VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Papildomi duomenų vizualizavimo skyriai

Dimensijos mažinimo algoritmų pristatymas. PCA

Atliko: 3 kurso 2 grupės studentai:

Matas Amšiejus

Roland Gulbinovič

Darbo vadovė: dr. Jolita Bernatavičienė

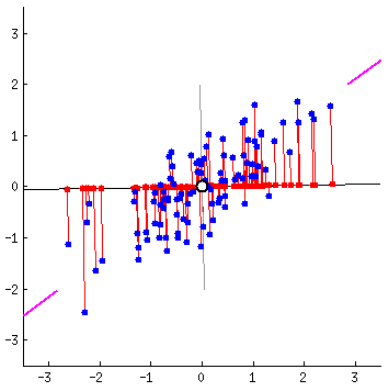
Vilnius, 2022

PCA principinių komponenčių analizė

PCA tikslas – sumažinti sudėtingų duomenų dimensiją į lengvai suprantamą, prarandant kuo mažiau informacijos. Tai pasiekti galima transformuojant duomenis pagal principines komponentes (toliau PC), kurios yra nekoreliuotos ir pirmosios išlaiko didžiausią dispersiją.

Kas yra principinės komponentės?

Principinės komponentės yra nauji kintamieji, sudaryti kaip tiesinės pradinių kintamųjų išraiškos. Duomenų išreiškimas per PC pagal didžiausią dispersiją leidžia mažinti dimensiją prarandant kuo mažiau informacijos taip atsisakant nereikšmingų komponenčių ir paliekant tik svarbiausias. Tačiau PC neturi realios reikšmės, nes jos yra išvestos tiesinės kintamųjų išraiškos, o ne tiesioginės reikšmės.



1 pav. Geometrinė PC reikšmė

Geometriškai, PC yra bandoma parinkti tokias, kad dispersija būtų didžiausia (išsaugoma daugiausiai informacijos). Tai reiškia, kad siekiama maksimizuoti taškų projekcijų (raudoni taškai) į principinę komponentę dispersiją. Trumpai tariant, principinės komponentės yra naujos ašys, skirtos išryškinti skirtumus tarp stebinių.

Tam, kad rastume optimalias principines komponentes reikia atlikti tokius žingsnius:

- Sukuriame matricą be klasės kintamojo;
- Normuojame duomenis pagal vidurkį ir dispersiją;
- Randame kovariacijos matricą;
- Suskaičiuojame tikrines reikšmes *(eigenvalues*) ir tikrinius vektorius *(eigenvectors);*
- Parenkame principines komponentes su didžiausiomis dispersijomis;
- Vizualizuojame pagal PC.

Geriausia informaciją perteikti kartu darant pavyzdį, kurį atliksime su iris duomenimis.

Pirma sukuriame matricą atrinkę tik vilkdalgių parametrus (viso 4 : *sepal length, sepal width, petal length, petal width*)) ir sunormuojame pagal vidurkį ir dispersiją:

$$x_{norm} = \frac{x - \bar{x}}{\sqrt{\sigma^2}}.$$

Dabar kiekvienas atributas turi vidurkį 0 ir dispersiją 1.

Dabar ieškosime kovariacijos matricos. Kovariacijos formulė:

$$C_{x,y} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}).$$

Bet kadangi duomenys jau sunormuoti, naudosime formulę

$$C = \frac{X^T X}{n-1},$$

kur X – mūsų turima duomenų matrica. Bendra formulė kovariacijos matricos (kai duomenų matrica yra $n \times k$) atrodo taip (kai sunormuota pagal vidurkį ir dispersiją, tai ant diagonalės bus vienetai):

$$C = \begin{pmatrix} C_{1,1} = 1 & C_{1,2} & \dots & C_{1,k} \\ C_{2,1} & C_{2,2} = 1 & \dots & C_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ C_{k,1} & C_{k,2} & \dots & C_{k,k} = 1 \end{pmatrix}$$

Savo pavyzdyje gauname $k \times k$ matricą, kur k – atributų skaičius (4).

1 lentelė. Kovariacijos matrica

```
([[ 1.        , -0.10936925,  0.87175416,  0.81795363],
  [-0.10936925,  1.        , -0.4205161 , -0.35654409],
  [ 0.87175416, -0.4205161 ,  1.        ,  0.9627571 ],
  [ 0.81795363, -0.35654409,  0.9627571 ,  1.        ]])
```

Dabar skaičiuosime tikrines reikšmes ir tikrinius vektorius. Tikrinis vektorius – parodo ašių kryptis, kuriose dispersijos yra didžiausios (daugiausiai informacijos), kurias vadiname principinėmis komponentėmis. Tikrinės reikšmės – tai yra koeficientai priskirti kiekvienam tikriniam vektoriui, kurie parodo dispersiją aplink kiekvieną principinę komponentę. Tikrinio vektoriaus dydis ir kiekis bus lygus mūsų turimų atributų skaičiui. Kiekvienas tikrinis vektorius turi tikrinę reikšmę. Tam, kad rasti šias reikšmes, naudosime formulę

$$C \cdot v = \lambda \cdot v,$$

kur C – mūsų kovariacijos matrica, v – tikrinis vektorius, λ – tikrinė reikšmė. Tam, kad gautume tikrinę reikšmę, skaičiuojame:

$$C \cdot v - \lambda \cdot v = 0$$
$$\det(C - \lambda I) = 0,$$

kur λI yra vienetinė matrica, tik ant diagonalės yra λ.

$$\det\left(\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1k} \\ c_{21} & c_{22} & \cdots & c_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ c_{k1} & c_{k2} & \cdots & c_{kk} \end{pmatrix} - \begin{pmatrix} \lambda & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & \lambda \end{pmatrix}\right).$$

Taigi mūsų atveju (tęsiant su *iris* duomenimis) gauname tokią matricą:

$$det\left(\begin{pmatrix} 1-\lambda & -0{,}11 & 0{,}87 & 0{,}82 \\ -0{,}11 & 1-\lambda & -0{,}42 & -0{,}36 \\ 0{,}87 & -0{,}42 & 1-\lambda & 0{,}96 \\ 0{,}82 & -0{,}36 & 0{,}96 & 1-\lambda \end{pmatrix}\right) = 0$$

Išsisprendę gauname tokią formulę:

$$\lambda^4 - 4\lambda^3 + 3{,}33\lambda^2 - 0{,}47\lambda + 0{,}01 = 0.$$

Ją išsisprendę gauname visas 4 tikrines reikšmes:

$$\lambda_1 = 0{,}02;\ \lambda_2 = 0{,}14;\ \lambda_3 = 0{,}92;\ \lambda_4 = 2{,}91.$$

Jau iškart galime nustatyti svarbiausias komponentes, nes tikrinės reikšmės parodo dispersiją aplink PC. Taip pat galime suskaičiuoti, kokią dalį visos dispersijos aprašo kiekviena PC. Tą randame pasinaudojus formule:

$$\%var(t) = \frac{\lambda_t}{\sum_{i=1}^{k} \lambda_i},$$

Kur *t* – principinės komponentės, kurios dispersijos dalį norime skaičiuoti, indeksas. Tai mūsų *iris* atveju gauname, kad dispersija aplink λ_4 PC bus 72,9 %, o λ_3 – 23,1 %. Tarkime brėšime dvimatį grafiką, tai atrenkame dvi didžiausių dispersijų komponentes (λ_4, λ_3).

Dabar ieškosime tikrinių vektorių. Vėl pasinaudosime formule

$$C \cdot v = \lambda \cdot v,$$

$$\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1k} \\ c_{21} & c_{22} & \cdots & c_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ c_{k1} & c_{k2} & \cdots & c_{kk} \end{pmatrix} \cdot \begin{bmatrix} a \\ b \\ \vdots \\ k \end{bmatrix} = \lambda \cdot \begin{bmatrix} a \\ b \\ \vdots \\ k \end{bmatrix},$$

tačiau dabar turime λ reikšmes. Tam, kad rastume eigenvector, į lygti atskirai statome gautas eigenvalues. Taigi, įstačius λ_4 gauname

$$\begin{bmatrix} 1 & -0{,}11 & 0{,}87 & 0{,}82 \\ -0{,}11 & 1 & -0{,}42 & -0{,}36 \\ 0{,}87 & -0{,}42 & 1 & 0{,}96 \\ 0{,}82 & -0{,}36 & 0{,}96 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = 2{,}91 \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}.$$

Viską išreiškus per d kintamąjį, gauname visus kintamuosius su daugikliais (d daugiklis bus 1):

$$x \cdot a; y \cdot b; z \cdot c; d.$$

Tačiau juos dar reikia normalizuoti taip, kad vektoriaus ilgis būtų vienetas. Tam mes atliksime normavimą:

$$\frac{h}{\sqrt{x^2+y^2+z^2+1}},$$

kur h – kiekvienas gautas daugiklis (x,y,z,1). Taigi, po normavimo mūsų λ_4 tikrinis vektorius v_1 bus

$$v_1 = \begin{pmatrix} 0{,}52 \\ -0{,}26 \\ 0{,}58 \\ 0{,}57 \end{pmatrix}.$$

v_2 gaunamas analogiškai:

$$v_2 = \begin{pmatrix} -0{,}37 \\ -0{,}93 \\ -0{,}02 \\ -0{,}07 \end{pmatrix}.$$

Galiausiai sudedame vektorius į vieną matricą (pagrindinių komponenčių matrica):

$$v = \begin{pmatrix} 0{,}52 & -0{,}37 \\ -0{,}26 & -0{,}93 \\ 0{,}58 & -0{,}02 \\ 0{,}57 & -0{,}07 \end{pmatrix}.$$

Taigi, kad gautume galutinius transformuotus duomenis pagal PC, sudauginame:

$$X_{norm} \cdot v = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{pmatrix} \cdot \begin{pmatrix} v_{11} & \cdots & v_{1d} \\ v_{12} & \cdots & v_{2d} \\ \vdots & \vdots & \vdots \\ v_{k1} & \cdots & v_{kd} \end{pmatrix},$$

Kur d – mūsų pasirinkta dimensija, kurioje norėsime atvaizduoti duomenis (1,2,3). Skaičiavimas su mūsų pavyzdžiu atrodo taip:

$$X_{norm} \cdot v = \begin{pmatrix} -0{,}9 & 1{,}03 & -1{,}34 & -1{,}31 \\ -1{,}14 & -0{,}12 & -1{,}34 & -1{,}31 \\ -1{,}39 & 0{,}34 & -1{,}40 & -1{,}31 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \cdot \begin{pmatrix} 0{,}52 & -0{,}37 \\ -0{,}26 & -0{,}93 \\ 0{,}58 & -0{,}02 \\ 0{,}57 & -0{,}07 \end{pmatrix} = \begin{pmatrix} -2{,}26 & -0{,}51 \\ -2{,}09 & 0{,}66 \\ -2{,}37 & 0{,}32 \\ \vdots & \vdots \end{pmatrix}.$$

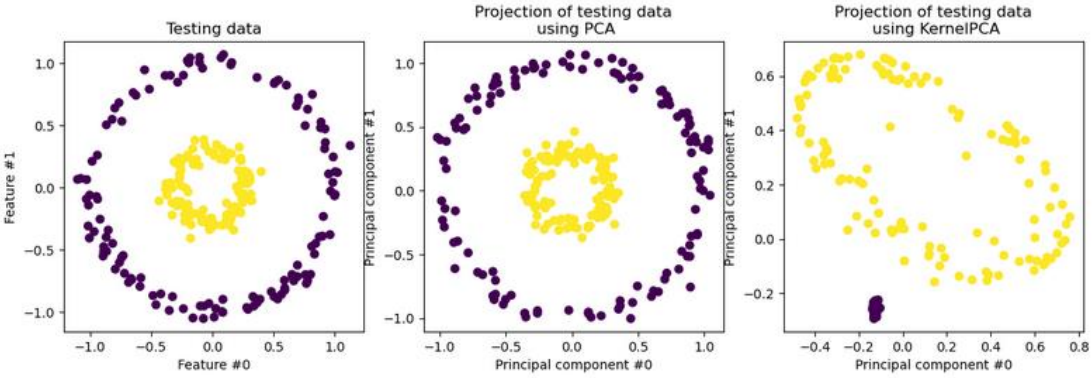
Viskas, turime transformuotus duomenis.

PCA modifikacijos

Incremental PCA (IPCA)

Ši modifikacija paprastai yra naudojama kai duomenys užima labai daug vietos ir kompiuteryje pradeda trūkti atminties. Papildomai yra sprendžiama minimizavimo problema, kur bandoma sumažinti matricos rangą prarandant kuo mažiau informacijos.

Kernel PCA



2 pav. PCA ir kernel PCA palyginimas

PCA transformuoja duomenis tiesiškai, juos centruojant ir pakeičiant skales pagal dispersiją. Tačiau kai duomenys yra išsidėstę izotropiškai (turi panašius atstumus nuo centro), jie yra atvaizduojami identiškai (2 pav.), negaunama jokia nauja informacija. Tačiau panaudojus *kernel* PCA, mes galime atlikti ne tiesinę projekciją taip tiksliau atskirdami skirtingas grupes.

jei vuzualizuojame tai imame pirmas dvi, bet realiai gyvenime galime pasiziureti didesni spektra komponenciu, nes ten galime rasti isskirtinumus.

Šaltiniai

The Math Behind: Everything about Principle Component Analysis (PCA). Autorius: Sera Giz Ozel. 2021 04 08. Nuoroda: The Math Behind: Everything about Principle Component Analysis (PCA) | DataDrivenInvestor

StatQuest: Principal Component Analysis (PCA), Step-by-Step (vaizdo įrašas). Autorius: Josh Starmer. 2018 04 12. Nuoroda: (32) StatQuest: Principal Component Analysis (PCA), Step-by-Step - YouTube

PCA : the math - step-by-step with a simple example. Autorius: „TileStats“. 2021 02 13. Nuoroda: (33) PCA : the math - step-by-step with a simple example - YouTube

A Step-by-Step Explanation of Principal Component Analysis (PCA). Autorius: Zakaria Jaadi. 2021 04 01. Nuoroda: A Step-by-Step Explanation of Principal Component Analysis (PCA) | Built In

Understanding the Covariance Matrix. Autorius: Marvin Lanhenke. 2021 12 29. Nuoroda: Understanding the Covariance Matrix | by Marvin Lanhenke | Towards Data Science

Kernel PCA dokumentacija. Autoriai: Mathieu Blondel, Andreas Mueller, Guillaume Lamaitre. Nuoroda: Kernel PCA — scikit-learn 1.0.2 documentation