

České vysoké učení technické v Praze

Fakulta elektrotechnická

**Katedra kybernetiky
Katedra počítačů**



Vytěžování dat – cvičení I

Organizace cvičení, úvod do Matlabu

Oleg Kovářík, Petr Pošík, Pavel Kordík

Program cvičení

- Administrativa
 - Harmonogram cvičení předmětu 336VD
 - Požadavky, podmínky zápočtu
 - Semestrální práce, report, bonusová prezentace
 - Stránky cvičení, odevzdávání úloh, materiály
- Úvod do Matlabu
 - Proč proboha Matlab?
 - Práce s Matlabem, licence, toolboxy
 - Základní příkazy, matice, funkce, grafy

Harmonogram cvičení

1. Úvod do Matlabu, import dat
2. Předzpracování a vizualizace dat
3. Klasifikace na základě pravděpodobnosti, hodnocení klasifikátoru
4. Lineární separace, metoda nejbližších sousedů
5. Učící křivka, přeučení, nedoučení, rozhodovací stromy
6. Zadání 1. semestrální úlohy
7. Konzultace a práce na úloze
8. Shluková analýza, K-means, hierarchické shlukování
9. Samoorganizující se mapy (SOM)
10. Zadání 2. semestrální úlohy
11. Konzultace a práce na úloze, TEST
12. Konzultace a práce na úloze
13. Prezentace nejlepších semestrálních úloh
14. Zápočet

Požadavky na zápočet

- Odevzdání semestrální úlohy 1 (report max. 20b)
- Odevzdání semestrální úlohy 2 (report max. 20b)
- TEST na cvičení (max. 20b).
- Účast, aktivita na cvičení, prezentace (max. 20b).
- **Minimum 40b ze semestru**

Hodnocení z předmětu

- **Zkouška (min. 10b, max. 20b):**
 - Test na zkoušce (min. 10b, max. 20b)
- **Celkové hodnocení:**

Body CV+ZK	Známka
100 - 90	výborně (A)
89 - 80	velmi dobře (B)
79 - 70	dobře (C)
69 - 60	uspokojivě (D)
59 - 50	dostatečně (E)
49 a méně	nevyhověl

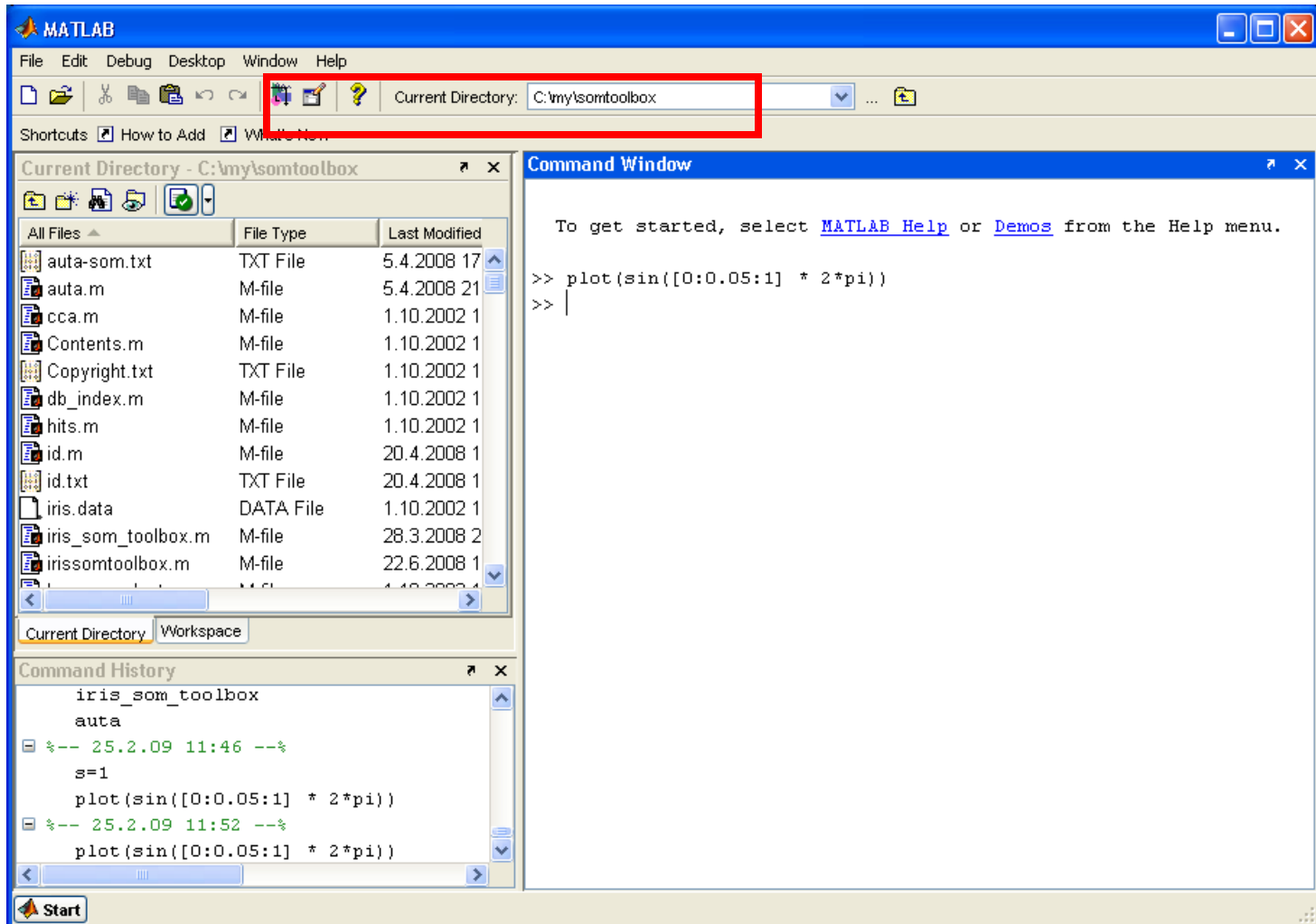
Podpora předmětu

- <http://cw.felk.cvut.cz/>
- <http://cw.felk.cvut.cz/doku.php/courses/y336vd/start>

MATLAB, proč?

- MATLAB je software pro vědeckotechnické výpočty
 - sdružuje prostředky pro výpočty, vizualizaci, programování a odladování
 - určen pro vývoj algoritmů, méně vhodný pro finální aplikace
- světový standard pro výuku a vývoj v technických a inženýrských oborech na universitách (>3500) i v průmyslu (letectví, biotechnologie, komunikace, elektronika, finančnictví, strojírenství, robotika, . . .)
- Pro programátora je jednodušší analyzovat kód, když se chce podívat „pod pokličku“.
- Kompromis mezi klikacím software a programováním metod v Javě či C++.
- ... <https://download.cvut.cz>

Matlab, pracovní adresář



The screenshot displays the MATLAB environment. The 'Current Directory' window shows a list of files in the directory 'C:\my\somtoolbox'. The 'Command Window' shows the following commands and output:

```
To get started, select MATLAB Help or Demos from the Help menu.  
>> plot(sin([0:0.05:1] * 2*pi))  
>> |
```

The 'Command History' window shows the following commands:

```
iris_som_toolbox  
auta  
%-- 25.2.09 11:46 --%  
s=1  
plot(sin([0:0.05:1] * 2*pi))  
%-- 25.2.09 11:52 --%  
plot(sin([0:0.05:1] * 2*pi))
```


Skaláry

- skalár = matice 1x1

```
>> s = 1      >> s = 1;  
s =  
    1
```

```
>> s          >> disp(s);  
s =          1  
    1
```

- operace

```
>> t = s + 3  
t =  
    4
```

```
>> t ^ 2  
ans =  
    16
```

- Příklad: zadejte rozměry obdélníku, vypočítejte obsah

Vektory

- řádkové, sloupcové = matice $D \times 1$ nebo $1 \times D$

```
>> vrow = [1 2 3]
```

```
vrow =
```

```
    1    2    3
```

```
>> vrow(2)
```

```
ans =
```

```
    2
```

```
>> length(vrow)
```

```
ans =
```

```
    3
```

```
>> vcol = [1; 2; 3]
```

```
nebo
```

```
>> vcol = [1 2 3]'
```

```
vcol =
```

```
    1
```

```
    2
```

```
    3
```

Vektory – operátor :

- vytvoření vektoru s ekvidistantními prvky

```
>> v1 = 1:10
```

```
v1 =
```

```
1 2 3 4 5 6 7 8 9 10
```

```
>> v2 = 10:-2:1
```

```
v2 =
```

```
10 8 6 4 2
```

Př.: vytvořte sloupcový vektor lichých čísel mezi
 12^2 a 13^2

Matice

- vytvoření vektoru s ekvidistantními prvky

```
>> M = [11 12 13; 21 22 23; 31 32 33]
```

```
>> M = [M; [41 42 43]]
```

```
>> M = [M [14 24 34 44]']
```

```
>> [M M]
```

```
>> [M; M]
```

```
>> [M M; M M]
```

```
>> size([M M])
```

Př.: vytvořte dvouřádkovou matici s prvky 0 až 9 v prvním řádku a 10 až 19 v druhém řádku

Matice – přístup k prvkům

■ indexování pomocí kulatých závorek

```
>> M = [0:9; 10:19, 20:29]
```

```
>> M(2,3)
```

```
>> M(10)
```

■ výběr podmatice

```
>> M(1:2,[1 2 4])          >> M(2,:)          % celý radek
```

```
>> M(:) % s1. vektor      >> M(:,3)          % celý sloupec
```

■ přiřazování

```
>> M(2,3) = -23          >> M(1,:) = [1 2 3 4] %radek
```

```
>> M(1:2,[2 4]) = [-12 -14; -22 -24] % podmatice
```

```
>> M(end,:) = []        % vymazani posledního radku
```

Maticy – spec., operace

■ speciální typy

`zeros(2,3)` `ones(2,3)` `eye(2,3)` `rand(2,3)`

`randn(2,3)` % normální rozdělení

■ maticové operace

`Ma = [1 2; 3 4]; Mb = [11 12; 13 14];`

`Ma + Mb` `Ma * Mb` `Ma .* Mb`

`inv(Ma)` `inv(Ma)*Ma` `Ma ^ 2`

`Ma .^ 2` `Ma / 2` `Mb ./ Ma`

`eig(Ma)`

Lineární algebra

- řešení soustavy $Ax = b$

```
>> A = [1 2 3; -1 0 2; 1 3 1]; b = [1 0 0]';
```

```
>> x = inv(A)*b % pomoci inverzni matice
```

```
>> x = A\b % lepe, pomoci Gaussovy eliminace
```

```
>> A*x-b % kontrola spravnosti
```

Logické operace

```
>> a = 1:6; b = a>3
>> b = (a>3) | (a==1)
>> b = (a>3) & (a~=5)
>> ind = find(b)      % indexy nenulových prvku
```


Grafy

■ plot

```
>> x = [0:0.05:1] * 2*pi;
```

```
>> ysin = sin(x);
```

```
>> plot(x,ysin);
```

```
>> hold on; % Podrzieme obrazek, aby se graf pridal
```

```
>> ycos = cos(x);
```

```
>> plot(x,ycos,'mx:');
```

```
>> plot(x,ysin,'ro--'); % nastaveni barvy (r), bodu  
% (o) a vzoru cary (--)
```

Grafy 2

```
>> close all; % Zavreme vsechny otevrene grafy
>> x = 1:10;
>> y = exp(-x);
>> plot(x,y);

>> semilogy(x,y);

>> subplot(2,1,1);
>> plot(x,y);
>> subplot(2,1,2);
>> semilogy(x,y);
```

Grafy 3: Když byl ten Valentýn...

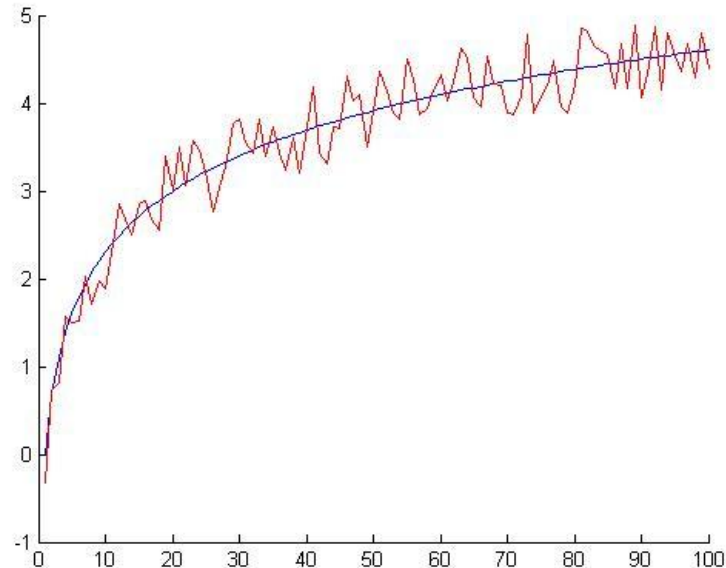
```
>> t = 0:2*pi/120:2*pi
>> r = (sin(t).*sqrt(abs(cos(t))))./(sin(t)+7/5)-2*sin(t)+2
>> x = r.*cos(t)
>> y = r.*sin(t)
>> plot(x,y,'r', 'Linewidth', 3); axis equal;
```

Příklad

- vytvořte vektor $v1$, obsahující 100 náhodných čísel od -0.5 do 0.5
- vytvořte matici s následujícími sloupci (i je číslo řádku):
 - i $v1(i)$ $\log(i)$
- vykreslete modře sloupeček $\log(i)$ a červeně $\log(i)+v1(i)$

Řešení

```
>> v1 = rand(100,1) - 0.5  
>> M = [(1:100)' v1 log(1:100)']  
  
>> hold on  
>> plot(M(:,1), M(:,3), 'b')  
>> plot(M(:,1), M(:,3)+M(:,2), 'r')
```



Programování - funkce

- funkce v m-souborech

```
function [soucet,rozdil] = SlozitaFunkce(a,b)
%
% Tohle je help k funkci SlozitaFunkce. Vypisuje se
% zadanim prikazu 'help SlozitaFunkce'.
%
    soucet = a+b;
    rozdil = a-b;
end % Nepovinne
```

Programování - funkce

- vypsání, spuštění, nápověda

```
>> type SlozitaFunkce
```

```
>> [s,r] = SlozitaFunkce(8,3)
```

```
>> help SlozitaFunkce
```

- funkce `return` okamžitě ukončuje funkci v místě svého volání a předává řízení volající funkci.

Programování - větvení

■ if-else

```
if s > 11
    disp('Soucet > 11. ');
elseif r < 5,
    disp('Soucet neni > 11 a rozdil je < 5. ')
else
    disp('Soucet neni > 11 a rozdil neni < 5. ');
end
```


Programování - větvení

■ switch-case

```
switch r
```

```
    case {0, 1, 2, 3}
```

```
        disp('Rozdil je v intervalu 0-3.');
```

```
    case 4
```

```
        disp('Rozdil je 4.');
```

```
    case {5, 6, 7}
```

```
        disp('Rozdil je v intervalu 5-7.')
```

```
    otherwise
```

```
        disp('Rozdil uplne jiny.')
```

```
end
```

Programování - cykly

- často se dají nahradit vektorovými operacemi, které bývají mnohem efektivnější
- cyklus for

```
for i = 1:10  
    fprintf('%d ',i);  
end
```

```
for i = [1 6 10000]  
    fprintf('%d ',i);  
end
```

Programování - cykly

- while

```
i = 1;
```

```
while i < 10
```

```
    fprintf('%d ',i);
```

```
    i = i+1;
```

```
end
```

- break – ukončuje cyklus

```
if i == 5, break, end
```

- continue – přeskakuje zbytek iterace

```
if i == 5, continue, end
```

Další příkazy

■ různé

help, doc - nápověda k funkcím a příkazům

who (whos) - výpis proměnných v pracovním prostoru

which - kde je uložen daný m-soubor

edit - otevře m-file v editoru

■ určitě vyzkoušejte

path, addpath

format, disp, fprintf, sprintf, load, save

inf, nan, any, all, isempty, round, ceil, floor, ops

min, max, sort, sum, mean, std

axis, title, xlabel, ylabel, legend, plot3, contour,

Založení projektu

- menu File->New->Blank M-File
- uložte do aktuálního adresáře
- následující příkazy pište do vytvořeného souboru a vyvolávejte z řádky jeho jménem

Následují 2 alternativní postupy

- První funguje s licencí MATLABu, která umožňuje využití statického toolboxu. Bez problémů by měla fungovat v nejnovější verzi matlabu staženého z download.cvut.cz, i v učebně K308.
- Druhá verze (od slide 35) je univerzálně použitelná, umožňuje však načíst pouze číselná data (soubor musí být předzpracován jinde).

Načtení dat

- stáhnout dataset auto-mpg.data-mod
- objekt dataset ze Statistics Toolboxu

```
auta = dataset('file', 'auto-mpg.data-mod', ...  
              'ReadVarNames', false, ...  
              'ReadObsNames', false, ...  
              'delimiter', '\t', ...  
              'TreatAsEmpty', 'NA');
```

- pojmenování sloupců

```
auta = set(auta, ...  
          'VarNames', {'mpg', 'cyl', 'disp', 'hp', ...  
                      'wt', 'acc', 'year', 'org', 'name'});  
auta(1:5, :)
```

Převody, souhrn

- Převedení proměnné org na nominální (1 - Amerika, 2 - Evropa, 3 - Japonsko)

```
tmporg = nominal(auta.org, {'America', 'Europe', 'Japan'});  
auta = replacedata(auta, tmporg, 'org');  
auta(1:5,:)
```

- Původní hodnoty zůstávají v datové sadě

```
double(auta.org(1:13)')
```

- Počáteční průzkum dat

```
summary(auta)
```


Chybějící hodnoty

- NaN – nejjednodušší je zbavit se řádků

```
promenne = get(auta, 'VarNames');  
for prom = 1:numel(promenne),  
    if isnumeric( auta.(promenne{prom}) ),  
        indChybejicich = isnan( auta.(promenne{prom}) );  
        auta( indChybejicich, : ) = [];  
    end  
end  
summary(auta)
```

Normalizace

- funkce minmax pro jeden vektor

```
function x01 = minmax(x)
    x01 = (x - min(x)) / (max(x) - min(x));
end
```

- aplikace na sloupce 1-7 datové sady

```
auta01 = auta;
x01 = datasetfun( @minmax, auta01(:,1:7), ...
    'UniformOutput', false );
x01 = [x01{:}]; % Převod cell array na matici
auta01 = replacedata( auta01, x01, 1:7);
mins = datasetfun( @min, auta01(:,1:7) )
maxs = datasetfun( @max, auta01(:,1:7) )
```

Načtení dat

- UCI repository - databáze automobilů

```
cars = csvread('auto-mpg.data-mod.csv');
```

- atributy:

- mpg: miles-per-galon, počet mil ujetých na 1 galon paliva
- cyl: cylinders, počet válců
- disp: displacement, zdiv
- hp: horsepower, koňských sil
- wgt: weight, hmotnost
- acc: acceleration, zrychlení
- year: rok výroby
- org: origin, původ (1 - Amerika, 2 - Evropa, 3 - Japonsko)

Normalizace dat

Verze 1: maticově

Obsah souboru norm01v1.m:

```
function data = norm01v1(data)
    pocet = size(data,1);
    minima = min(data);
    maxima = max(data);
    rozsah = maxima - minima;
    data = (data - repmat(minima,pocet,1)) ./ ...
           repmat(rozsah,pocet,1);

end
```

K normalizaci prvních 7 sloupcu v matici cars volat treba jako:

```
>> cars(:,1:7) = norm01v1( cars(:,1:7) );
```

Nevýhody:

- zbytečné vytváření velkých matic pomocí funkce repmat

Normalizace dat

Verze 2: po sloupcích

Obsah souboru norm01v2.m (může obsahovat obě funkce, vně bude přístupná jen ta se stejným názvem jako má m-file):

```
function data = norm01v2(data)
    for i = 1:size(data,2),
        data(:,i) = norm01vec(data(:,i));
    end
end

function vec = norm01vec(vec)
    vec = (vec - min(vec)) / (max(vec) - min(vec));
end
```

Proč normalizace?

- příště ...