SVD ile PCA Hesaplamak

PCA bolumunde anlatilan yontem temel bilesenlerin hesabinda ozdegerler ve ozvektorler kullandi. Alternatif bir yontem Tekil Deger Ayristirma (Singular Value Decomposition -SVD-) uzerinden bu hesabi yapmaktir. SVD icin Lineer Cebir Ders 29'a bakabilirsiniz. Peki ne zaman klasik PCA ne zaman SVD uzerinden PCA kullanmali? Bir cevap belki mevcut kutuphanelerde SVD kodlamasinin daha iyi olmasi, ayristirmanin ozvektor / deger hesabindan daha hizli isleyebilmesi [6].

Ayrica birazdan gorecegimiz gibi SVD, kovaryans matrisi uzerinde degil, A'nin kendisi uzerinde isletilir, bu hem kovaryans hesaplama asamasini atlamamizi, hem de kovaryans hesabi sirasinda ortaya cikabilecek numerik puruzlerden korunmamizi saglar (cok ufak degerlerin kovaryans hesabini bozabilecegi literaturde bahsedilmektedir).

PCA ve SVD baglantisina gelelim:

Biliyoruz ki SVD bir matrisi su sekilde ayristirir

$A = USV^T$

U matrisi $n \times n$ dikgen (orthogonal), V ise $m \times m$ dikgen. S'in sadece kosegeni uzerinde degerler var ve bu σ_j degerleri A'nin tekil degerleri (singular values) olarak biliniyor.

Simdi A yerine AA^T koyalim, yani A'nin kovaryans matrisinin SVD ayristirmasini yapalim, acaba elimize ne gececek?

$AA^T = (USV^T)(USV^T)^T$

$= (USV^T)(VS^TU^T)$

$= USS^TU^T$

S bir kosegen matrisi, o zaman SS^T matrisi de kosegen, tek farkla kosegen uzerinde artik σ_j^2 degerleri var. Bu normal.

SS^T yerine Λ sembolunu kullanalim, ve denklemi iki taraftan (ve sagdan) U ile carparsak (unutmayalim U ortanormal bir matris ve $U^TU = I$),

$AA^TU = U\Lambda U^TU$

$AA^TU = U\Lambda$

Son ifadeye yakindan bakalim, U'nun tek bir kolonuna, u_k diyelim, odaklanacak olursak, ustteki ifadeden bu sadece kolona yonelik nasil bir esitlik cikartabilirdik? Soyle cikartabilirdik,

$(AA^T)u_k = \sigma^2 u_k$

Bu ifade tanidik geliyor mu? Ozdeger / ozvektor klasik yapisina eristik. Ustteki esitlik sadece ve sadece eger u_k, AA^T'nin ozvektoru ve σ^2 onun ozdegeri ise gecerlidir. Bu esitligi tum U kolonlari icin uygulayabilecegimize gore demek ki U'nun kolonlarinda AA^T'nin ozvektorleri vardir, ve AA^T'nin ozdegerleri A'nin tekil degerlerinin karesidir.

Bu muthis bir bulus. Demek ki AA^T'nin ozektorlerini hesaplamak icin A uzerinde SVD uygulayarak U'yu bulmamiz yeterli, kovaryans matrisini hesaplamak bile gerekmiyor! AA^T ozdegerleri uzerinde buyukluk karsilastirmasi icin ise A'nin tekil degerlerine bakmak yeterli!

Ornek

Ilk PCA yazisindaki ornege donelim, ve ozvektorleri SVD uzerinden hesaplatalim. Once eig usulu ile

```
from pandas import *
data = read_csv("testSet.txt",sep="\t",header=None)
means = mean(data, axis=0)
meanless_data = data - means
cov_mat = cov(meanless_data, rowvar=0)
eigs,eigv = linalg.eig(mat(cov_mat))
eigv
```

```
matrix([[-0.85389096, -0.52045195],
        [ 0.52045195, -0.85389096]])
```

Simdi de SVD usulu ile

```
U,s,Vt = svd(meanless_data.T,full_matrices=False)
U
```

```
array([[-0.52045195, -0.85389096],
       [-0.85389096,  0.52045195]])
```

```
np.dot(U.T,U)
```

```
array([[ 1.,  0.],
       [ 0.,  1.]])
```

Goruldugu gibi ayni ozvektorleri bulduk.

New York Times Yazıları Analizi Simdi daha ilginc bir ornege bakalim. Bir arastirmaci belli yillar arasindaki NY Times makalelerinde her yazida hangi kelimenin kac kere ciktiginin verisini toplamis [1,2,3], bu veri 4000 kusur kelime, her satir (yazi) icin bir boyut (kolon) olarak kaydedilmis. Bu veri nytimes.csv uzerinde ek bir normalize isleminden sonra, onun uzerinde boyut indirgeme yapabiliriz.

Veri setinde her yazi ayrica ek olarak sanat (arts) ve muzik (music) olarak etiketlenmis, ama biz PCA kullanarak bu etiketlere hic bakmadan, verinin boyutlarini azaltarak acaba verinin "ayrilabilir" hale indirgenip indirgenemedigine bakacagiz. Sonra etiketleri veri ustune koyup sonucun dogrulugunu kontrol edecegiz.

Bakmak derken veriyi (en onemli) iki boyuta indirgeyip sonucu grafikleyecegiz. Illa 2 olmasi gerekmez tabii, 10 boyuta indirgeyip (ki 4000 kusur boyuttan sonra bu hala muthis bir kazanim) geri kalanlar uzerinde mesela bir kumeleme algoritmasi kullanabilirdik.

Ana veriyi yukleyip birkac satirini ve kolonlarini gosterelim.

```
from pandas import *
import numpy.linalg as lin
nyt = read_csv ("nytimes.csv")
labels = nyt['class.labels']
nyt.ix[:8,102:107]
```

```
   after  afternoon  afterward  again  against
0      1          0          0      0        0
1      1          1          0      0        0
2      1          0          0      1        2
3      3          0          0      0        0
4      0          1          0      0        0
5      0          0          0      1        2
6      7          0          0      0        1
7      0          0          0      0        0
8      0          0          0      0        0
```

Yuklemeyi yapip sadece etiketleri aldik ve onlari bir kenara koyduk. Simdi onemli bir normalizasyon islemi gerekiyor - ki bu isleme ters dokuman-frekans agirliklandirmasi (inverse document-frequency weighting -IDF-) ismi veriliyor - her dokumanda aşırı fazla ortaya cikan kelimelerin onemi ozellikle azaltiliyor, ki diger kelimelerin etkisi artabilsin.

IDF kodlamasi alttaki gibidir. Once class.labels kolonunu atariz. Sonra "herhangi bir deger iceren" her hucrenin 1 digerlerinin 0 olmasi icin kullanilan DataFrame uzerinde astype(bools) isletme numarasini kullaniriz, boylece asiri buyuk degerler bile sadece 1 olacaktir. Bazi diger islemler sonrasi her satiri kendi icinde tekrar normalize etmek icin o satirdaki tum degerlerin karesinin toplaminin karekokunu aliriz ve satirdaki tum degerler bu karekok ile bolunur. Buna oklitsel (euclidian) normalizasyon denebilir.

Not: Oklitsel norm alirken toplamin hemen ardindan cok ufak bir 1e-16 degeri eklememize dikkat cekelim, bunu toplamin sifir olma durumu icin yapiyoruz, ki sonra sifirla bolerken NaN sonucundan kacinalim.

```
nyt = nyt.drop(['class.labels'],axis=1)
freq = nyt.astype(bool).sum(axis=0)
freq = freq.replace(0,1)
w = np.log(float(nyt.shape[0])/freq)
nyt = nyt.apply(lambda x: x*w,axis=1)
nyt = nyt.apply(lambda x: x / np.sqrt(np.sum(np.square(x))+1e-16), axis=1)
```

```
nyt=nyt.ix[:,1:] # ilk kolonu atladik
nyt.ix[:8,102:107]
```

```
   afterward     again   against       age  agent
0          0  0.000000  0.000000  0.051085      0
1          0  0.000000  0.000000  0.000000      0
2          0  0.021393  0.045869  0.000000      0
3          0  0.000000  0.000000  0.000000      0
```

```
4          0  0.000000  0.000000  0.000000      0
5          0  0.024476  0.052480  0.000000      0
6          0  0.000000  0.008536  0.000000      0
7          0  0.000000  0.000000  0.000000      0
8          0  0.000000  0.000000  0.000000      0
```

SVD yapalim

```
nyt = nyt - nyt.mean(0)
u,s,v = lin.svd(nyt.T,full_matrices=False)
```

```
s[:10]
```

```
array([ 1.41676764,  1.37161893,  1.31840061,  1.24567955,  1.20596873,
        1.18624932,  1.15118771,  1.13820504,  1.1138296 ,  1.10424634])
```

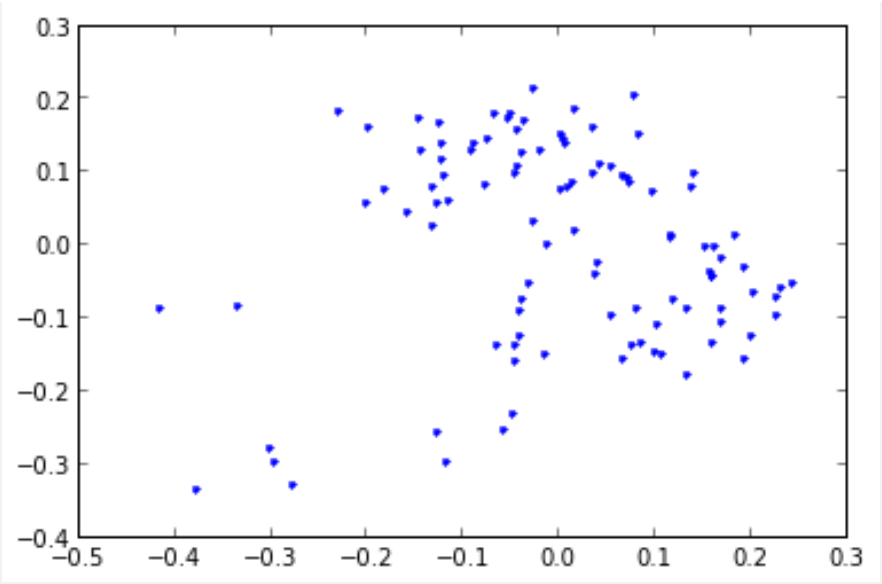
```
u.shape
```

```
(4430, 102)
```

SVD'nin verdigi u icinden iki ozvektoru seciyoruz (en bastakiler, cunku Numpy SVD kodu bu ozvektorleri zaten siralanmis halde dondurur), ve veriyi bu yeni kordinata izdusumluyoruz.

```
proj = np.dot(nyt, u[:,:2])
proj.shape
plot(proj[:,0],proj[:,1],'.')
```

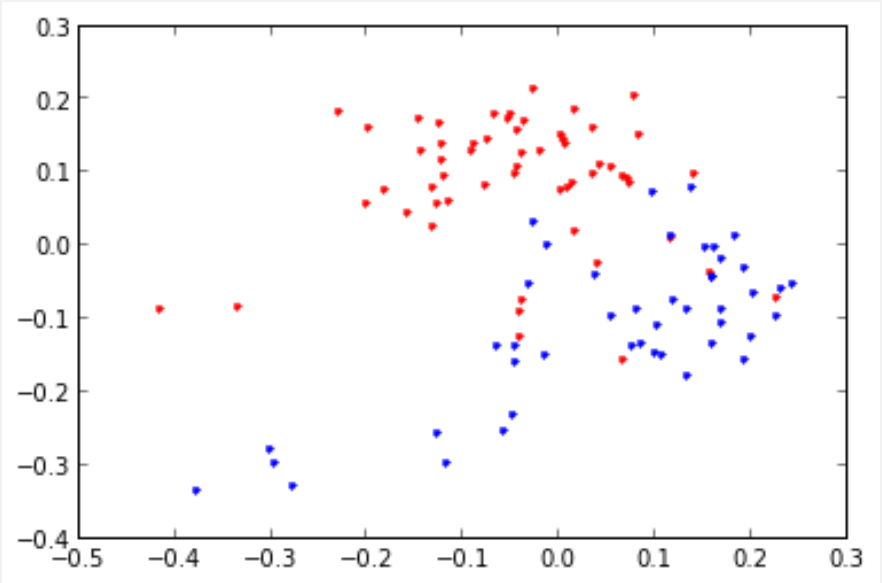
```
[<matplotlib.lines.Line2D at 0xb03bb2c>]
```



Simdi ayni veriyi bir de etiket bilgisini devreye sokarak cizdirelim. Sanat kirmizi muzik mavi olacak.

```
arts =proj[labels == 'art']
music =proj[labels == 'music']
plot(arts[:,0],arts[:,1],'r.')
plot(music[:,0],music[:,1],'b.')
```

```
[<matplotlib.lines.Line2D at 0xb58eecc>]
```



Goruldugu gibi veride ortaya cikan / ozvektorlerin kesfettigi dogal ayirim, hakikaten dogruymus.

Metotun ne yaptigina dikkat, bir suru boyutu bir kenara atmamiza ragmen geri kalan en onemli 2 boyut uzerinden net bir ayirim ortaya cikartabiliyoruz. Bu PCA yonteminin iyi bir is becerdigini gosteriyor, ve kelime sayilarinin makalelerin icerigi hakkinda ipucu icerdigini ispatliyor.

[1] Cosma Rohilla Shalizi, Advanced Data Analysis from an Elementary Point of View

[2] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2008T19

[3] http://www.stat.cmu.edu/∼cshalizi/490/pca/

[5] http://www.math.nyu.edu/faculty/goodman/teaching/RPME/notes/Section3.pdf

[6] Lineer Cebir notlarimizda SVD turetilmesine bakinca ozdeger/vektor mantigina atif yapildigini gorebiliriz ve akla su gelebilir; “ozdeger / vektor rutini isletmekten kurtulalim dedik, SVD yapiyoruz, ama onun icinde de ozdeger/vektor hesabi var”. Fakat sunu belirtmek gerekir ki SVD numerik hesabini yapmanin tek yontemi ozdeger/vektor yontemi degildir. Mesela Numpy Linalg kutuphanesi icindeki SVD, LAPACK dgesdd rutinini kullanir ve bu rutin ic kodlamasinda QR, ve bir tur bol / istila et (divide and conquer) algoritmasi isletmektedir.