PCA: Motion Capture

(T.Werner, V.Franc 2014+, V.Voráček)

Tato domácí úloha má tři podúlohy. Zadání se může zdát dlouhé, ale každou požadovanou funkci lze napsat na pár řádků. Úlohu vypracujte v Matlabu nabo Pythonu. Můžete využít templaty pro Matlab [/wiki/_media/courses/b0b33opt/cviceni/hw/pca1/matlab_template_mocap.zip] a Python [/wiki/_media/courses/b0b33opt/cviceni/hw/pca1/python_template_mocap.zip].

1. Proložení bodů podprostorem

Jsou dány body $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^m$ a přirozené číslo $k \leq m$. Najděte body $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$ takové, aby ležely v (lineárním) podprostoru dimenze k prostoru $\mathbb{R}^m$ a byly co nejblíže bodům $\mathbf{a}_1, \ldots, \mathbf{a}_n$ ve smyslu nejmenších čtverců, tj. minimalizovaly výraz

$$\sum_{i=1}^{n} \|\mathbf{a}_i - \mathbf{b}_i\|^2. \tag{1}$$

Zdůrazněme, že zmíněný podprostor předem neznáme, máme ho najít zároveň s body $\mathbf{b}_1, \ldots, \mathbf{b}_n$. Tento podprostor budeme reprezentovat jeho ortonormální bází $\mathbf{u}_1, \ldots, \mathbf{u}_k \in \mathbb{R}^m$. Dále máme najít souřadnice $c_{11}, \ldots, c_{kn}$ nalezených bodů $\mathbf{b}_1, \ldots, \mathbf{b}_n$ v této bázi, tedy

$$\mathbf{b}_j = \sum_{i=1}^{k} c_{ij}\mathbf{u}_i = \mathbf{U}\mathbf{c}_j \qquad \forall j = 1, \ldots, n \tag{2}$$

kde $\mathbf{U} \in \mathbb{R}^{m\times k}$ je matice se sloupečky $\mathbf{u}_1, \ldots, \mathbf{u}_k$ (splňující $\mathbf{U}^T\mathbf{U} = \mathbf{I}$) a $\mathbf{c}_j \in \mathbb{R}^k$ je vektor s prvky $c_{1j}, \ldots, c_{kj}$.

Někdy řešíme pozměněnou úlohu: k daným bodům $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^m$ hledáme body $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$, které leží v *afinním* podprostoru dimenze k a minimalizují chybu (1) Pak místo (2) máme

$$\mathbf{b}_j = \mathbf{b}_0 + \mathbf{U}\mathbf{c}_j \qquad \forall j = 1, \ldots, n, \tag{3}$$

kde $\mathbf{b}_0 \in \mathbb{R}^m$ je (neznámé) posunutí afinního podprostoru vůči počátku.

Nahrazení sekvence $\mathbf{a}_1, \ldots, \mathbf{a}_n$ sekvencí $\mathbf{c}_1, \ldots, \mathbf{c}_n$ lze vnímat jako kompresi dat: druhá sekvence obsahuje typicky daleko méně čísel než první (velikost báze $\mathbf{U}$ je zanedbatelná).

Je výhodné uspořádat vektory $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^m$, $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{R}^m$, $\mathbf{c}_1, \ldots, \mathbf{c}_n \in \mathbb{R}^k$ do sloupců matic $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{B} \in \mathbb{R}^{m \times n}$, $\mathbf{C} \in \mathbb{R}^{k \times n}$. Pak účelovou funkci (1) můžeme napsat jako $\|\mathbf{A} - \mathbf{B}\|^2$ (kde $\|\cdot\|$ je zde Frobeniova norma) a rovnici (2) příp. (3) jako $\mathbf{B} = \mathbf{UC}$ příp. $\mathbf{B} = \mathbf{b}_0\mathbf{1}^T + \mathbf{UC}$.

Úkoly:

1. Implementujte matlabskou funkci `[U,C]=fitlin(A,k)` , jejímž vstupem je matice $\mathbf{A}$ a číslo k a výstupem jsou matice $\mathbf{U}$ a $\mathbf{C}$, které minimalizují (1) za podmínky (2).
   Výstup úkolu: soubor `fitlin.m` .
2. Implementujte funkci `[U,C,b0]=fitaff(A,k)` , jejímž vstupem je matice $\mathbf{A}$ a číslo k a výstupem jsou matice $\mathbf{U}, \mathbf{C}$ a vektor $\mathbf{b}_0$, které minimalizují (1) za podmínky (3).
   Výstup úkolu: soubor `fitaff.m` .

Poznámky:

- Předpokládejte, že nejen $k \leq m$ ale i $k \leq n$.
- Neměli byste nikde vytvořit matici rozměru $n \times n$, protože počet bodů n může být veliký.
- Implementované funkce nemají nic vypisovat ani vykreslovat, mají jen vrátit požadovaný výstup.
- Smíte používat jen základní funkce Matlabu (tedy žádné toolboxy), viz stránka cvičení.
- Funkci `fitlin` lze napsat na 4 řádky, funkci `fitaff` na 3 řádky.

2. Proložení bodů přímkou

Nyní použijete výsledek výše na prokládání množiny bodů v rovině přímkou ($m = 2$ a $k = 1$), která nemusí procházet počátkem. Představte si např., že někdo body naklikal myší v grafickém rozhraní (to můžete udělat v Matlabu sami příkazem `ginput`) a vaším úkolem je proložit jimi nejlepší přímku.

Úkoly:

1. Implementujte funkci `drawfitline(A)` , která má na vstupu matici $\mathbf{A} \in \mathbb{R}^{2 \times n}$ se zadanými body a nakreslí optimální přímku zeleně, body $\mathbf{a}_1, \ldots, \mathbf{a}_n$ jako červené křížky, a n červených úseček kde $i$tá úsečka spojuje bod $\mathbf{a}_i$ s bodem $\mathbf{b}_i$. Funkci si můžete vyzkoušet na matici $\mathbf{A}$ v souboru line.mat [/wiki/_media/courses/b0b33opt/cviceni/hw/pca1/line.mat] (nahrajete ho příkazem `load line`).
   Výstup úkolu: soubor `drawfitline.m` .
   Poznámky:
   - Uvědomte si, že místo přímky musíte vlastně nakreslit úsečku (protože přímka je nekonečná a na obrazovku se nevejde) a tedy musíte nějak rozumně zvolit koncové body této úsečky.

- Pro vykreslení použijte příkazy `plot` , `hold on` , `hold off` . Po vykreslení zavolejte příkaz `axis equal` , aby měřítko obou os bylo stejné.
- Uvnitř funkce neotvírejte ani nezavírejte matlabský obrázek (tj. nevolejte funkce `figure` ani `close`).
- Funkci lze napsat do 15 řádků.

3. Komprese a analýza sekvence z motion capture

Při tvorbě počítačových her nebo filmů se používá technologie *motion capture*. Na živého herce se připevní terčíky odrážející infračervené světlo. Terčíky se připevňují na významné body na těle, jako klouby apod. Speciální soustava kamer snímají polohy terčíků a z těch se počítá poloha každého terčíku v třírozměrném prostoru pro každý snímek. Polohy terčíků v prostoru se pak použijí např. pro animaci postav syntetizovaných počítačovou grafikou. Viz např. wikipedie [http://en.wikipedia.org/wiki/Motion_capture].

Pro získání plynulého pohybu je třeba snímat s vysokou frekvencí. Například data použitá v naší úloze byla snímána s vzorkovací frekvencí 120 Hz. Ve výsledku je pak třeba pracovat s velkými objemy dat. Naším úkolem bude snížit objem dat tak, abychom sejmuté body poškodili co nejméně.

Prostorová poloha jednoho terčíku v jednom snímku je dána trojicí souřadnic. V našem případě máme $\ell = 41$ terčíků. Poloha všech terčíků v jednom snímku je dána vektorem $\mathbf{a} \in \mathbb{R}^m$ kde $m = 3\ell$. Ten si lze představit jako bod v m-rozměrném prostoru. Celkově máme n snímků, tedy vektory $\mathbf{a}_1, \ldots, \mathbf{a}_n \in \mathbb{R}^m$.

Hledáme body, které co nejlépe aproximují původní body a zároveň se dají reprezentovat menším objemem dat. Přesněji, hledáme body $\mathbf{b}_1, \ldots, \mathbf{b}_n$, které leží v afinním podprostoru dané dimenze $k < m$ a minimalizují chybu (1).

Úkoly:

1. Stáhněte si data [/wiki/_media/courses/b0b33opt/cviceni/hw/pca1/data.zip] (tanec Macarena nastudujte zde [https://www.youtube.com/watch?v=MMRVbhbIkjk]). Každý soubor obsahuje jednu matici $\mathbf{A}$, nahrajete ji příkazem `A=load('soubor.txt')'` (pozor na transpozici). Pro vizualizaci sekvencí použijte příkaz `playmotion(conn,A)` (vyžaduje funkci playmotion.m [/wiki/_media/courses/b0b33opt/cviceni/hw/pca1/playmotion.m] a soubor `connected_points.txt` , který nahrajete příkazem `conn=load('connected_points.txt')`). Toto, i ekvivalent v pythonu je implementováno v templatech.
   Výstup úkolu: nic.
2. Aproximujte sekvenci příkazem `[U,C,b0]=fitaff(A,k)` pro různě zvolená $k \in \{1, \ldots, m\}$, aproximovaná sekvence je pak dána vzorcem (3). Obě sekvence zároveň si přehrajte příkazem `playmotion(conn,A,B)` . Pokud máte vše správně, sekvence si budou podobné. Zkoušejte, jak se mění kvalita aproximace pro různá k a různé sekvence. Dumejte,

proč se některé sekvence lépe komprimují (stačí menší k) než jiné.
Výstup úkolu: nic.

3. Nahrazení sekvence $\mathbf{a}_1, \ldots, \mathbf{a}_n$ sekvencí $\mathbf{c}_1, \ldots, \mathbf{c}_k$ může mít i jiné výhody než kompresi: protože druhá sekvence 'žije' v prostoru nižší dimenze, dá se např. snadněji zobrazit (vizualizace dat) či rozpoznat z ní typ pohybu herce (pattern recognition). Zkuste si to: pro $k \in \{2, 3\}$ si zobrazte trajektorii bodu $\mathbf{c}_i$ v rovině jako funkci času i, kde po sobě jdoucí body spojíte úsečkami (použijte příkaz `plot` příp. `plot3`; pro $k = 3$ si na matlabském okně s obrázkem zvolte rotaci a točte si 3-D grafem v prostoru). Všimněte si, jak se trajektorie liší pro různé vstupní sekvence (`walk1, makarena1, …`). Implementujte funkci `plottraj2(C)` se vstupem $\mathbf{C} \in \mathbb{R}^{2 \times n}$, která zobrazí tuto trajektorii pro $k = 2$.
Výstup úkolu: soubor `plottraj2.m`.

4. Chceme spočítat optimální chybu aproximace (1) pro všechny dimenze $k = 1, \ldots, m$ afinního podprostoru. Dostaneme tedy čísla $d_1, \ldots, d_m$, kde d_k je chyba aproximace pro dimenzi k. Implementujte funkci `d=erraff(A)`, která pro sekvenci $\mathbf{A} \in \mathbb{R}^{m \times n}$ spočítá tato čísla a uloží je do vektoru $\mathbf{d} \in \mathbb{R}^m$ (funkce nemá nic vypisovat ani vykreslovat). Ve funkci `erraff` smíte zavolat matlabskou funkci `eig` příp. `svd` **jen jednou**. V BRUTE je jeden test na matici asi 1000×1000, funkce musí být dost rychlá, aby to stihla.
Výstup úkolu: soubor `erraff.m`.

courses/b0b33opt/cviceni/hw/pca1/start.txt · Last modified: 2021/10/28 11:40 by wernetom

Copyright © 2025 CTU in Prague | Operated by IT Center of Faculty of Electrical Engineering | Bug reports and suggestions Helpdesk CTU