Arcfelismerés főkomponens analízis segítségével LAPACK könyvtár használatával `C++`-ban

Rudolf Ádám, ELTE TTK, Fizikus MSc

2012. december

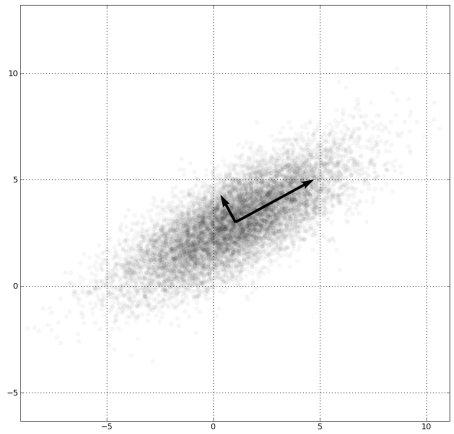
Bevezetés

A Számítógépes modellezés laboratórium órán egyik feladatunk az volt, hogy `C`, vagy `C++` nyelven olyan programot írjunk, ami képek főkomponens analízisét végzi. A számolás elvégzéséhez LAPACK függvénykönyvtárat kellett használni. A képek analízise előtt egy véletlenszerűen generált adatsoron kellett tesztelni a program megbízhatóságát. A teszt után egy adatbázis alapján arcokat ábrázoló képek főkomponens analízisét végeztem, majd az adatbázistól független arcképeket fejtettem ki a főkomponensek szerint.

1. Főkomponens analízis

A főkomponens analízis egy matematikai módszer, ami egy adott bázison ábrázolt N dimenziós adatsort (pontok, vagyis vektorok egy N dimenziós térben) áttranszformál egy másik bázisra oly módon, hogy az adatok lehetőleg minél kevésbé korreláljanak. Ez segít kevesebb változóval jó közelítéssel leírni az adatokat. Ez azt is jelenti, hogy a módszer megtalálja azt a tengelyt, ami mentén az adatok legnagyobb mértékben szórnak, egy rá merőlegeset, ami mentén a második legnagyobb a szórás, stb.

Szemléletes jelentése, hogy ha az adatokra nagyjából illeszteni lehet egy N dimenziós ellipszoidot, akkor a módszer megtalálja az ellipszoid főtengelyeit, és azokra vetíti le az adatokat. Így megtalálhatunk egy adott paramétert, ami leginkább felelős az adataink varianciájáért. Ezt nevezzük első főkomponensnek. Ez az eredeti N darab bázisvektoron kifejthető valahogy. A második főkomponens a második legnagyobb varianciáért felel, stb. Hogyha elvégezzük a transzformációt, de nem mind az N darab, csak annál kevesebb főkom-



1. ábra. A főkomponens analízis szemléltetése egy kétdimenziós Gauss eloszláson. A fekete nyilak az új bázist ábrázolják. A nagyobbik az első, a kisebbik a második főkomponens. Az ábra a Wikipédiáról származik.

ponens szerint fejtjük ki az adatokat, akkor is jó közelítést kaphatunk, mert a legkisebb varianciáért felelő változókat hagytuk el. Ez egyrészt segíthet megmagyarázni az adatok változatosságát, másrészt az adatainkat viszonylag kis veszteséggel levetíti egy kisebb dimenziós altérre, azaz tömörítési eljárásként is használható. (Kevesebb adattal tudjuk jellemezni a pontokat, ami nagy N dimenziószámnál igen hasznos lehet. Az egymáshoz valamennyire hasonló képek, pl. arcok esetén ez hasznos és jól működő eljárás.)

1.1. A főkomponens analízis konkrét végrehajtása

Van egy N dimenziós adatsorunk, ami T darab vektorból áll. A főkomponens analízis (*Principal Component Analysis - PCA*) elvégzéséhez először ki kell vonni a mintából a mintaátlagot. (Komponensenként átlagoljuk az adatsort, majd a kapott értéket mindegyikből kivonjuk. Ezzel gyakorlatilag eltoljuk a minta súlypontját az origóba.) Ezután a PCA már csak az adathalmaz forgatását jelenti, esetleg ezután vetítéseket.

Tekintünk egy $\mathbf{X}$ $N \times T$-s mátrixot, ami a következőképpen áll elő: T darab sora tartalmazza az N dimenziós, már 0-ba tolt adatainkat. (Gyakran nem ezt a mátrixot nevezik $\mathbf{X}$-nek, hanem a transzponáltját, de én ezt a jelölést fogom használni.)

A főkomponens analízis szoros kapcsolatban áll a szinguláris érték dekompozícióval (*Singular Value Decomposition - SVD*). Ez $\mathbf{X}$ felbontása 3 mátrix

szorzatára a következőképpen:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} \tag{1}$$

ahol $\mathbf{U}$ és $\mathbf{V}^{\mathrm{T}}$ tartalmazza $\mathbf{X}$ baloldali és jobboldali szinguláris vektorait, $\mathbf{\Sigma}$ pedig a szinguláris értékeket a főátlójában. Fontos, hogy $\mathbf{U}$ és $\mathbf{V}^{\mathrm{T}}$ általános esetben hermitikusak, azaz valós esetben ortogonálisak. Mi valós esettel fogunk csak foglalkozni. Ezek nem mások, mint az $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ kovarianciamátrix sajátvektorai és sajátértékei, ami érthető is, mert a kovarianciamátrix tartalmazza az információt arról, hogy az adatok milyen irányban korrelálnak egymással jobban, és milyen irányban kevésbé. Ennek a sajátvektorai adják meg a képzeletbeli ellipszoidunk főtengelyeinek irányát.

Maga a PCA lényege a következő művelet:

$$\mathbf{Y} = \mathbf{X}\mathbf{U} \tag{2}$$

ahol $\mathbf{Y}$ mátrix fogja $\mathbf{X}$-hez hasonló módon tartalmazni a transzformált adatokat. Ezek már a főkomponensek szerint lesznek kifejtve.

Fontos megjegyzés, hogy az SVD-ben használt három mátrix nem egyértelmű. A sajátvektorok, és sajátértékek sorrendje tetszőlegesen cserélhető. Elterjedt konvenció, hogy a sajátértékek (vagyis szinguláris értékek) szerint csökkenő sorrendben adjuk meg az értékeket, és vektorokat. Ez azért hasznos, mert így ha $\mathbf{Y}$ mátrixot szeretnénk $M < N$ dimenzióra redukálni, akkor az első M komponenst kell megtartani. Ez egyértelműsíti $\mathbf{\Sigma}$ mátrixot, de $\mathbf{U}$-t nem, mivel a sajátvektorokban még mindig vagy szabadságunk egy -1-gyel való szorzás erejéig. Ez a kifejtésnél nem okoz nagy gondot, ugyanis csak a kifejtési együtthatók előjelét befolyásolja, azonban néha érdemes odafigyelni rá.

A kifejtés úgy történik, hogy egy bármely kifejtendő mintát, aminek origóba toltját jelöljük $\mathbf{X}'$-vel, hasonló módon $\mathbf{U}$-val szorozzuk:

$$\mathbf{Y}' = \mathbf{X}'\mathbf{U} \tag{3}$$

Ezután a visszatranszformálás, mivel $\mathbf{U}$ ortogonális, így inverze a transzponáltja, a következőképp történik:

$$\mathbf{X}' = \mathbf{X}'\mathbf{U}\mathbf{U}^{\mathrm{T}} = \mathbf{Y}'\mathbf{U}^{\mathrm{T}} \tag{4}$$

Ha csak az első $M < N$ főkomponens szerint akarjuk kifejteni, akkor a mátrix szorzásnál N helyett csak M-ig kell mennie a szummának. Ha az origóba tolt, első M komponens szerint kifejtett, eredeti bázison ábrázolt adatsort $\mathbf{X}''$-vel jelöljük:

$$\mathbf{X}''_{ik} = \sum_{j=1}^{M} \mathbf{Y}'_{ij} \mathbf{U}^{\mathrm{T}}{}_{jk} \tag{5}$$

Ezután már csak az a dolgunk, hogy a kapott vektorokhoz hozzáadjuk a korábban kapott átlagot, mint nulladik kifejtési együttható. Így megkaptuk a kifejtett adatokat az eredeti bázison.

2. Technikai háttér

Linux Mint alatt dolgoztam, `g++` fordítóval, `C++` nyelven. A felhasznált csomagok: GSL (*GNU Scientific Library*), valamint LAPACK (*Linear Algebra PACKage*). Az SVD-hez a LAPACK `dgesvd` rutinját használtam, aminél beállítható, hogy mit szeretnénk kiszámoltatni vele. Nekünk csak az **U** mátrixra van szükségünk. Ez a rutin is használja azt a konvenciót, hogy a szinguláris értékek szerint csökkenő sorrendben adja meg az adatokat, így ezzel nem kell foglalkoznunk. A rutin FORTRAN-ban van írva, így az `extern` utasítással kell meghívnunk a programunkba.

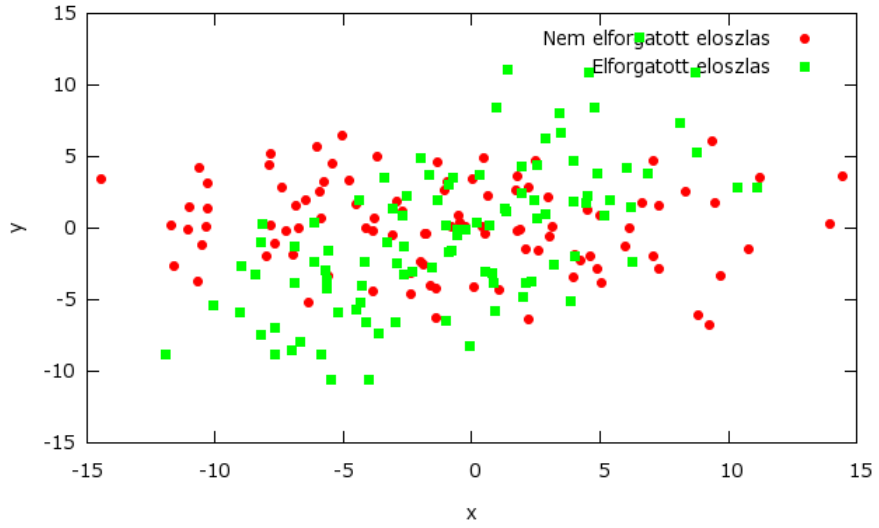
Az adatok ábrázolásást GNUPLOT-tal végeztem.

3. Próbaprogram generált adatsorral

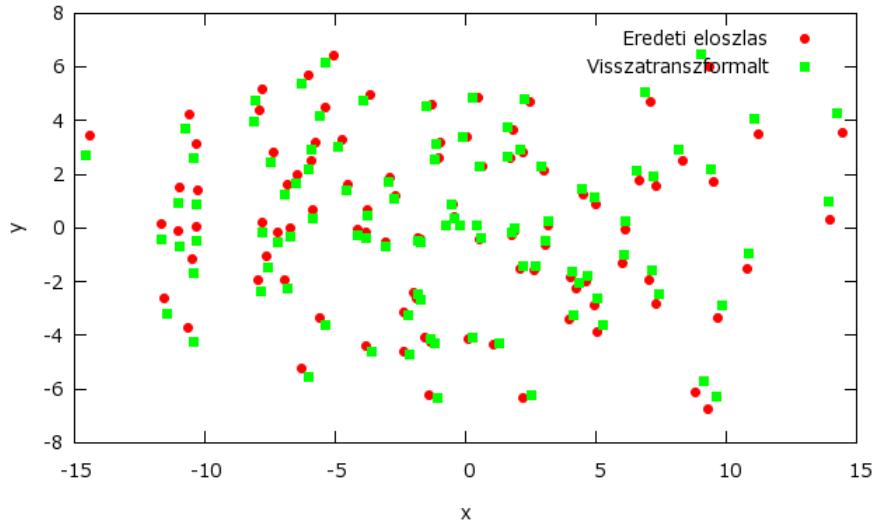
Hogy működőképes programot hozzak létre, ami értelmes eredményeket is ad, először véletlenszerűen generált adatokat hoztam létre, és ezeken végeztem el a főkomponens analízist, jól ellenőrzött körülmények között. A GSL függvénykönyvtár `gsl_rng_mt19937` véletlenszám generátorával, a `gsl_ran_gaussian` függvény segítségével $T = 100$ db, az ábrázolhatóság kedvéért $N = 2$ dimenziós Gauss eloszlású véletlen vektorral töltöttem fel az **X** mátrixot. A különböző komponensekhez külön-külön, különböző szórású eloszlásokat használtam. Mikor ezzel készen voltam, egy adott φ szöggel elforgattam az adatsort, hogy legyen mit visszatranszformálni. A PCA feladata, hogy az adatsort visszaforgassa, vagyis egy $-\varphi$ szögű forgatást hajtson végre. (*Megjegyzés:* természetesen a GSL biztosít számunkra olyan függvényt, ami egyből többdimenziós, bárhogyan álló Guass eloszlású értékeket ad, de én azért jártam el így, mert így a kapott **U** mátrixot közvetlenül összehasonlíthatjuk a forgásmátrix inverzével.)

A teszteket több φ szöggel, többféle szórási aránnyal több realizációra is megcsináltam, és a program jól működött. Egy $\varphi = 50°$-os, $\sigma_x = 7$, $\sigma_y = 3$ szórású példát konkrétan be is mutatok. A generált eredeti adatsor, és az

elforgatottja a 2. ábrán látható. Az elforgatott adatsor visszatranszformáltja az eredetivel összehasonlítva a 3. ábrán látható.



2. ábra. PCA-hoz létrehozott véletlen adatsor, és annak $\varphi = 50°$-os elforgatottja.



3. ábra. PCA-hoz létrehozott véletlen adatsor, és az elforgatás után visszatranszformált változata. Jól látható, hogy a két adathalmaz nagyjából jól illeszkedik.

Az egzakt visszaforgatás mátrixa a $-50°$-os forgásmátrix lenne, a valódi transzformációt viszont $\mathbf{U}$-val végezzük. A két mátrix számértékei:

$$\mathbf{U} = \begin{pmatrix} 0,680 & 0,733 \\ -0,733 & 0,680 \end{pmatrix} \approx \begin{pmatrix} 0,643 & 0,766 \\ -0,766 & 0,643 \end{pmatrix} = \mathbf{R}(-\varphi) \tag{6}$$

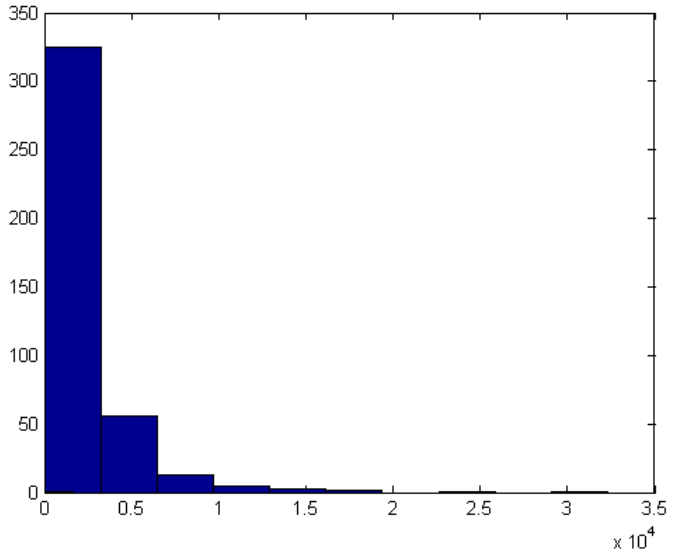
Az esetleges eltéréseket okozhatja egyrészt az, hogy az empirikus mintaátlag a konkrét realizációra eltér az elméleti várható értéktől, valamint a számítás is okozhat numerikus hibákat. Összességében azonban mondhatjuk, hogy a főkomponens analízis jól működött, a programot használhatjuk további problémák kezelésére.

4. Arcfelismerés

A módszert arra használom a következőkben, hogy a AT&T Laboratories Cambridge laboratóriumban készített, http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html címen elérhető arckép adatbázist beolvassam, és ezek PCA analízisét végezzem el a következő módon. Ezek a képek 92×112 pixeles, szürkeárnyalatos képek. Minden pixel 256 féle színértéket tud felvenni. A képeket tekinthetjük $N = 92 \times 112 = 10304$ dimenziós vektoroknak, aminek komponensei 256 értéket vehetnek fel. 40 alanyról 10-10 képet készítettek, vagyis az adatbázis gyakorlatilag 400 pont ebben a több, mint 10000 dimenziós térben. A dimenziószámból fakadó szabadsági fokok száma sokkal nagyobb, mint az adatpontok száma, ráadásul az arcoknak vannak bizonyos tulajdonságai, amik mindegyikre jellemzőek (2 szem, egy orr középen, stb.), így arra számítunk, hogy a pontok nem teljesen véletlenszerűen lesznek szétszórva ebben a sok dimenziós térben, hanem viszonylag jól valamilyen átlagarc körül fognak szórni, és az azonos alanyhoz tartozó fényképek kis, jól elkülönülő felhőket fognak alkotni. Mindezek alapján jogosan számíthatunk arra, hogy a fényképeket viszonylag jó közelítéssel leírhatjuk 10000-nél jóval kevesebb adattal is.

A képek `.pgm` formátumúak. Ezeknek van egy speciális headerük, ami tartalmazza a verziószámot, esetünkben ez `P5`, a felbontásra és a színmélységre tartozó indformációkat, valamint esetlegesen kommenteket. Ezután következnek folytonosan byte-onként a pixelértékek. A headert levágva a fájlokat betöltöttem, az említett átlagolást, és a főkomponens analízist elvégeztem. A sajátértékeket kimentettem, és az eloszlásukról MATLAB-bal hisztogramot készítettem.

A sajátértékek nagyjából exponenciálisan levágnak. Elméletileg 10304 darab van belőlük, de kevesebb, mint 400 db belőlük 0-tól különböző. Ez



4. ábra. Arc adatbázis kovarianciamátrixának sajátértékeloszlása.

is jelzi, hogy a sorfejtést elég csak bizonyos tagokig elvégezni, és százas nagyságrendűből már pontos értéket kapunk.