Tereprendezés

1. Bevezetés

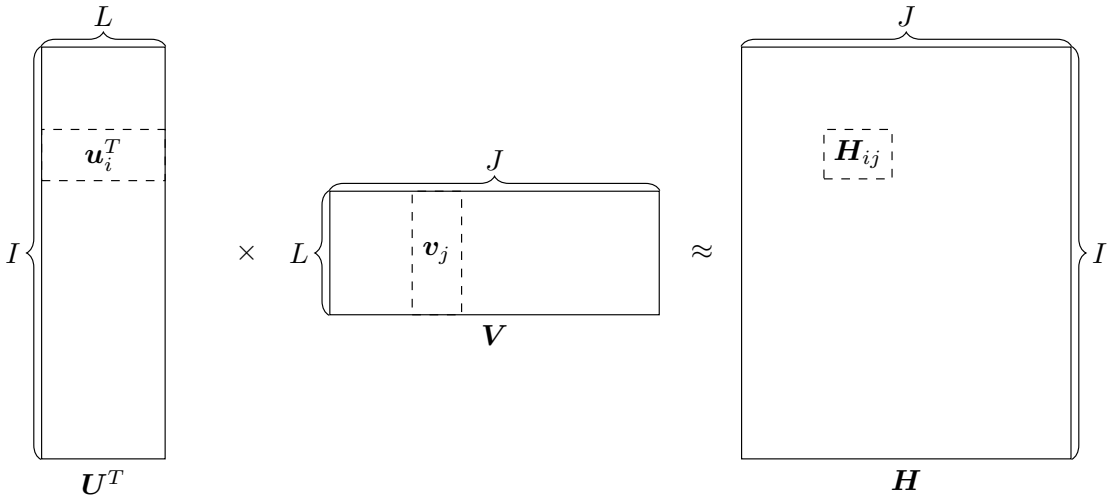
Csapatával stratégiai játékot fejlesztenek. A vezetés úgy dönt, hogy a fejlesztői eszköztárat a modderek számára ingyenesen elérhetővé kell tenni; ennek fontos része a pályagenerátor is. Kollégája el is készül egy terepgenerátorral, amely a jól bevált Diamond-Square algoritmust használja, ám a kapott terepek egytől egyig „tüskések", zajosak, ráadásul sok helyet foglalnak a memóriában. Önre esett a választás, hogy orvosolja a problémát egy intelligens simító-tömörítő algoritmus implementációjával[1].

1.1. Alacsony rangú approximáció

Legyen adott egy $\boldsymbol{H} \in \mathbb{R}^{I \times J}$ magasságtérkép (*heightmap*), amely tehát egy $I \times J$ mátrix, amely elemeiben az egyes mezők magasságát tartalmazza. A feladatunk, hogy találjunk egy $\hat{\boldsymbol{H}}$ magasságtérképet, amely kellőképpen „sima", mégis „hasonlít" az eredetihez. Nyilvánvalóan valamiféle közelítő megoldásra törekszünk, amely egyúttal a tárigény csökkentéséről is gondoskodik. Keressük hát a $\hat{\boldsymbol{H}}$ mátrixot a következő alakban:

$$\boldsymbol{H} \approx \hat{\boldsymbol{H}} = \boldsymbol{U}^T \boldsymbol{V},$$

ahol $\boldsymbol{U} \in \mathbb{R}^{L \times I}$ és $\boldsymbol{V} \in \mathbb{R}^{L \times J}$. Legyen $L \ll I, J$, így a $\boldsymbol{U}$ és $\boldsymbol{V}$ sokkal kisebb helyen tárolható, mint $\hat{\boldsymbol{H}}$ (ún. alacsony rangú approximáció). Az alábbi ábra ezt az elgondolást szemlélteti:

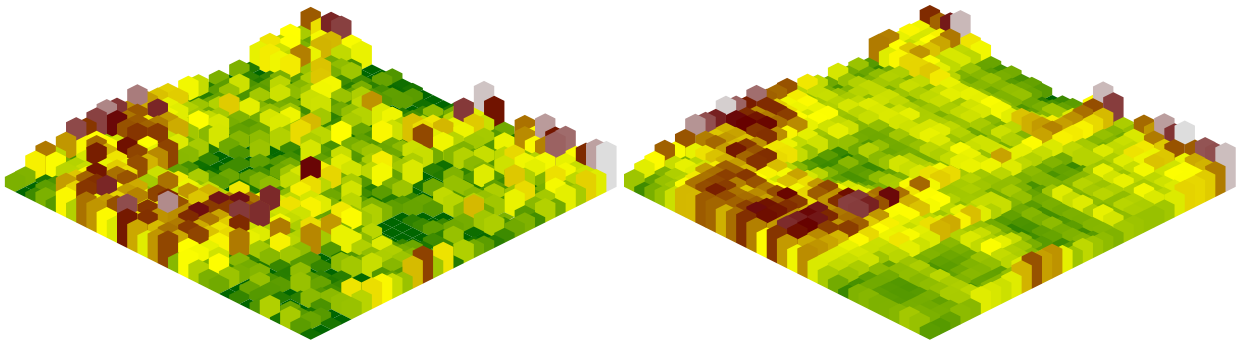


A $\boldsymbol{H}$ mátrix i, j. elemét így is írhatjuk:

$$\boldsymbol{H}_{ij} \approx \boldsymbol{u}_i^T \boldsymbol{v}_j,$$

ahol $\boldsymbol{u}_i$ és $\boldsymbol{v}_j$ a megfelelő mátrix i. illetve j. oszlopvektora; ezek a vektorok L hosszúak.

[1] Megjegyzendő, hogy a problémára ennél sokkal praktikusabb algoritmusok is léteznek. Ez a feladat inkább egy módszer bemutatását célozza, semmint egy gyakorlatban is jól használható algoritmus implementációját.



1. ábra. Bal oldalt az eredeti 30×30 magasságtérkép, jobb oldalt a simított változat $L = 3$ rang esetén. A tárigény $30^2 = 900$ értékről $2 \times 30 \times 3 = 180$ értékre csökkent.

1.2. A bayesi megközelítés

Éljünk azzal naiv feltevéssel, hogy a generált, egyenetlen terep valójában egy normál zajjal terhelt sima terep, ahol az egyes mezők függetlenek egymástól. Ez durva egyszerűsítése a problémának, a számításokat viszont jelentősen megkönnyíti, hiszen ekkor a $\boldsymbol{H}$ mátrix eloszlása a következőképpen írható:

$$p(\boldsymbol{H}|\boldsymbol{U},\boldsymbol{V},\beta) = \prod_{i=1}^{I}\prod_{j=1}^{J} p(\boldsymbol{H}_{ij}|\boldsymbol{u}_i,\boldsymbol{v}_j,\beta) = \prod_{i=1}^{I}\prod_{j=1}^{J} \mathcal{N}(\boldsymbol{H}_{ij}|\boldsymbol{u}_i^T\boldsymbol{v}_j,\beta^{-1}), \tag{1}$$

ahol $\mathcal{N}$ jelöli a normál eloszlást. Az első egyenlőség azt állítja, hogy a $\boldsymbol{H}_{ij}$ értékek függetlenek egymástól; a második pedig azt, hogy $\boldsymbol{H}_{ij}$ értéke az előbb látott $\boldsymbol{u}_i^T\boldsymbol{v}_j$ szám körül mozog β^{-1} szórással. A bayesi megközelítésben szükségünk van a $p(\boldsymbol{U})$ és $p(\boldsymbol{V})$ priorokra is; tudjuk, hogy ha ezeket szintén normál eloszlásnak választjuk, a poszterior is ilyen típusú lesz („konjugált prior”), azaz legyen

$$p(\boldsymbol{U}|\alpha_u) = \prod_{i=1}^{I} p(\boldsymbol{u}_i|\alpha_u) = \prod_{i=1}^{I} \mathcal{N}(\boldsymbol{u}_i|\boldsymbol{0},\alpha_u^{-1}\boldsymbol{I}), \tag{2}$$

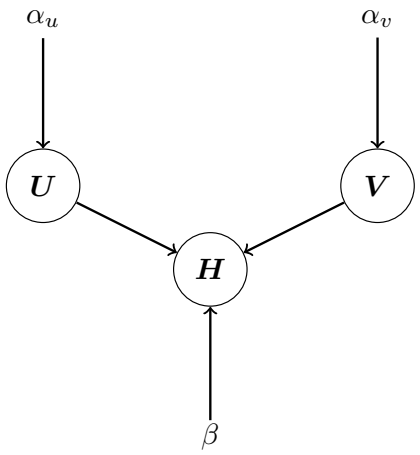
$$p(\boldsymbol{V}|\alpha_v) = \prod_{j=1}^{j} p(\boldsymbol{v}_j|\alpha_v) = \prod_{j=1}^{J} \mathcal{N}(\boldsymbol{v}_j|\boldsymbol{0},\alpha_v^{-1}\boldsymbol{I}). \tag{3}$$

Másként fogalmazva, a függetlenség az $\boldsymbol{U}$, $\boldsymbol{V}$ mátrixok oszlopvektoraira is teljesül, valamint ezek $\boldsymbol{0}$ várható értékű, α^{-1} szórású körszimmetrikus kovarianciamátrixú normál eloszlást követnek. A fenti modell a 2. ábrán látható Bayes-hálóval írható le.

1.3. Következtetés Gibbs-mintavételezéssel

Feladatunk, hogy az ismert $\boldsymbol{H}$ terep alapján megtaláljuk az $\boldsymbol{U}$ és $\boldsymbol{V}$ mátrixokat, ehhez a 2. ábrán látható Bayes-hálóban kell következtetnünk. A feladat valószínűségi természetéből fakadóan az eredmény nem is egy pontbecslés lesz $\boldsymbol{U}$-ra és $\boldsymbol{V}$-re, hanem egy egész eloszlás. Ezen eloszlás konstrukciójára az egyik legegyszerűbb, széles körben alkalmazott algoritmus a Gibbs-mintavételezés, amelynek általános leírása megtalálható a tankönyv 14. fejezetében. Az eljárás során az $\boldsymbol{U}$ és $\boldsymbol{V}$ változókat felváltva mintavételezzük, az aktuálison kívül minden más változót fixálva; a két keresett eloszlás ezen két mintahalmazzal becsülhető. Így az is nyilvánvaló, hogy az algoritmus nem determinisztikus, különböző futtatásoknál más-más eredményeket kaphatunk; megmutatható azonban, hogy az iterációk számának növelésével azonban a megoldás konvergál a „valódi” eloszláshoz. Lépésenként:

1. Megadjuk az α_u, α_v hiperparamétereket. Minél magasabbak, annál nagyobb szórást engedünk meg az $\boldsymbol{U}$ és $\boldsymbol{V}$ mátrixoknál.



2. ábra. A modell gráfos ábrázolása. A csomópontok a változókat, a nyilak feltételes függőségi relációkat jelölnek.

2. Megadjuk a β precizitási paramétert. Minél magasabb, annál jobban szeretnénk közelíteni az értékeket az eredeti $\boldsymbol{H}$ mátrixban, más szóval annál inkább „bízunk" az eredeti értékekben.

3. $\boldsymbol{H}$ értékeit a zajos terep alapján állítjuk be, $\boldsymbol{U}$ és $\boldsymbol{V}$ értékeit pedig a prior alapján, véletlenszerűen.

4. Mintavételezzük (frissítjük) $\boldsymbol{U}$-t a $p(\boldsymbol{U}|\boldsymbol{V}, \boldsymbol{H}, \alpha_u, \alpha_v, \beta)$ eloszlás alapján.

5. Mintavételezzük (frissítjük) $\boldsymbol{V}$-t a $p(\boldsymbol{V}|\boldsymbol{U}, \boldsymbol{H}, \alpha_u, \alpha_v, \beta)$ eloszlás alapján.

6. Ismételjük a 4-6. lépéseket; az első meghatározott számú mintát eldobjuk („burn-in"), a továbbiakat rögzítjük, amíg el nem érjük az előre megszabott maximális iterációszámot. Végül, mivel valójában csak pontbecslésre van szükségünk, a rögzített minták átlagát vesszük.

Megfelelő paraméterezéssel a fenti sémát „elég ideig" iterálva olyan megoldáshoz jutunk, ahol az $\boldsymbol{U}^T\boldsymbol{V}$ szorzat valóban jól becsüli $\boldsymbol{H}$-t. A számításigény csökkentésére több lehetőség is adódik, de ezekre most nem térünk ki.

1.1. Állítás. *Az (1), (2), (3) egyenletekkel definált modell esetén $\boldsymbol{U}$ feltételes eloszlása szintén normál, a következő paraméterekkel:*

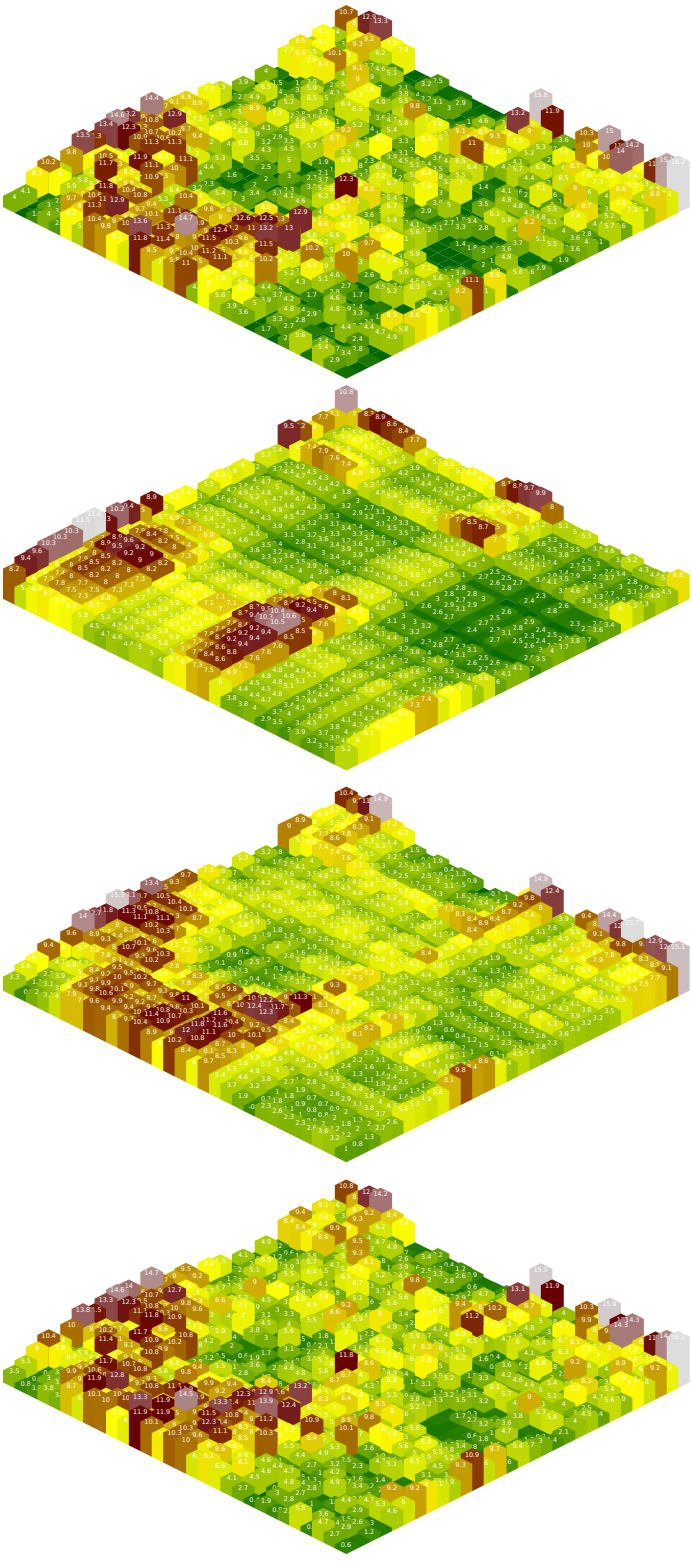
$$p(\boldsymbol{U}|\boldsymbol{V}, \boldsymbol{H}, \alpha_u, \alpha_v, \beta) = \prod_{i=1}^{I} \mathcal{N}(\boldsymbol{u}_i|\boldsymbol{\psi}_i, \boldsymbol{\Lambda}_i^{-1}),$$

$$\boldsymbol{\Lambda}_i = \beta \sum_{j=1}^{J} \boldsymbol{v}_j \boldsymbol{v}_j^T + \alpha_u \boldsymbol{I},$$

$$\boldsymbol{\psi}_i = \boldsymbol{\Lambda}_i^{-1} \beta \sum_{j=1}^{J} \boldsymbol{H}_{ij} \boldsymbol{v}_j.$$

Bizonyítás. Lásd a Függelékben. □

$\boldsymbol{U}$ mintavételezéséhez tehát sorra ki kell számítani a $\boldsymbol{\Lambda}_i$ mátrixokat és $\boldsymbol{\psi}_i$ vektorokat a fenti képletek alapján, ezek ismeretében az egyes $\boldsymbol{u}_i$ oszlopok egymástól függetlenül frissíthetők. $\boldsymbol{V}$ mintavételezése hasonlóképpen alakul, csak $\boldsymbol{u}_i$ és $\boldsymbol{v}_j$ szerepet cserélnek.



3. ábra. Eredeti terep és simított verziói $L = 1$, $L = 3$ ($\alpha_u, \alpha_v = 1$, $\beta = 100$), valamint $L = 20$ ($\alpha_u, \alpha_v = 10$, $\beta = 1000$) paraméterekkel. Látható, hogy utóbbi már igen pontos közelítést ad. Minden esetben 20 burn-in után 100 iterációt használtunk.

2. Feladatok

Implementálja a Gibbs-mintavételezést Java nyelven. A lineáris algebrai számításokhoz és a normál eloszlásból való mintavételezéshez használja az Apache Commons könyvtárat (`http://commons.apache.org`), ezen kívül kódot ne emeljen át külső forrásból. A megoldás tartalmazzon egy `Main` osztályt, ezen belül egy `main()` függvényt. A bemeneti mátrixot a standard inputon várja, a kimenetet a standard outputra írja. A program forráskódját `zip` file-ba tömörítve töltse fel.

2.1. Bemenet

A bemenet az I, J, L számok vesszővel elválaszva, utána a β paraméter, majd $\boldsymbol{H}$ magasságtérkép `csv` formátumban. Példa:

```
5,6,2,100.0
0.20,0.50,2.51,1.12,6.98,4.43
1.08,0.20,1.87,1.31,1.44,3.96
0.67,4.29,4.48,5.68,6.03,3.49
3.93,1.64,1.84,3.32,2.08,4.63
4.89,7.30,3.20,2.38,3.04,0.73
```

(ennél nagyobb, kb. 50×50 mátrixokra számítson).

2.2. Kimenet

A kimenetre az $\boldsymbol{U}$ és $\boldsymbol{V}$ mátrixokat írja `csv` formátumban, egy üres sorral elválaszva. Példa:

```
0.25,2.65,1.61,7.63,2.88
3.79,2.31,5.40,3.88,6.00

1.68,5.91,3.38,3.24,0.57,2.61
2.37,7.10,5.43,0.05,0.01,3.14
```

2.3. Értékelés

Az értékelés során a $\boldsymbol{H}$ és a $\boldsymbol{U}^T\boldsymbol{V}$ mátrixok elemenként vett négyzetes eltérésének átlagát vesszük alapul, több tesztesetre kiszámítva. Tanácsos ezt a megoldás részeként is kiszámítani és figyelembe venni a paraméterek finomhangolásánál. A sikerhez minden tesztnek meg kell felelni; sikertelenség esetén természetesen lehet újra próbálkozni. Tanácsos az algoritmust több, véletlenszerűen generált tesztesettel kipróbálni.

Függelék

Az 1.1. Állítás bizonyítása. A bizonyításhoz először felírjuk az együttes eloszlást. Ehhez kell a Bayes-tétel:

$$p(\boldsymbol{H}, \boldsymbol{U}, \boldsymbol{V} | \beta, \alpha_u, \alpha_v) \propto p(\boldsymbol{H} | \boldsymbol{U}, \boldsymbol{V}, \beta) p(\boldsymbol{U} | \alpha_u) p(\boldsymbol{V} | \alpha_v)$$

Ennek vesszük a logaritmusát, így a szorzatok helyett összegeket írjatunk. Ezenkívül az $\boldsymbol{U}$-t nem tartalmazó tagokat konstansnak vesszük. Így az (1) és (2) egyenletek alapján

$$\ln p(\boldsymbol{H}, \boldsymbol{U}, \boldsymbol{V} | \beta, \alpha_u, \alpha_v) = \sum_{i=1}^{I} \sum_{j=1}^{J} \ln \mathcal{N}(\boldsymbol{H}_{ij} | \boldsymbol{u}_i^T \boldsymbol{v}_j, \beta^{-1}) + \sum_{i=1}^{I} \ln \mathcal{N}(\boldsymbol{u}_i | \boldsymbol{0}, \alpha_u^{-1} \boldsymbol{I}) + \text{const.}$$

Most felhasználjuk a normál eloszlás képletét, így

$$\ln p(\boldsymbol{H}, \boldsymbol{U}, \boldsymbol{V} | \beta, \alpha_u, \alpha_v) = \sum_{i=1}^{I} \sum_{j=1}^{J} -\frac{1}{2} \beta (\boldsymbol{H}_{ij} - \boldsymbol{u}_i^T \boldsymbol{v}_j)^2 - \frac{1}{2} \sum_{i=1}^{I} \alpha_u \boldsymbol{u}_i^T \boldsymbol{I} \boldsymbol{u}_i + \text{const},$$

ahol az $\boldsymbol{U}$-t nem tartalmazó tényezőket ismét beépítettük a konstans tagba. Ebből kifejezhetjük $\boldsymbol{u}_i$-t:

$$\sum_{i=1}^{I} -\frac{1}{2}\boldsymbol{u}_i^T \left(\beta \sum_{j=1}^{J} \boldsymbol{v}_j \boldsymbol{v}_j^T + \alpha_u \boldsymbol{I}\right) \boldsymbol{u}_i + \left(\beta \sum_{j=1}^{J} \boldsymbol{H}_{ij} \boldsymbol{v}_j\right)^T \boldsymbol{u}_i + \text{const},$$

ahonnan látható, hogy ezek $\boldsymbol{u}_i$-ban kvadratikus tagok. Így nincs más dolgunk, mint kiegészíteni teljes négyzetté, és azonnal észrevesszük, hogy a normál eloszlás kitevőjének is pontosan ilyen alakja van:

$$\sum_{i=1}^{I} -\frac{1}{2}(\boldsymbol{u}_i - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\varphi}_i)^T \boldsymbol{\Lambda}_i (\boldsymbol{u}_i - \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\varphi}_i) + \text{const}.$$

A logaritmust visszalakítva tehát normál eloszlást kapunk:

$$\begin{aligned} p(\boldsymbol{U}|\boldsymbol{H}, \boldsymbol{V}, \alpha_u, \alpha_v, \beta) &= \prod_i \mathcal{N}(\boldsymbol{u}_i|\boldsymbol{\psi}_i, \boldsymbol{\Lambda}_i^{-1}), \\ \boldsymbol{\Lambda}_i &= \beta \sum_{j=1}^{J} \boldsymbol{v}_j \boldsymbol{v}_j^T + \alpha_u \boldsymbol{I}, \\ \boldsymbol{\psi}_i &= \boldsymbol{\Lambda}_i^{-1}\boldsymbol{\varphi}_i = \boldsymbol{\Lambda}_i^{-1}\beta \sum_{j=1}^{J} \boldsymbol{H}_{ij}\boldsymbol{v}_j, \end{aligned}$$

ahol a konstans tagok automatikusan bekerültek a normalizációs tagba. □