ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ, OBOR GEODÉZIE A KARTOGRAFIE
KATEDRA GEOMATIKY

název předmětu

GEOINFORMATIKA

číslo úlohy	název úlohy				
2	Shluková analýza Laserové skenování Analýza hlavních komponent				
školní rok	**studijní skup.**	**číslo zadání**	**Zpracovali:**	**datum**	**klasifikace**
2024	C-101	-	Josef Jehlička Kateřina Chromá Štěpán Šedivý	10.1. 2024	

TECHNICKÁ ZPRÁVA

SHLUKOVÁ ANALÝZA

Zadání: Úkolem bylo náhodně vygenerovat tři množiny bodů (klastry). Na těchto bodech bylo nutné implementovat vlastní formu algoritmu shlukování k-means a tu porovnat s vestavěnou funkcí k-means v softwaru Matlab.

Postup práce:

1. Generování náhodných dat:
    - Inicializace tří matic A, B a C obsahující náhodné body v 2D prostoru.
    - Vytvoření prvního grafu, který zobrazuje tyto body různými barvami.

```
A = randn(10, 2);
B = randn(15, 2) * 1.2 + [11, 18];
C = randn(5, 2) * 1.7 + [19, 7];

figure(1)
plot(A(:, 1), A(:, 2), "ro", B(:, 1), B(:, 2), "bx", C(:, 1), C(:, 2), "gs")
title('Initial Data Points');
```

2. Kombinace dat do jediné matice:
    - Vytvoření matice X spojující body z matic A, B a C.
    - Vytvoření druhého grafu pro zobrazení kombinovaných dat.

```
X = [A; B; C];

figure(2)
plot(X(:, 1), X(:, 2), "bx")
title('Combined Data Points');
```

3. Výpočet a zobrazení průměru a extrémů:
    - Výpočet minimálního, maximálního a průměrného bodu.
    - Vytvoření třetího grafu pro zobrazení dat a průměru s extrémy.

```
S1 = min(X);
S2 = max(X);
S3 = mean(X);
S = [S1; S2; S3];

figure(3)
plot(X(:, 1), X(:, 2), "bx", S(:, 1), S(:, 2), "gx");
title('Mean and Extremes');
```

4. Přiřazení bodů do nejbližšího shluku:
    - Vytvoření vektoru *nearest_point* pro přiřazení každého bodu k nejbližšímu shluku.
    - Použití smyčky for k výpočtu vzdálenosti od každého bodu k průměrům a přiřazení do shluků.

```
for i = 1:size(X, 1)
    dist_S1 = sqrt((X(i, 1) - S1(1))^2 + (X(i, 2) - S1(2))^2);
    dist_S2 = sqrt((X(i, 1) - S2(1))^2 + (X(i, 2) - S2(2))^2);
    dist_S3 = sqrt((X(i, 1) - S3(1))^2 + (X(i, 2) - S3(2))^2);

    if dist_S1 <= min([dist_S1, dist_S2, dist_S3])
        nearest_point(i) = 1;
    elseif dist_S2 <= min([dist_S1, dist_S2, dist_S3])
        nearest_point(i) = 2;
    else
        nearest_point(i) = 3;
    end
end
```

5. Vytvoření matic pro zobrazení shluků:
    - Vytvoření matic pro jednotlivé shluky (modrá, červená, zelená).
    - Vytvoření čtvrtého grafu pro zobrazení počátečních shluků.

```
plot_matrix = [X, nearest_point];

modra = plot_matrix(plot_matrix(:, 3) == 1, :);
cervena = plot_matrix(plot_matrix(:, 3) == 2, :);
zelena = plot_matrix(plot_matrix(:, 3) == 3, :);

figure(4);
scatter(modra(:, 1), modra(:, 2), 'b');
hold on
scatter(cervena(:, 1), cervena(:, 2), 'r');
scatter(zelena(:, 1), zelena(:, 2), 'g');
title('Initial Clustered Data');
```

6. Implementace K-Means algoritmu:
    - Nastavení maximálního počtu iterací a inicializace smyčky pro opakování algoritmu.
    - V každé iteraci:
        i. Přiřazení bodů do nejbližších shluků.
        ii. Aktualizace průměrů shluků.

```
axIter = 100;

for iteration = 1:maxIter
    % Assign points to clusters
    nearest_point = zeros(size(X, 1), 1);

    for i = 1:size(X, 1)
        dist_S1 = sqrt((X(i, 1) - S1(1))^2 + (X(i, 2) - S1(2))^2);
        dist_S2 = sqrt((X(i, 1) - S2(1))^2 + (X(i, 2) - S2(2))^2);
        dist_S3 = sqrt((X(i, 1) - S3(1))^2 + (X(i, 2) - S3(2))^2);

        if dist_S1 <= min([dist_S1, dist_S2, dist_S3])
            nearest_point(i) = 1;
        elseif dist_S2 <= min([dist_S1, dist_S2, dist_S3])
            nearest_point(i) = 2;
        else
            nearest_point(i) = 3;
        end
    end

    % Update cluster means
    S1 = mean(X(nearest_point == 1, :));
    S2 = mean(X(nearest_point == 2, :));
    S3 = mean(X(nearest_point == 3, :));
end
```

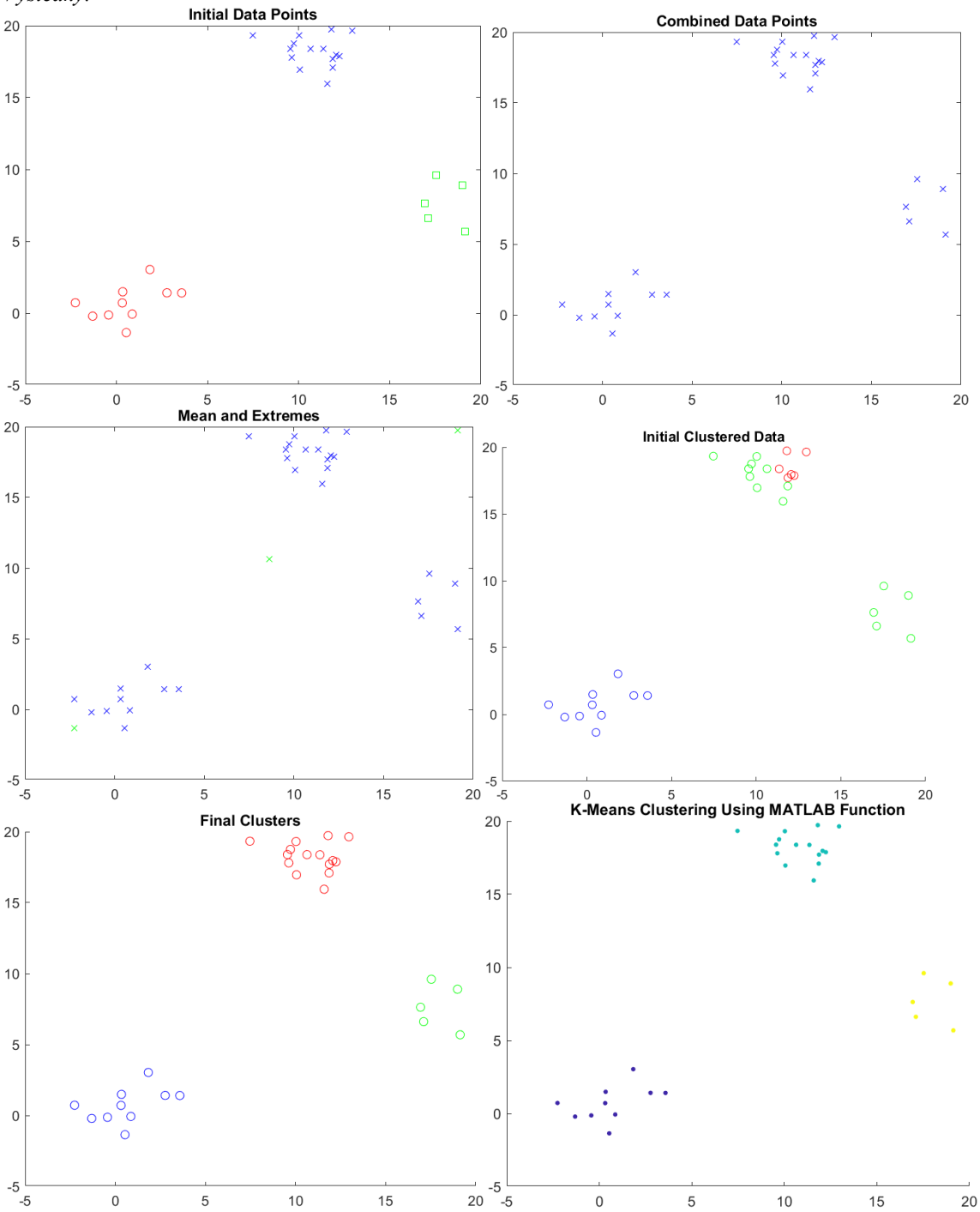
7. Zobrazení konečných shluků:
    - Vytvoření matic pro jednotlivé konečné shluky.
    - Vytvoření pátého grafu pro zobrazení konečných shluků.

```
modra = X(nearest_point == 1, :);
cervena = X(nearest_point == 2, :);
zelena = X(nearest_point == 3, :);

figure;
scatter(modra(:, 1), modra(:, 2), 'b');
hold on;
scatter(cervena(:, 1), cervena(:, 2), 'r');
scatter(zelena(:, 1), zelena(:, 2), 'g');
title('Final Clusters');
```

8. Použití k-means funkce v MATLABu:
    - Použití vestavěné funkce kmeans pro srovnání s vlastní implementací K-Means.
    - Zobrazení shluků pomocí šestého grafu.

```
[idx, C] = kmeans(X, 3); % Assuming 3 clusters

figure;
scatter(X(:, 1), X(:, 2), 10, idx, 'filled');
hold on;
title('K-Means Clustering Using MATLAB Function');
```

Výsledky:



Závěr: Implementace K-Means algoritmu v jazyku MATLAB byla úspěšně provedena. Výsledky vlastní implementace k-means se shodují s výsledky vestavěné funkce softwaru MATLAB. Lze konstatovat, že algoritmus K-means je užitečný při shlukování dat do homogenních skupin, což umožňuje identifikaci vzorů a struktury v datech.

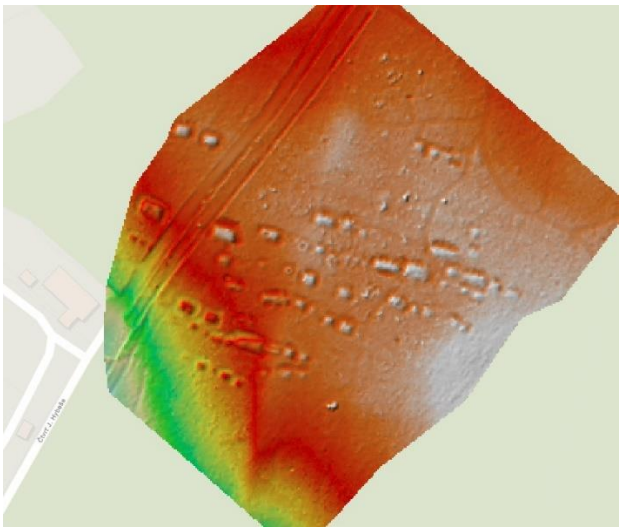
Přílohy: Soubor výpočetního softwaru MATLAB (k_means.m)

LASEROVÉ SKENOVÁNÍ:

Zadání: Úkolem bylo provést filtraci bodového mračna pořízeného při leteckém laserovém skenování z části archeologické lokality Klobásná u Veselí nad Lužnicí pomocí softwarů CloudCompare a LasTools. Vyfiltrované bodové mračno bylo nutno převést na rastr.

Postup zpracování: K tomuto úkolu byly postupně využívány softwary LasTools a CloudCompare. Mračno bodů bylo pořízeno kvůli lokalizaci jednotlivých mohyl pohřebiště na území lesa, a z tohoto důvodu bylo nutné, aby mračno bylo filtrováno, dokud nezbyly pouze body terénního reliéfu. V první fázi byly různé nástroje obsažené v softwaru LasTools využívány k tomu, aby bylo mračno filtrováno. V druhé fázi byl na základě přefiltrovaného mračna pomocí softwaru CloudCompare vytvořen výškový rastr zobrazovaného území. Po přiřazení vhodné barevné škály jsou v tomto výškovém rastru dobře znatelné jednotlivé mohyly.

Výsledek:



Ukázka georeferencovaného rastru v sw. ArcGIS Pro

Závěr:

Díky kombinovanému použití softwarů LasTools a CloudCompare bylo bodové mračno filtrováno a následně převedeno na výškový rastr, který jasně zobrazuje terénní reliéf a umožňuje detailní identifikaci jednotlivých mohyl na pohřebišti v lese.

ANALÝZA HLAVNÍCH KOMPONENT:

Zadání: Úkolem bylo vytvořit dvě dvourozměrné datové sady o dvaceti pozorování, na nichž bylo potřeba ukázat význam transformace hlavních komponent. V prvním případě bude po transformaci první hlavní komponenta obsahovat alespoň 70% informace datového souboru. Ve druhém případě bude vliv transformace minimální. Obsah informace v původních a transformovaných osách se nebude lišit o více než o 10%. V obou případech bylo nutné spočítat vlastní čísla a vlastní vektory kovarianční matice.

Postup zpracování:

1) Generování dat:
    - Vytvoření dvourozměrných datových sad L1 a L2.
    - Transformace dat do vektorů V1 a V2.
    - Normalizace vektorů odečtením jejich průměrů.

```
m = 10;
n = 2;
L1 = randn(m, n);
L2 = randn(m, n) * 3;

V1 = reshape(L1, [1, m * n]);
V2 = reshape(L2, [1, m * n]);
V1 = double(V1);
V2 = double(V2);

P1(:, 1) = V1 - mean(V1);
P1(:, 2) = V2 - mean(V2);
```

2) Analýza hlavních komponent:
   - Výpočet kovarianční matice S1.
   - Výpočet korelační matice R1.
   - Výpočet vlastních čísel a vektorů kovarianční matice.
   - Transformace dat do nových os hlavních komponent (P_rot1).

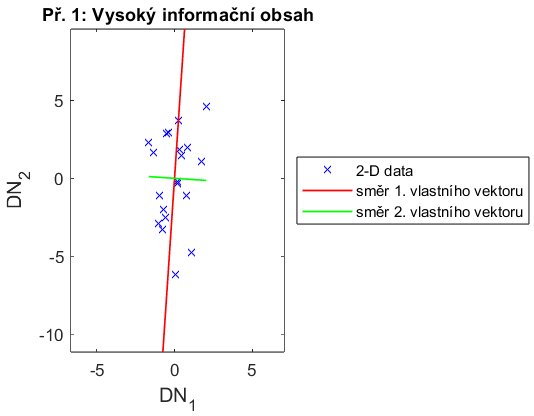
```
disp('kovarianční matice (př 1.):');
S1 = cov(P1)
disp('korelační matice (př 1.):');
R1 = corrcoef(P1)
[V1, D1] = eig(S1);

[d1, ind1] = sort(abs(diag(D1)), 'descend');
disp('vlastní čísla kovarianční matice (př 1.):');
Ds1 = D1(ind1, ind1)
disp('vlastní vektory kovarianční matice (př 1.):');
Vs1 = V1(:, ind1);
pc1 = Vs1;

P_rot1 = P1 * pc1;
```

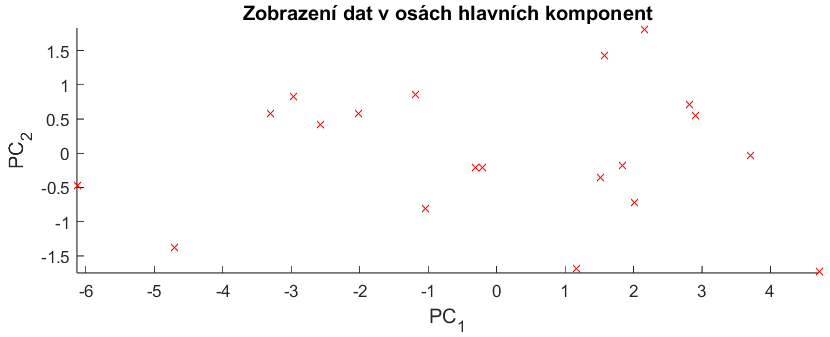
3) Vizualizace výsledků:
   - Vytvoření grafů pro původní data a transformovaná data.
   - Zobrazení směrů hlavních vektorů na grafu.
   - Výpis informačního obsahu komponent
   - Výpis informačního obsahu první a druhé komponenty.

4) Generace dat s vyšším rozptylem a opakování procesu pro druhý příklad

Výsledky:

- Př.1: Vysoký informační obsah





kovarianční matice:

0.9677	0.5131
0.5131	8.5604

vlastní čísla kovarianční matice:

8.5949	0
0	0.9331

korelační matice:

1.0000	0.1783
0.1783	1.0000

vlastní vektory kovarianční matice:

0.0671	-0.9977
0.9977	0.0671

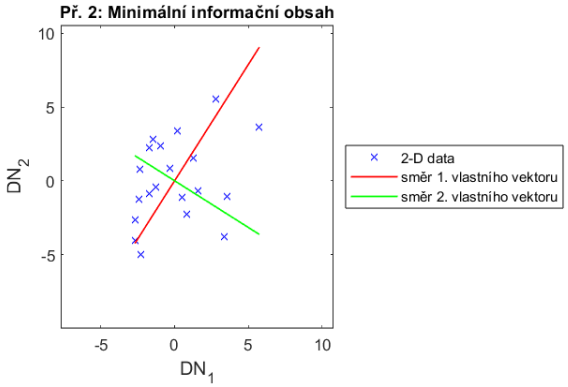
informační obsah první komponenty:

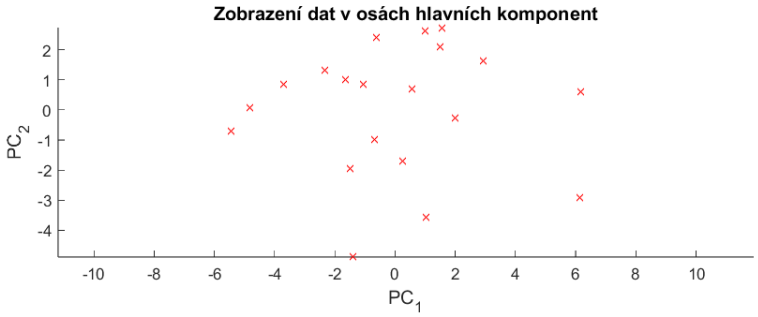
0.9021

informační obsah druhé komponenty:

0.0979

- Př. 2: Minimální informační obsah






```
kovarianční matice:
    5.8842    2.1594
    2.1594    7.9249

korelační matice:
    1.0000    0.3162
    0.3162    1.0000

vlastní čísla kovarianční matice:
    9.2929         0
         0    4.5162

vlastní vektory kovarianční matice:
    0.5352   -0.8448
    0.8448    0.5352

informační obsah první komponenty
    0.6730
informační obsah druhé komponenty:
    0.3270
```

Závěr: V prvním případě, kde byl vysoký informační obsah první hlavní komponenty, transformace dat odhalila, že více než 90% informace je obsaženo v první komponentě. Vlastní vektory ukázaly, že první hlavní komponenta je dominující směr v datech. Naopak, ve druhém případě, kde byl minimální informační obsah, transformace potvrdila, že obě hlavní komponenty přispívají podstatně k informaci, ačkoliv s různým podílem. Celkově lze konstatovat, že transformace hlavních komponent efektivně zaznamenala rozložení informace v obou případech.

Přílohy: Soubor výpočetního softwaru MATLAB (pca.m)

Dne 10.1.2024 v Praze Josef Jehlička, Kateřina Chromá, Štěpán Šedivý