Линеара Алгебра

Проект за лабораториски вежби

Тема: Principal Component Analysis – PCA

https://github.com/ilijavishinov/LinearAlgebra_Project

Изработил:

Илија Вишинов

161078

Ментори:

Марија Михова

Жанета Попеска

1. Опис

Principal Component Analysis (PCA) е метод кој се користи во голема мера во областа на машинското учење и статистичката анализа на податоци. Потребата од ПЦА се истакнува во проблеми во кои сакаме да избегнеме curse of dimensionality. Дури и да немаме случај во кој се справуваме со curse of dimensionality, намалувањето на бројот на димензии на податочното множество може драстично да го намали времето на тренирање на моделите и со тоа да ја зголеми брзината на креирање на прототипни модели и наоѓањето на оптималните хиперпараметри за проблемот.
PCA алгоритмот го трансформира податочното м-во, така што креира нова ортогонална база во која на прво место доаѓа насоката во податочното м-во во која има најмогу варијанса, потоа на второ место насоката која е втора по најголема варијанса итн. Овие насоки се сопствените вредности на матрицата на коваријанса на податочното м-во и се наречени principal components. Интуицијата позади ова е што сакаме колоните во м-вото да не се корелирани едни со други, односно матрицата на коваријанса да е дијагонална, што се постигнува со наоѓањето на нејзините сопствени вредности и вектори објаснето подетално во следниот дел. Бидејќи трансформацијата се основа на варијансата, клучно е претходно податоците да се нормализираат со тоа што нивната стандардна девијација ќе се изедначи. За полесна пресметка на матрицата на коваријасна практично е податоците на сите параметри да се центрирани во 0, односно нивната средна вредност да е 0. Заради доведувањето на средната вредност на 0 и изедначувањето на варијансата, врз податоците најпрактично е да се направи z-score нормализација.
Количеството информација релативно на другите колони на м-вото кое principal components го опишуваат е дадено од соодветните сопствени вредности на сопствените вектори. Ако сите сопствени вектори се соодветни на различни сопствени вредности, тогаш м-вото на сопствени вектори е независно, што е секогаш случај со симетрични матрици.
Два методи за пронаоѓањето на сопствени вектори и сопствени вредности се EVD (eigen value decomposition) и SVD (singular value decomposition). Постојат и други како ортогонална декомпозиција и декомпозиција на Schur кои нема да се разгледани во оваа проектна задача поради обемноста.
Во зависност од потребата на проблемот за кој се употребува PCA, може да се изберат првите n базни вектори, односно principal components и да се проектира податочното м-во во тој векторски потпростор.

2. Доказ и математичка нотација

За нормализација на податоците се користи z-score нормализација

$$z = \frac{(X - \ mean)}{std.dev.}$$

Нормализацијата се применува на секоја колона од матрицата која го претставува податочното множество. Ова ги нормализира сите колони со тоа што тие сега имаат

$$mean = 0, \quad std.dev. = 1$$

Во случајот на нормализирани податоци во матрицата на податоци X, матрицата на коваријанса се пресметува на следниот начин:

$$C_X = \ X^T X$$

Целта на алгоритмот е да се трансформира податочното м-во така што колоните на податочната матрица ќе имаат најмала меѓусебна корелација, односно матрицата на коваријанса целиме да е дијагонална матрица. Со други зборови сакаме да најдеме матрица на линеарна трансформација T, која ќе го трансформира податочното м-во односно матрица во матрица Y, така што матрицата на коваријанса C_Y ќе има некорелирани колони. Матрицата на коваријанса на трансформираното м-во може да се запише како:

$$C_Y = Y^T Y = (XT)^T (XT) = \ T^T (X^T X) T = T^T C_X T$$

Бидејќи матрицата на коваријанса е реална и симетрична, а сите реални симетрични матрици може да се дијаголанизираат, и теоремата за декомпозиција на симетрични матрици гласи:

$$\mathrm{A} = Q \Lambda Q^T,$$

каде Q е ортогонална матрица која ги содржи сопствените вредности на A по своите колони и Λ е реална дијагонална матрица која ги содржи сопствените вредности на A. Кога ќе се замеени C_X so $Q \Lambda Q^T$, се добива:

$$C_Y = T^T Q \Lambda Q^T T$$

Бидејќи за ортогонални матрици, нивната транспонирана е еднаква на нивната инверзна, од равенката е очигледно дека ако се избере T = Q, се добива дијагонална матрица за C_Y. Заменуваме:

$$C_Y = Q^T Q \Lambda Q^T Q = I \Lambda I = \Lambda$$

Со тоа што ја одбравме матрицата на линеарна трансформација T да е еднаква на Q, чии колони се сопствените вредности на матрицата на коваријанса на матрицата на податоци X и се доби дијагонална матрица, се покажува дека за трансформирање на податочното множество во простор во кој неговите колони се најмалку корелирани, потребни се сопствените вредности и вектори на матрицата на коваријанса на тоа множество.

Пресметувањето на сопствените вредности на квадратна симетриччна матрица (која може да се дијагонализира) може да се направи според фундаменталното својство на сопствените вредности. Вектор x е сопствен вектор ако го задоволува равенството:

$$\mathrm{A}\mathbf{x} = \lambda\mathbf{x}$$

Од ова равенство изведуваме:

$$\lambda \boldsymbol{x} - A\boldsymbol{x} = \mathbf{0}$$

$$\lambda I \boldsymbol{x} - A\boldsymbol{x} = \mathbf{0}$$

$$(\lambda I - A)\boldsymbol{x} = \mathbf{0}$$

Оваа равенка има ненулто репение ако и само ако $\lambda I - A$ е сингуларна матрица, односно нејзината детерминанта е 0. Со развивање на оваа равенка се добива полином од n-ти степен по ламбда.

$$\lambda^{\mathrm{n}} + \mathrm{c}_{n-1}\lambda^{\mathrm{n}-1} + \ldots + \mathrm{c}_1\lambda + \mathrm{c}_0 = 0$$

Корените на равенката се сопствените вредности на матрицата A. Откако ќе се добијат сопствените вредности се решава системот равенки:

$$(\lambda_i I - A)\boldsymbol{x} = \mathbf{0},$$

Делот во заградата може да се претстави и како $A - \lambda_i I$, што е поинтуитивно за претставување со матрици. Од дијагоналата на матрицата A се одзема сопствената вредност која е сместена по дијагоналата на матрицата матрица $\lambda_i I$ и оваа матрица се множи со векторот x. Решението го дава сопствениот вектор за таа сопствена вредност.

Друг начин на изведување на сопствените вредности и вектори е прекеу SVD – singular value decomposition. SVD е попрактичен метод бидејќи важи во генерален случај за сите матрици, квадратни и неквадратни. Проблемот со сопствени векотри и вредности во случајот на неквадратна матрица е тоа што еден вектор може да премине од m во n

димензии и со тоа се губи смислата на сопствените вредности и вектори.
Но, во нашиот случај матрицата на коваријанса е квадратна и симетрична и иако пресметувањето на сопствени вредности и вектори е доволно добре преку EVD, ќе го разгледаме и случајот на SVD. Теоремата на СВД гласи

$$A = U\Sigma V$$

Матрицата U ги содржи левите сингуларни вектори, а матрицата V ги содржи десните сингуларни вредности. Σ е дијагонална матрица која ги содржи сингуларните вредности. Надвор од темата на проектот, интуицијата позади оваа теорема е дека секоја матрица може да се факторизира како $U\Sigma V$, каде U и V се матрици на ротација а Σ е матрица на скалирање. Во случајот на симетрична квадратна матрица, резултатот на SVD и EVD е идентичен со тоа што матрицата од десни сингуларни вектори кај SVD ќе биде трансопонираната матрица од онаа на леви сингуларни вектори, а сингуларните вредности ќе се еднакви на сопствените вредности.

3. Имплементација

За имплементација на алгоритмот е даден линк со кодот на github.

Еве го повторно: https://github.com/ilijavishinov/LinearAlgebra_Project

4. Теориска комплексност

Скалирањето на податоците не влегува во самиот алгоритам, но ќе биде вклучено во анализата на комплексност бидејќи е задолжителен чекор за добивање на правилни резултати.

Нека податочното м-во се наоѓа во матрица X со димензии m, n (m редови – samples, n колони – features).

Скалирањето на податоците вклучува изминување на податоците еднаш за пресметување на средната вредност и стандардната девијација и уште еднаш за нормализирање на секоја вредност, доколку овој процес е паралелизиран има комплексност O(m^2), а доколку е секвенцијален има комплексност O(m^2n).

Нареден чекор е пресметувањето на матрицата на коваријанса. За нејзино пресметување, потребно е да се помножи X^T (со димензија [nxm]) со X (со димензија [mxn]. За транспонирањето на матрицата е потребно O(mxn) ако таа се генерира на страна, а оптимално би било ако се пристапува по индекс и не се чува на страна со што се елиминира оваа временска комплексност. Множењето има комплексност $O(nm^2)$. Ова се добива бидејќи секој (I,j) елемент од резултантната матрица е добиен како производ на i-тиот ред на левата матрица и j-тата колона на десната матрица. Пресметувањето на овој елемент има временска комплексност O(m). Ова се прави за секој елемент на резултантанта матрица која има димензии [nxn], и нејзиното изминување има комплексност $O(n^2)$.

Пресметувањето на сопствените вредности се состои од решавање на полиномна равенка од n-ти степен, што има комплексност ___не сум сигурен___.

За сопствените вектори потребно е да се решат системите равенки $(\lambda I - A)\boldsymbol{x} = \mathbf{0}$ за секој λ_i кои имаат комплексност $O(n^3)$. Матрицата има вкупно n сопствени вредности па комплексноста за наоѓање на сите сопствени вредности во најлош случај е $O(n^4)$. Ова е бидејќи елиминацијата на ненултите елементи под секој водечки елемент има комплексност $O(n^2)$ (менување редици и множење) во најлош случај, а O(1) vo najdoba па повторувајќи го ова за секоја колона се добива $O(n^3)$ во најлош случај а $O(n^2)$ во најдобар. Во најдобар случај целокупна комплексност е $O(n^2)$ за на пресметување на сите сопствени вредности.

5. Анализа над симулирани податоци и споредба на брзината на различни методи

Скриптата за анализа над симулирани податоци може да се види на:

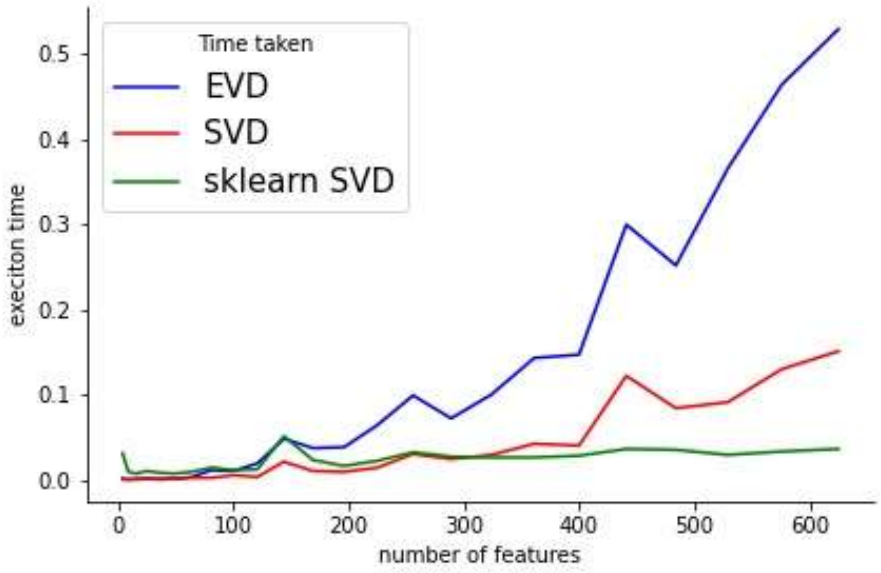
https://github.com/ilijavishinov/LinearAlgebra_Project/blob/master/py/Analysis_SimulatedData.ipynb

Во неа се направени неколку експерименти:

- PCA врз независно генерирани гаусови features.
    - Очекувања: сите principal components објаснуваат прилично еднакво количечство од варијансата/информацијата во м-вото.
    - Резултати: Во согласностсо очекувањата
- PCA врз неколку независно генерирани гаусови features и изведени features од нив со додаден шум.

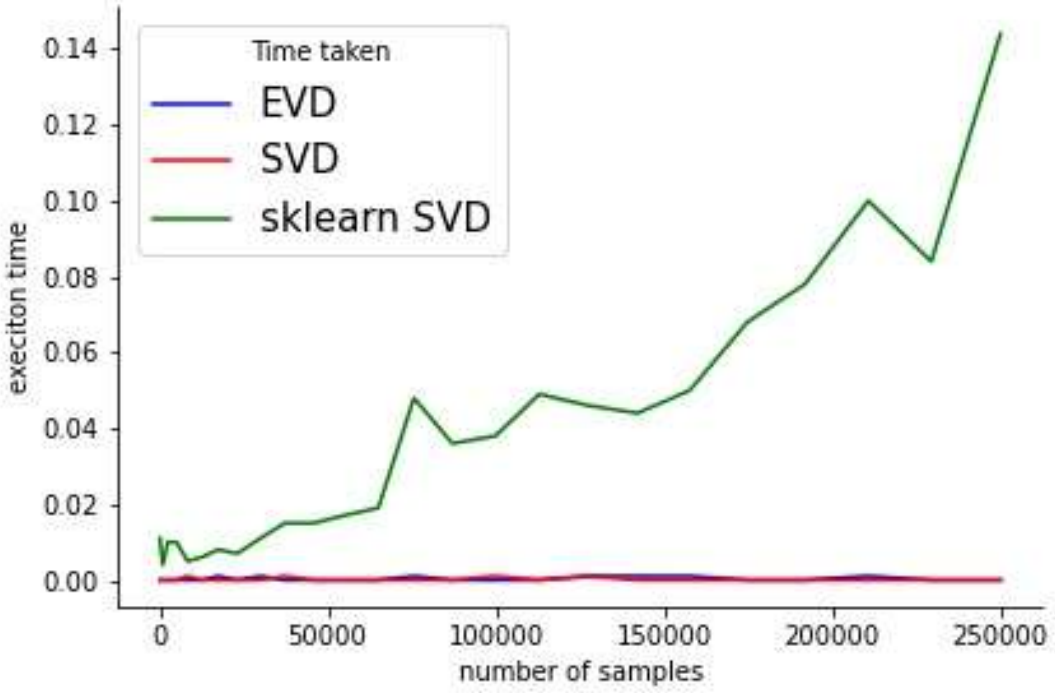
- Очекувања: првите principal components колку што има независно генерирани објаснуваат најголем дел од количеството информација во м-вото. После нив следат оние кои се слабо корелирани, па посилно корелираните со најмалку објаснување на информацијата.
- Резултати: Во согласностсо очекувањата

- PCA врз независно генерирани гаусови features и изведени од нив со перфектна корелација.
  - Очекувања: првите principal components колку што има независно генерирани го објаснуваат целото количествоинформација во м-вото. Останатите сопствени вредности т.е. количество објаснета информација од останатите features се 0.
  - Резултати: Во согласностсо очекувањата
- PCA врз експоненцијално растечки број на колони во матрицата на податоци. Забелешка: во временската комплексност е вклучена и нормализацијата на податоците.
  - Очекувања: За рачно имплементираните методи, приближно експоненцијален раст во најлош случај.
  - Резултати: EVD има најлоши перформанси и експоненцијален раст. Потоа доаѓа рачно имплементираниот SVD. Споредба беше направена и со PCA од библиотеката на sklearn, кој е многу добро оптимизиран, па покажа најдобро време на извршување.



- PCA врз експоненцијално растечка големина на бројот на samples – големина на features. Забелешка: во временската комплексност е вклучена и нормализацијата на податоците.

- Очекувања: Приближно еднакво време на извршување за рачно имплементираните методи бидејќи после пресметување на матрицата на коваријанса, димензијата на матриците е иста за секој метод.
- Резултати: Идентични временски перформанси на рачно имплементираните EVD и SVD. Готовите функции од пакетот sklearn дадоа прилично лоши резултати со вертикално зголемување на податочното м-во.



6. Анализа над симулирани податоци и споредба на брзината на различни методи

Скриптата за анализа над симулирани податоци може да се види на:

https://github.com/ilijavishinov/LinearAlgebra_Project/blob/master/py/Analylsis_RealData.ipynb

Анализирани се реалните податочни множества вградени во пакетот sklearn. Анализирано е колку principal components односно димензии се потребни за да се опише секое м-во со дополнување со графици и методи од машинско учење кои се во тек.