητές στην χρήση του περιβάλλοντος MATLAB σε εφαρμογές Αναγνώρισης Προτύπων και Μηχανικής Μάθησης, με την βοήθεια χαρακτηριστικών παραδειγμάτων από τον προγραμματισμό. Έμφαση δίδεται σε εργαλεία στατιστικής ανάλυσης, εισαγωγής και επεξεργασίας δεδομένων και αναγνώρισης προτύπων.

2 Υπόβαθρο – Προετοιμασία

- Βασικές γνώσεις προγραμματισμού σε περιβάλλον MATLAB ή OCTAVE.
- Κατανόηση βασικών αρχών στατιστικής ανάλυσης.
- Εγκατάσταση του περιβάλλοντος MATLAB.
- Εγκατάσταση του Statistics and Machine Learning Toolbox του περιβάλλοντος MATLAB.
- Εγκατάσταση του Curve Fitting Toolbox Toolbox του περιβάλλοντος MATLAB.

3 Εκτέλεση της Άσκησης

Η παρούσα άσκηση χωρίζεται σε τρία μέρη με επιμέρους ασκήσεις. Να τοποθετήσετε τον κώδικα Matlab σε ξεχωριστά αρχεία scripts με όνομα τον αριθμό της άσκησης (------.m). Εκτελέσετε τον κώδικα και κάντε προσαρμογές όπου χρειάζεται. Αποθηκεύσατε τα αποτελέσματα και τα γραφήματα. Σε ασκήσεις με μεγάλο αριθμό δεδομένων και μεταβλητών συνιστάτε να αποθηκεύτε το workspace σε αρχείο .mat.

Η παρουσίαση της άσκησης γίνεται σε ένα ενιαίο αρχείο world doc ή pdf που περιλαμβάνει σύντομη περιγραφή του προβλήματος, τον κώδικα που χρησιμοποιήσατε, τα αποτελέσματα και τα γραφήματα.

Τα αρχεία ανεβαίνουν στο e-class μέχρι την καταληκτική ημερομηνία παράδοσης της άσκησης.

Μέρος Α. ΤΟ ΠΕΡΙΒΑΛΛΟΝ MATLAB ΓΙΑ ΕΦΑΡΜΟΓΕΣ Α.Π.Μ.Μ.

Α1. Άσκηση #2 Κλήση συναρτήσεων

Διαθέτουμε 40 νομίσματα των 50 λεπτών, 40 νομίσματα του 1€, 40 νομίσματα των 2€, και 40 χαρτονομίσματα των 5€. Να αναπτύξετε πρόγραμμα στο MATLAB το οποίο:

i) Θα υπολογίζει και θα εμφανίζει όλους τους δυνατούς συνδυασμούς με τους οποίους μπορούμε να χρησιμοποιήσουμε 40 συνολικά νομίσματα-χαρτονομίσματα (ανεξαρτήτως αξίας) έτσι ώστε η συνολική τους αξία να είναι 40€.

ii) Θα υπολογίζει και θα εμφανίζει το πλήθος των δυνατών συνδυασμών.

Ενδεικτική λύση

Για την αλγοριθμική επίλυση του ανωτέρω παραδείγματος απεικονίζουμε με τις ακόλουθες μεταβλητές τα εξής στοιχεία του προβλήματος:
x το πλήθος των νομισμάτων με αξία 0.5 ευρώ. Αυτή η τιμή μπορεί να λάβει τιμές **από 0 έως 40.**
y το πλήθος των νομισμάτων με αξία 1.0 ευρώ. ομοίως
z το πλήθος των νομισμάτων με αξία 2.0 ευρώ. ομοίως
w το πλήθος των χαρτονομισμάτων με αξία 5 ευρώ. ομοίως
Από ότι φαίνεται, με την χρήση βρόχων *for* μπορούμε να αναπαραστήσουμε τις δυνητικές τιμές του πλήθους των νομισμάτων. Για παράδειγμα, η εντολή *for x=0:40* δηλώνει ότι η μεταβλητή x (*το πλήθος των νομισμάτων με αξία 0.5 ευρώ*) μπορεί να είναι από 0 (δηλαδή καθόλου νομίσματα με αξία 0.5 ευρω) μέχρι 40 (όλα τα νομίσματα με αξία 0.5 ευρώ).
Συνεχίζοντας, επιθυμούμε να υπολογίσουμε το πλήθος των νομισμάτων ώστε να έχουμε 40 **συνολικά** νομίσματα-χαρτονομίσματα. Η προφανής έκφραση για το **συνολικό** πλήθος των νομισμάτων είναι: x+y+z+w. Επίσης η προφανής έκφραση για την αναπαράσταση της **αξίας** του παραπάνω **συνολικού** πλήθους είναι:
x*0.5 +y*1+z*2+w*5. Ο κώδικας που ακολουθεί δίνει την προτεινόμενη λύση:

```
clear all , clc, close all,
% x το πλήθος των νομισμάτων με αξία 0.5 ευρώ.
% y το πλήθος των νομισμάτων με αξία 1.0 ευρώ.
% z το πλήθος των νομισμάτων με αξία 2.0 ευρώ.
% w το πλήθος των χαρτονομισμάτων με αξία 5 ευρώ.
% Αρχικοποίηση του πλήθους των συνδυασμών που ικανοποιούν τις
συνθήκες
PS=0;
for x=0:40
for y=0:40
for z=0:40
for w=0:40
syn_plithos_nom=x+y+z+w;
syn_aksia_nom=x*0.5+y*1+z*2+w*5;
if syn_plithos_nom==40 && syn_aksia_nom==40
PS=PS+1;
disp(['Combination #' num2str(PS) ': ' num2str(x) 'x0.5E ' num2str(y)
'x1E ' num2str(z) 'x2E ' num2str(w) 'x5E '])
end
end
end
end
end
PS
```

Αποθηκεύσατε το ανωτέρω αρχείο με όνομα ***ask1_1.m***. Στην συνέχεια εκτελέστε το με τους ακόλουθους τρόπους:
i) Κλήση του ονόματός του από την γραμμή εντολών. **>> *ask1_1***
ii) Αντιγράψτε όλες τις εντολές του αρχείου και επικολλήστε τις στο παράθυρο εντολών του matlab.
γ) Υλοποίηση της άσκησης με συνάρτηση (function) του matlab :
Στην περίπτωση που θέλουμε να υλοποιήσουμε την άσκηση με χρήση συνάρτησης θα πρέπει να δούμε τις πιθανές εισόδους και τις εξόδους αυτής. Για παράδειγμα, θα μπορούσαμε να δώσουμε ως είσοδο τις αξίες των 4 νομισμάτων-χαρτονομισμάτων. Έτσι ένας πιθανός ορισμός - κλήση της συνάρτησης θα ήταν:

```
function PS=ask1_1f(val1, val2, val3, val4)
PS=0;
for x=0:40
for y=0:40
for z=0:40
for w=0:40
syn_plithos_nom=x+y+z+w;
syn_aksia_nom=x*val1+y*val2+z*val3+w*val4;
if syn_plithos_nom==40 && syn_aksia_nom==40
PS=PS+1;
disp(['Combination #' num2str(PS) ': ' num2str(x) 'x0.5E ' num2str(y)
'x1E ' num2str(z) 'x2E ' num2str(w) 'x5E '])
end
end
end
end
end
```

Αποθηκεύουμε το ανωτέρω πρόγραμμα με όνομα `askl1_1f.m` και το καλούμε από την γραμμή εντολών ως εξής για να αναθέσουμε στις μεταβλητές val1, val2, val3, val4 τις τιμές 0.5, 1, 2, 5 αντίστοιχα:

```
>> PS=askl_1f(0.5, 1, 2, 5);
>> PS
```

Α2. Χειρισμός διανυσμάτων και πινάκων.

Να εκτελέσετε επιμέρους κώδικες Matlab σχετικά με τον χειρισμό πινάκων που θα βρείτε στο αρχείο ΄¨Eisagogi sto matlab¨ σελ. 1-25. Βεβαιωθείτε ότι κατανοείτε την χρήση των συναρτήσεων σχετικά με τη Ταξινόμηση, Μέση και Ενδιάμεση τιμή, Διασπορά και Τυπική απόκλιση, Παραγωγή τυχαίων αριθμών.

Α3. Άσκηση #2 Παραγωγή τυχαίων αριθμών.

Να παράγετε τυχαία δεδομένα από μια διακριτή και μια συνεχή κατανομή πιθανότητας σύμφωνα με το παράδειγμα και να αποθηκεύσετε τα γραφήματα αυτών.

As an example we generate N = 1000 random numbers given by a binomial distribution with n = 9 trials and p = 0:8. Thus each of the 100 random number will be an integer between 0 and 9. Find the result of the code below showing the result in Figures.

```
N = 1000; data = binornd(9, 0.8, N,1); % generate the random numbers
[height,centers] = hist(data,unique(data)) % data for the histogram
bar(centers,height/sum(height))
xlabel('value'); ylabel('experimental probability')
title('Binomial distribution with n=9, p=0.8')
```

Commands to Work with Probability Distributions
rand() uniform distribution
randi() random integers
randn() normal distribution
rande() exponentially distributed
randp() Poisson distribution
randg() gamma distribution
normrnd() normal distribution
binornd() binomial distribution
exprnd() exponential distribution
trnd() Student-t distribution
discrete rnd() discrete distribution

Α4. Άσκηση #2 Data Sets.

Στον πίνακα παρουσιάζεται μία λίστα με συλλογές δεδομένων ή μετρήσεων που είναι ενσωματωμένες στο Matlab και μπορείτε να τις χρησιμοποιήσετε με άδεια Creative Commons. Χρησιμοποιούνται συχνά για την αξιολόγηση μοντέλων μηχανικής μάθησης. Σύνδεσμος:

https://www.mathworks.com/help/matlab/import_export/matlab-example-data-sets.html

Παρόμοια επίσης μια συλλογή εικόνων από την Microsoft.

https://www.microsoft.com/en-us/download/details.aspx?id=52644

Επιπλέον δεδομένα βρίσκονται στη σελίδα του εργαστηρίου στο στο e-class. Φορτώστε στο Matlab 4 data sets με διαφορετικά χαρακτηριστικά ως προς τον τύπο και τον αριθμό μεταβλητών. Δοκιμάστε να τα επεξεργαστείτε με το εργαλείο διαχείριση δεδομένων του Matlab.

File	Description of Data Set
`acetylene.mat`	Chemical reaction data with correlated predictors
`arrhythmia.mat`	Cardiac arrhythmia data from the UCI machine learning repository

File	Description of Data Set
batterysmall.mat	Sensor data (voltage, current, and temperature) and state of charge for a Li-ion battery; a subset of the data in [1]
carbig.mat	Measurements of cars, 1970–1982
carsmall.mat	Subset of carbig.mat. Measurements of cars, 1970, 1976, 1982
census1994.mat	Adult data from the UCI machine learning repository
cereal.mat	Breakfast cereal ingredients
cities.mat	Quality of life ratings for U.S. metropolitan areas
discrim.mat	A version of cities.mat used for discriminant analysis
examgrades.mat	Exam grades on a scale of 0–100
fisheriris.mat	Fisher's 1936 iris data
flu.mat	Google Flu Trends estimated ILI (influenza-like illness) percentage for various regions of the US, and CDC weighted ILI percentage based on sentinel provider reports
gas.mat	Gasoline prices around the state of Massachusetts in 1993
hald.mat	Heat of cement vs. mix of ingredients
hogg.mat	Bacteria counts in different shipments of milk
hospital.mat	Simulated hospital data
humanactivity.mat	Human activity recognition data of five activities: sitting, standing, walking, running, and dancing
imports-85.mat	1985 Auto Imports Database from the UCI repository
ionosphere.mat	Ionosphere dataset from the UCI machine learning repository
kmeansdata.mat	Four-dimensional clustered data
lawdata.mat	Grade point average and LSAT scores from 15 law schools
mileage.mat	Mileage data for three car models from two factories
moore.mat	Biochemical oxygen demand on five predictors
morse.mat	Recognition of Morse code distinctions by non-coders

File	Description of Data Set
nlpdata.mat	Natural language processing data extracted from the MathWorks® documentation
ovariancancer.mat	Grouped observations on 4000 predictors [2][3]
parts.mat	Dimensional run-out on 36 circular parts
polydata.mat	Sample data for polynomial fitting
popcorn.mat	Popcorn yield by popper type and brand
reaction.mat	Reaction kinetics for Hougen-Watson model
spectra.mat	NIR spectra and octane numbers of 60 gasoline samples
stockreturns.mat	Simulated stock returns

Μέρος Β. Άσκηση #1 Πιθανότητες & Τυχαίες μεταβλητές.

Υπόδειξη: Να μελετήσετε τα κεφάλαια 4 Data Reduction Commands & 5 Performing Linear Regression από το φυλλάδιο ‘’StatisticsWithMatlabOctave’’ στη σελίδα του εργαστηρίου στο e-class.

Β.1. Άσκηση #2 Υπολογισμόςμέσης τιμής, διαδποράς και σφαλμάτων για διαφορετικές κατανομές.

```
clear all; close all;clc;
mu = 4;
sigma =2;  % matlab uses sigma, not sigma^2
iter = [100 1000 10000];%% ari8mos deigmatwn
k=1; %% voi8itiki metavliti

for k=1:length(iter)
for i=1:100

%create gaussian samples
samples = normrnd(mu,sigma,iter(k),1);

%%%o sxediamos tis pdf k tou kanonikopoimenou istogramatos einai se
sxolia
%%%epeidh to peirama epanalamvanete 100 fores...
% %create actual gaussian pdf
% x = mu-4*sigma:0.01:mu+4*sigma;
% y = normpdf(x,mu,sigma);
%
% %create and normalize histogram
% [n,xout] = hist(samples,30);
% bw = xout(2)-xout(1);   % column width
% n1 = n/sum(n.*bw);      % sum(n*bw) = area under the histogram
%
% %plot histogram and pdf
```

```
% bar(xout,n1)
% hold on
% plot(x,y,'-r','Linewidth',3);
% hold off
%%% evresi mesis timis k diasporas me mean kai var
Ey(i,k)=mean(samples);
s(i,k)=var(samples);
%% evresi mesis timis k diasporas vasei ml
Eyy(i,k)=(1/iter(k))*sum (samples);
ss(i,k)=(1/iter(k))*sum ((samples-Eyy(i,k)).^2);
end;

%%sfalma  mesis timis
ermt(:,k)=(1/100)*((Ey(:,k)-Eyy(:,k)).^2);
errmt(k)=mean(ermt(:,k));
%%sfalma diasporas
erd(:,k)=(1/100)*((s(:,k)-ss(:,k)).^2);
errd(k)=mean(erd(:,k));

k=k+1;
end;
```

Β.2. Άσκηση #2 Υπολογισμός μέσης τιμής, διασποράς και σφαλμάτων για διαφορετικές κατανομές.

```
clear all; close all; clc;

sigma=4;

iter = [100 1000 10000];%% ari8mos deigmatwn
k=1; %% voi8itiki metavliti

for k=1:length(iter)
for i=1:100

%% create rayleigh samples
samples = raylrnd(sigma,iter(k),1);

%%%o sxediamos tis pdf k tou kanonikopoimenou istogramatos einai se
sxolia
%%%epeidh to peirama epanalamvanete 100 fores...
%
% %create actual rayleigh pdf
% x =0:0.01:20;
% y = raylpdf(x,sigma);
%
% %create and normalize histogram
% [n,xout] = hist(samples,30);
% bw = xout(2)-xout(1);   % column width
% n1 = n/sum(n.*bw);      % sum(n*bw) = area under the histogram
%
% %plot histogram and pdf
% bar(xout,n1)
% hold on
% plot(x,y,'-r','Linewidth',3);
% hold off

%%% evresi mesis timis k diasporas me mean kai var
```

```
Ey(i,k)=mean(samples);
s(i,k)=var(samples);
%%% evresi mesis timis k diasporas vasei ml
sigmaa(i,k)=sqrt((1/(2*iter(k)))*sum(samples.^2));
Eyy(i,k)=sigmaa(i,k)*sqrt((pi/2));
ss(i,k)=((4-pi)/2)*(sigmaa(i).^2);
end;

%%sfalma mesis timis
ermt(:,k)=(1/100)*((Ey(:,k)-Eyy(:,k)).^2);
errmt(k)=mean(ermt(:,k));
%%sfalma diasporas
erd(:,k)=(1/100)*((s(:,k)-ss(:,k)).^2);
errd(k)=mean(erd(:,k));

k=k+1;
end;
```

Β.3. Άσκηση #2 Πολυδιάστατες κατανομές.

Να παράγετε $N = 500$ δισδιάστατα σημεία από μία κανονική κατανομή $\mathcal{N}(m, S)$, με μέση τιμή $m = [0,\ 0]^T$ και μήτρα συνδιασποράς $S = \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{bmatrix}$, για τις παρακάτω περιπτώσεις

(α) $\sigma_1^2 = \sigma_2^2 = 1,\ \sigma_{12} = 0$
(β) $\sigma_1^2 = \sigma_2^2 = 0.2,\ \sigma_{12} = 0$
(γ) $\sigma_1^2 = \sigma_2^2 = 2,\ \sigma_{12} = 0$
(δ) $\sigma_1^2 = 0.2,\ \sigma_2^2 = 2,\ \sigma_{12} = 0$
(ε) $\sigma_1^2 = 2,\ \sigma_2^2 = 0.2,\ \sigma_{12} = 0$
(στ) $\sigma_1^2 = \sigma_2^2 = 1,\ \sigma_{12} = 0.5$
(ζ) $\sigma_1^2 = 0.3,\ \sigma_2^2 = 2,\ \sigma_{12} = 0.5$
(η) $\sigma_1^2 = 0.3,\ \sigma_2^2 = 2,\ \sigma_{12} = -0.5$

Απεικονίστε γραφικά το παραπάνω σύνολο και σχολιάστε τα σχήματα των ομάδων που προκύπτουν.

Υπόδειξη: Για την παραγωγή του πρώτου συνόλου δεδομένων χρησιμοποιούμε τις παρακάτω εντολές:

```
randn('seed',0)  %Initialization of the randn function
m=[0 0]';
S=[1 0; 0 1];
N=500;
X=mvnrnd(m,S,N)';
```

όπου X είναι το μητρώο που περιέχει τα διανύσματα στις στήλες του.

Για τη γραφική απεικόνιση των δεδομένων του πρώτου συνόλου χρησιμοποιούμε τις παρακάτω εντολές:

```
figure(1), plot(X(1,:), X(2,:),'.');
figure(1), axis equal
figure(1), axis([-7 7 -7 7])
```

Για τα υπόλοιπα σύνολα εργαζόμαστε με παρόμοιο τρόπο.

Γ. Άσκηση #1 Παραμετρική προσαρμογή δεδομένων και στατιστικά μοντέλα.

Γ.1. Άσκηση #2 Προσαρμογή καμπυλών. Μέθοδος ελαχίστων τετραγώνων.

Η πολυωνυμική προσαρμογή ελαχίστων τετραγώνων υλοποιείται από τη συνάρτηση polyfit:
p=polyfit(x,y,n)
η οποία βρίσκει τους συντελεστές του πολυωνύμου ελαχίστων τετραγώνων p(x) βαθμού n που προσαρμόζεται στα σημεία (x(i), y(i)). Το αποτέλεσμα είναι ένα διάνυσμα-γραμμή με τους συντελεστές του πολυωνύμου σε φθίνουσα διάταξη.
Για να δώσουμε ένα γράφημα της παρεμβολής, πρέπει να παράγουμε σημεία στον άξονα των x και με βάση αυτά να παράγουμε τις τιμές του p(x) με τη συνάρτηση polyval. Η διαδικασία αυτή φαίνεται στο παρακάτω παράδειγμα:

Παράδειγμα Για τα δεδομένα (-1,4), (1,2.5), (2,4), (3,3.2), (4,5.1), (5,7.4) θα υπολογίσουμε τα πολυώνυμα ελαχίστων τετραγώνων βαθμών 2 και 5. Στη συνέχεια , σχεδιάζουμε τα γραφήματά τους σε μια ενιαία γραφική παράσταση.

```
» x=[-1,1,2,3,4,5] ;
» y=[4,2.5,4,3.2,5.1,7.4];
» p2=polyfit(x,y,2) %κατασκευή πολυώνυμο ελαχ. τετραγώνων 2ου βαθμού
p2 = 0.2845 -0.6081 3.1300
» p5=polyfit(x,y,5) %κατασκευή πολυώνυμο ελαχ. τετραγώνων 5ου βαθμού
p5 = -0.0943 1.1104 -4.1410 4.4229 3.4853 -2.2833
» xi=linspace(-1,5,100); %δημιουργία δεδομένων στον άξονα των x
» z2=polyval(p2,xi); %δημιουργία δεδομένων στον άξονα των y για το p2
» z5=polyval(p5,xi); %δημιουργία δεδομένων στον άξονα των y για το p5
» plot(x,y,'o',xi,z2,':',xi,z5) %σχεδίαση γραφημάτων για τα p2 και p5
%το γράφημα του p2 είναι με διακεκομμένη γραμμή
» xlabel('x'),ylabel('y=f(x)'), %ενδείξεις στους άξονες
» title('2nd and 5th order Curve Fitting') %έκδοση τίτλου γραφήματος
```

Γ.2. Άσκηση #2 Παρεμβολή.

Η παρεμβολή μονοδιάστατων δεδομένων υλοποιείται από τη εντολή interp1. Τα σημεία (xi, yi) που θα παρεμβληθούν πρέπει να αποθηκευθούν σε δύο διανύσματα x (για τα xi)και y (για τα yi).
• yi=interp1(x,y,xi): υπολογίζει στο διάνυσμα yi τις τιμές που αντιστοιχούν στα δεδομένα του διανύσματος xi και οι οποίες καθορίζονται έπειτα από εφαρμογή γραμμικής παρεμβολής στα σημεία (xi,yi) των διανυσμάτων x και y.
• yi=interp(x,y,xi, 'method'): προσδιορίζει την μέθοδο παρεμβολής που θα εφαρμοσθεί, όπου method
μπορεί να είναι:
-'linear' για γραμμική παρεμβολή
-'spline' για παρεμβολή με splines
-'cubic' για κυβική παρεμβολή. Εδώ απαιτείται οι τιμές του x να είναι ομοιόμορφα κατανεμημένες.

Σε όλες τις περιπτώσεις, οι τιμές του διανύσματος x θα πρέπει να δίνονται διατεταγμένες.
Παράδειγμα
Για τα δεδομένα του προηγούμενου παραδείγματος έχουμε:

```
» x=[-1,1,2,3,4,5] ;
» y=[4,2.5,4,3.2,5.1,7.4];
» xi=-1:0.1:5; %δημιουργία διαστήματος ομοιόμορφα κατανεμημένων τιμών
» y0=interp1(x,y,0,'spline') %υπολογισμός τιμής της παρεμβολής splines στο σημείο 0
ans = y0= 0.9453
» y0=interp1(x,y,0) %υπολογισμός τιμής της γραμμικής παρεμβολής στο σημείο 0
y0 =  3.2500
%Εφαρμογή τριών μεθόδων παρεμβολής
» Yspl=interp1(x,y,xi,'spline');
» Yl=interp1(x,y,xi,'linear');
» Yc=interp1(x,y,xi,'cubic');
» plot(x,y,'o',xi,Yl,xi,Yc,':',xi,Yspl) %σχεδίαση κοινού γραφήματος
» xlabel('x'),ylabel('Y=f(x)'),
» title('Spline interpolation versus Linear and Cubic interpolation')
```

Γ.3. Άσκηση #2 Οπτικοποίηση και επεξεργασία πολυδιάστατων δεδομένων.

This example shows how to visualize multivariate data using various statistical plots. Many statistical analyses involve only two variables: a predictor variable and a response variable. Such data are easy to visualize using 2D scatter plots, bivariate histograms, boxplots, etc. It's also possible to visualize trivariate data with 3D scatter plots, or 2D scatter plots with a third variable encoded with, for example color. However, many datasets involve a larger number of variables, making direct visualization more difficult. This example explores some of the ways to visualize high-dimensional data in MATLAB®, using Statistics and Machine Learning Toolbox™.

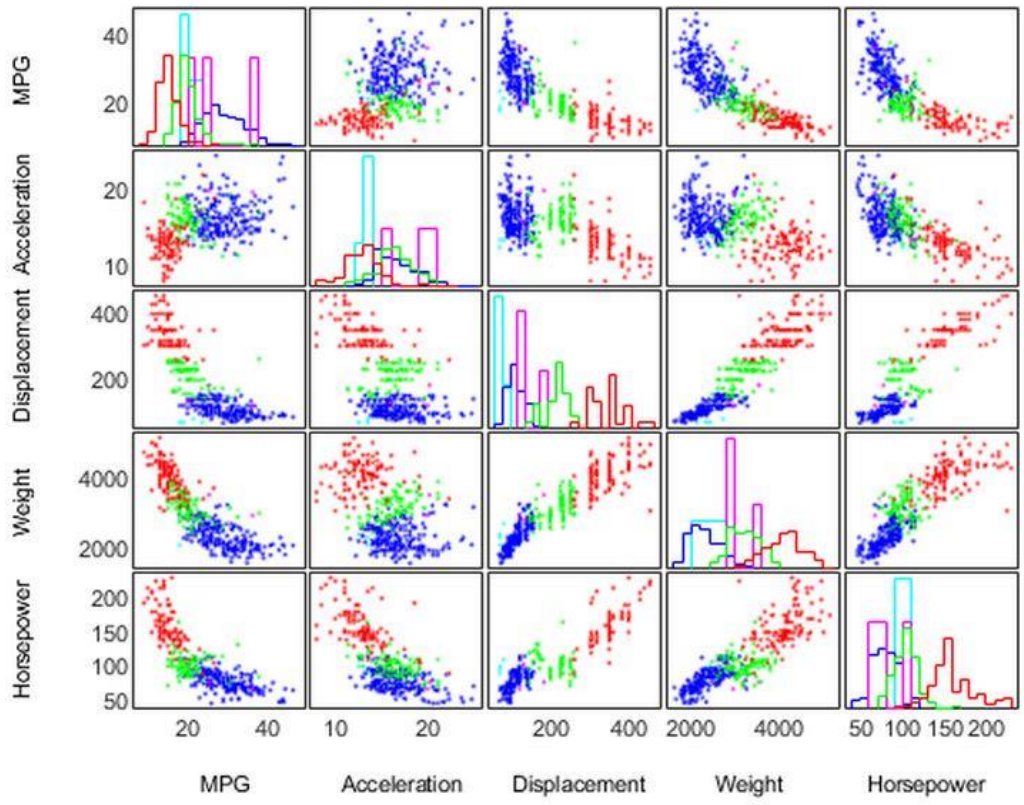
In this example, we'll use the `carbig` dataset, a dataset that contains various measured variables for about 400 automobiles from the 1970's and 1980's. We'll illustrate multivariate visualization using the values for fuel efficiency (in miles per gallon, MPG), acceleration (time from 0-60MPH in sec), engine displacement (in cubic inches), weight, and horsepower. We'll use the number of cylinders to group observations.

```
load carbig
X = [MPG,Acceleration,Displacement,Weight,Horsepower];
varNames = {'MPG'; 'Acceleration'; 'Displacement'; 'Weight'; 'Horsepower'};
```

Scatterplot Matrices

Viewing slices through lower dimensional subspaces is one way to partially work around the limitation of two or three dimensions. For example, we can use the `gplotmatrix`function to display an array of all the bivariate scatterplots between our five variables, along with a univariate histogram for each variable.

```
figure
gplotmatrix(X,[],Cylinders,['c' 'b' 'm' 'g' 'r'],[],[],false);
text([.08 .24 .43 .66 .83], repmat(-.1,1,5), varNames, 'FontSize',8);
text(repmat(-.12,1,5), [.86 .62 .41 .25 .02], varNames, 'FontSize',8,
'Rotation',90);
```



The points in each scatterplot are color-coded by the number of cylinders: blue for 4 cylinders, green for 6, and red for 8. There is also a handful of 5 cylinder cars, and rotary-engined cars are listed as having 3 cylinders. This array of plots makes it easy to pick out patterns in the relationships between pairs of variables. However, there may be important patterns in higher dimensions, and those are not easy to recognize in this plot.

Parallel Coordinates Plots

The scatterplot matrix only displays bivariate relationships. However, there are other alternatives that display all the variables together, allowing you to investigate higher-dimensional relationships among variables. The most straight-forward multivariate plot is the parallel coordinates plot. In this plot, the coordinate axes are all laid out horizontally, instead of using orthogonal axes as in the usual Cartesian graph. Each observation is represented in the plot as a series of connected line segments. For example, we can make a plot of all the cars with 4, 6, or 8 cylinders, and color observations by group.

```
Cyl468 = ismember(Cylinders,[4 6 8]);
parallelcoords(X(Cyl468,:), 'group',Cylinders(Cyl468), ...
               'standardize','on', 'labels',varNames)
```

The horizontal direction in this plot represents the coordinate axes, and the vertical direction represents the data. Each observation consists of measurements on five variables, and each measurement is represented as the height at which the corresponding line crosses each coordinate axis. Because the five variables have widely different ranges, this plot was made with standardized values, where each variable has been standardized to have zero mean and unit variance. With the color coding, the graph shows, for example, that 8 cylinder cars

typically have low values for MPG and acceleration, and high values for displacement, weight, and horsepower.

Even with color coding by group, a parallel coordinates plot with a large number of observations can be difficult to read. We can also make a parallel coordinates plot where only the median and quartiles (25% and 75% points) for each group are shown. This makes the typical differences and similarities among groups easier to distinguish. On the other hand, it may be the outliers for each group that are most interesting, and this plot does not show them at all.

```
parallelcoords(X(Cyl468,:), 'group',Cylinders(Cyl468), ...
               'standardize','on', 'labels',varNames, 'quantile',.25)
```

Appendix 1.

Dataset Arrays in the Variables Editor

- Modify Variable and Observation Names
- Reorder or Delete Variables
- Add New Data
- Sort Observations
- Select a Subset of Data
- Create Plots

Open Dataset Arrays in the Variables Editor

The MATLAB Variables editor provides a convenient interface for viewing, modifying, and plotting dataset arrays.

First, load the sample data set, `hospital`.

```
load hospital
```

The dataset array, `hospital`, is created in the MATLAB workspace.

The dataset array has 100 observations and 7 variables.

To open `hospital` in the Variables editor, click **Open Variable**, and select `hospital`.

The Variables editor opens, displaying the contents of the dataset array (only the first 10 observations are shown here).

In the Variables editor, you can see the names of the seven variables along the top row, and the observations names down the first column.

Modify Variable and Observation Names

You can modify variable and observation names by double-clicking a name, and then typing new text.

All changes made in the Variables editor are also sent to the command line.

The sixth variable in the data set, `BloodPressure`, is a numeric array with two columns. The first column shows systolic blood pressure, and the second column shows diastolic blood pressure. Click the arrow that appears on the right side of the

variable name cell to see the units and description of the variable. You can type directly in the units and description fields to modify the text. The variable data type and size are shown under the variable description.

Reorder or Delete Variables

You can reorder variables in a dataset array using the Variables editor. Hover over the left side of a variable name cell until a four-headed arrow appears.

After the arrow appears, click and drag the variable column to a new location.

The command for the variable reordering appears in the command line.

You can delete a variable in the Variables editor by selecting the variable column, right-clicking, and selecting **Delete Column Variable(s)**.

The command for the variable deletion appears in the command line.

Add New Data

You can enter new data values directly into the Variables editor. For example, you can add a new patient observation to the `hospital` data set. To enter a new last name, add a character vector to the end of the variable `LastName`.

The variable `Gender` is a `nominal` array. The levels of the categorical variable appear in a drop-down list when you double-click a cell in the `Gender` column. You can choose one of the levels previously used, or create a new level by selecting **New Item**.

You can continue to add data for the remaining variables.

To change the observation name, click the observation name and type the new name.

The commands for entering the new data appear at the command line.

Notice the warning that appears after the first assignment. When you enter the first piece of data in the new observation row—here, the last name—default values are assigned to all other variables. Default assignments are:

- `0` for numeric variables
- `<undefined>` for categorical variables
- `[]` for cell arrays

You can also copy and paste data from one dataset array to another using the Variables editor.

Sort Observations

You can use the Variables editor to sort dataset array observations by the values of one or more variables. To sort by gender, for example, select the variable `Gender`. Then click**Sort**, and choose to sort rows by ascending or descending values of the selected variable.

When sorting by variables that are cell arrays of character vectors or of nominal data type, observations are sorted alphabetically. For ordinal variables, rows are sorted by the ordering of the levels. For example, when the observations of `hospital` are sorted by the values in `Gender`, the females are grouped together, followed by the males.

To sort by the values of multiple variables, press **Ctrl** while you select multiple variables.

When you use the Variables editor to sort rows, it is the same as calling `sortrows`. You can see this at the command line after executing the sorting.

Select a Subset of Data

You can select a subset of data from a dataset array in the Variables editor, and create a new dataset array from the selection. For example, to create a dataset array containing only the variables `LastName` and `Age`:

1. Hold **Ctrl** while you click the variables `LastName` and `Age`.
2. Right-click, and select **New Workspace Variable from Selection** > **New Dataset Array**.

The new dataset array appears in the Workspace window with the name `hospital1`. The Command Window shows the commands that execute the selection.

You can use the same steps to select any subset of data. To select observations according to some logical condition, you can use a combination of sorting and selecting. For example, to create a new dataset array containing only males aged 45 and older:

1. Sort the observations of `hospital` by the values in `Gender` and `Age`, descending.
2. Select the male observations with age 45 and older.
3. Right-click, and select **New Workspace Variables from Selection** > **New Dataset Array**. The new dataset array, `hospital2`, is created in the Workspace window.
4. You can rename the dataset array in the Workspace window.

Create Plots

You can plot data from a dataset array using plotting options in the Variables editor. Available plot choices depend on the data types of variables to be plotted.

For example, if you select the variable `Age`, you can see in the **Plots** tab some plotting options that are appropriate for a univariate, numeric variable.

Sometimes, there are plot options for multiple variables, depending on their data types. For example, if you select both `Age` and `Gender`, you can draw box plots of age, grouped by gender.

Appendix 2.

Working with Probability Distributions

Probability distributions are theoretical distributions based on assumptions about a source population. The distributions assign probability to the event that a random variable has a specific, discrete value, or falls within a specified range of continuous values.

Statistics and Machine Learning Toolbox™ offers several ways to work with probability distributions.

- Use Probability Distribution Objects to fit a probability distribution object to sample data, or to create a probability distribution object with specified parameter values.
- Use Probability Distribution Functions to work with data input from matrices.
- Use Probability Distribution Apps and User Interfaces to interactively fit, explore, and generate random numbers from probability distributions. Available apps and user interfaces include:
  - The **Distribution Fitter** app
  - The **Probability Distribution Function** user interface
  - The Random Number Generation user interface (`randtool`)

For a list of distributions supported by Statistics and Machine Learning Toolbox, see Supported Distributions.

You can define a probability object for a custom distribution and then use the Distribution Fitter app or probability object functions, such as `pdf`, `cdf`, `icdf`, and `random`, to evaluate the distribution, generate random numbers, and so on. For details, see Define Custom Distributions Using the Distribution Fitter App. You can also define a custom distribution using a function handle and use the `mle` function to find maximum likelihood estimates. For an example, see Fit Custom Distribution to Censored Data.

Probability Distribution Objects

Probability distribution objects allow you to fit a probability distribution to sample data, or define a distribution by specifying parameter values. You can then perform a variety of analyses on the distribution object.

Create Probability Distribution Objects

Estimate probability distribution parameters from sample data by fitting a probability distribution object to the data using `fitdist`. You can fit a single specified parametric or nonparametric distribution to the sample data. You can also fit multiple distributions of the same type to the sample data based on grouping variables. For most distributions,`fitdist` uses maximum likelihood estimation (MLE) to estimate the distribution parameters from the sample data. For more information and additional syntax options, see`fitdist`.

Alternatively, you can create a probability distribution object with specified parameter values using `makedist`.

Work with Probability Distribution Objects

Once you create a probability distribution object, you can use object functions to:

- Compute confidence intervals for the distribution parameters (`paramci`).
- Compute summary statistics, including mean (`mean`), median (`median`), interquartile range (`iqr`), variance (`var`), and standard deviation (`std`).
- Evaluate the probability density function (`pdf`).
- Evaluate the cumulative distribution function (`cdf`) or the inverse cumulative distribution function (`icdf`).
- Compute the negative loglikelihood (`negloglik`) and profile likelihood function (`proflik`) for the distribution.
- Generate random numbers from the distribution (`random`).
- Truncate the distribution to specified lower and upper limits (`truncate`).

Save a Probability Distribution Object

To save your probability distribution object to a .MAT file:

- In the toolbar, click **Save Workspace**. This option saves all of the variables in your workspace, including any probability distribution objects.
- In the workspace browser, right-click the probability distribution object and select **Save as**. This option saves only the selected probability distribution object, not the other variables in your workspace.

  Alternatively, you can save a probability distribution object directly from the command line by using the `save` function. `save` enables you to choose a file name and specify the probability distribution object you want to save. If you do not specify an object (or other variable), MATLAB® saves all of the variables in your workspace, including any probability distribution objects, to the specified file name. For more information and additional syntax options, see `save`.

Analyze Distribution Using Probability Distribution Objects

Open Live Script

This example shows how to use probability distribution objects to perform a multistep analysis on a fitted distribution.

The analysis illustrates how to:

- Fit a probability distribution to sample data that contains exam grades of 120 students by using `fitdist`.
- Compute the mean of the exam grades by using `mean`.
- Plot a histogram of the exam grade data, overlaid with a plot of the pdf of the fitted distribution, by using `plot` and `pdf`.
- Compute the boundary for the top 10 percent of student grades by using `icdf`.
- Save the fitted probability distribution object by using `save`.

Load the sample data.

```
load examgrades
```

The sample data contains a 120-by-5 matrix of exam grades. The exams are scored on a scale of 0 to 100.

Create a vector containing the first column of exam grade data.

```
x = grades(:,1);
```

Fit a normal distribution to the sample data by using `fitdist` to create a probability distribution object.

```
pd = fitdist(x,'Normal')
pd =
  NormalDistribution

  Normal distribution
       mu = 75.0083   [73.4321, 76.5846]
    sigma =  8.7202   [7.7391, 9.98843]
```

`fitdist` returns a probability distribution object, `pd`, of the type `NormalDistribution`. This object contains the estimated parameter values, `mu` and `sigma`, for the fitted normal distribution. The intervals next to the parameter estimates are the 95% confidence intervals for the distribution parameters.

Compute the mean of the students' exam grades using the fitted distribution object, `pd`.

```
m = mean(pd)
m = 75.0083
```

The mean of the exam grades is equal to the `mu` parameter estimated by `fitdist`.

Plot a histogram of the exam grades. Overlay a plot of the fitted pdf to visually compare the fitted normal distribution with the actual exam grades.

```
x_pdf = [1:0.1:100];
y = pdf(pd,x_pdf);

figure
histogram(x,'Normalization','pdf')
line(x_pdf,y)
```

The pdf of the fitted distribution follows the same shape as the histogram of the exam grades.

Determine the boundary for the upper 10 percent of student exam grades by using the inverse cumulative distribution function (`icdf`). This boundary is equivalent to the value at which the cdf of the probability distribution is equal to 0.9. In other words, 90 percent of the exam grades are less than or equal to the boundary value.

```
A = icdf(pd,0.9)
A = 86.1837
```

Based on the fitted distribution, 10 percent of students received an exam grade greater than 86.1837. Equivalently, 90 percent of students received an exam grade less than or equal to 86.1837.

Save the fitted probability distribution, `pd`, as a file named `myobject.mat`.

```
save('myobject.mat','pd')
```

Probability Distribution Functions

You can also work with probability distributions using distribution-specific functions. These functions are useful for generating random numbers, computing summary statistics inside a loop or script, and passing a cdf or pdf as a function handle (MATLAB) to another function. You can also use these functions to perform computations on arrays of parameter values rather than a single set of parameters. For a list of supported probability distributions, see Supported Distributions.

Analyze Distribution Using Distribution-Specific Functions

Open Live Script

This example shows how to use distribution-specific functions to perform a multistep analysis on a fitted distribution.

The analysis illustrates how to:

- Fit a probability distribution to sample data that contains exam grades of 120 students by using `normfit`.
- Plot a histogram of the exam grade data, overlaid with a plot of the pdf of the fitted distribution, by using `plot` and `normpdf`.
- Compute the boundary for the top 10 percent of student grades by using `norminv`.
- Save the estimated distribution parameters by using `save`.

You can perform the same analysis using a probability distribution object.
See Analyze Distribution Using Probability Distribution Objects.

Load the sample data.

```
load examgrades
```

The sample data contains a 120-by-5 matrix of exam grades. The exams are scored on a scale of 0 to 100.

Create a vector containing the first column of exam grade data.

```
x = grades(:,1);
```

Fit a normal distribution to the sample data by using `normfit`.

```
[mu,sigma,muCI,sigmaCI] = normfit(x)
mu = 75.0083
sigma = 8.7202
muCI = 2×1

   73.4321
   76.5846

sigmaCI = 2×1

    7.7391
    9.9884
```

The `normfit` function returns the estimates of normal distribution parameters and the 95% confidence intervals for the parameter estimates.

Plot a histogram of the exam grades. Overlay a plot of the fitted pdf to visually compare the fitted normal distribution with the actual exam grades.

```
x_pdf = [1:0.1:100];
y = normpdf(x_pdf,mu,sigma);

figure
histogram(x,'Normalization','pdf')
line(x_pdf,y)
```

The pdf of the fitted distribution follows the same shape as the histogram of the exam grades.

Determine the boundary for the upper 10 percent of student exam grades by using the normal inverse cumulative distribution function. This boundary is equivalent to the value at which the cdf of the probability distribution is equal to 0.9. In other words, 90 percent of the exam grades are less than or equal to the boundary value.

```
A = norminv(0.9,mu,sigma)
A = 86.1837
```

Based on the fitted distribution, 10 percent of students received an exam grade greater than 86.1837. Equivalently, 90 percent of students received an exam grade less than or equal to 86.1837.

Save the estimated distribution parameters as a file named `myparameter.mat`.

```
save('myparameter.mat','mu','sigma')
```

Use Probability Distribution Functions as Function Handle

Open Live Script

This example shows how to use the probability distribution function `normcdf` as a function handle in the chi-square goodness of fit test (`chi2gof`).

This example tests the null hypothesis that the sample data contained in the input vector, `x`, comes from a normal distribution with parameters μ and σ equal to the mean (`mean`) and standard deviation (`std`) of the sample data, respectively.

```
rng('default') % For reproducibility
x = normrnd(50,5,100,1);
h = chi2gof(x,'cdf',{@normcdf,mean(x),std(x)})
h = 0
```

The returned result `h = 0` indicates that `chi2gof` does not reject the null hypothesis at the default 5% significance level.

This next example illustrates how to use probability distribution functions as a function handle in the slice sampler (`slicesample`). The example uses `normpdf` to generate a random sample of 2,000 values from a standard normal distribution, and plots a histogram of the resulting values.

```
rng('default') % For reproducibility
x = slicesample(1,2000,'pdf',@normpdf,'thin',5,'burnin',1000);
histogram(x)
```

The histogram shows that, when using `normpdf`, the resulting random sample has a standard normal distribution.

If you pass the probability distribution function for the exponential distribution pdf (`exppdf`) as a function handle instead of `normpdf`, then `slicesample` generates the 2,000 random samples from an exponential distribution with a default parameter value of μ equal to 1.

```
rng('default') % For reproducibility
x = slicesample(1,2000,'pdf',@exppdf,'thin',5,'burnin',1000);
histogram(x)
```

The histogram shows that the resulting random sample when using `exppdf` has an exponential distribution.

Probability Distribution Apps and User Interfaces

Apps and user interfaces provide an interactive approach to working with parametric and nonparametric probability distributions.

Distribution Fitter App

The **Distribution Fitter** app allows you to interactively fit a probability distribution to your data. You can display different types of plots, compute confidence bounds, and evaluate the fit of the data. You can also exclude data from the fit. You can save the data, and export the fit to your workspace as a probability distribution object to perform further analysis.

Load the Distribution Fitter app from the Apps tab, or by entering `distributionFitter` in the command window. For more information, see Model Data Using the Distribution Fitter App.

Probability Distribution Function Tool

The **Probability Distribution Function** user interface visually explores probability distributions. You can load the Probability Distribution Function user interface by entering `disttool` in the command window.

Random Number Generation Tool

The Random Number Generation user interface generates random data from a specified distribution and exports the results to your workspace. You can use this tool to explore the effects of changing parameters and sample size on the distributions.

The Random Number Generation user interface allows you to set parameter values for the distribution and change their lower and upper bounds; draw another sample from the same distribution, using the same size and parameters; and export the current random sample to your workspace for use in further analysis. A dialog box enables you to provide a name for the sample.