

SVD ile PCA Hesaplamak

PCA bolumunde anlatilan yontem temel bileşenlerin hesabinda özdeğerler ve özvektörler kullandı. Alternatif bir yontem Tekil Değer Ayrıştırma (Singular Value Decomposition -SVD-) üzerinden bu hesabi yapmaktır. SVD için Lineer Cebir Ders 29'a bakabilirsiniz. Peki ne zaman klasik PCA ne zaman SVD üzerinden PCA kullanmalı? Bir cevap belki mevcut kutuphanelerde SVD kodlamasının daha iyi olması, ayrıştırmanın özvektör / değer hesabından daha hızlı işleyebilmesi [6].

Ayrıca birazdan göreceğimiz gibi SVD, kovaryans matrisi üzerinde değil, A 'nin kendisi üzerinde işletilir, bu hem kovaryans hesaplama aşamasını atlamamızı, hem de kovaryans hesabi sırasında ortaya çıkabilecek numerik puruzlerden korunmamızı sağlar (çok ufak değerlerin kovaryans hesabını bozabileceği literatürde bahsedilmektedir).

PCA ve SVD bağlantısına gelelim:

Biliyoruz ki SVD bir matrisi şu şekilde ayırır

$$A = USV^T$$

U matrisi $n \times n$ dikgen (orthogonal), V ise $m \times m$ dikgen. S 'in sadece köşegeni üzerinde değerler var ve bu σ_j değerleri A 'nin tekil değerleri (singular values) olarak biliniyor.

Şimdi A yerine AA^T koyalım, yani A 'nin kovaryans matrisinin SVD ayrıştırmasını yapalım, acaba elimize ne gelecek?

$$\begin{aligned} AA^T &= (USV^T)(USV^T)^T \\ &= (USV^T)(VS^T U^T) \\ &= USS^T U^T \end{aligned}$$

S bir köşegen matrisi, o zaman SS^T matrisi de köşegen, tek farkla köşegen üzerinde artık σ_j^2 değerleri var. Bu normal.

SS^T yerine Λ sembolünü kullanalım, ve denklemi iki taraftan (ve sağdan) U ile çarparsak (unutmayalım U ortanormal bir matris ve $U^T U = I$),

$$AA^T U = U \Lambda U^T U$$

$$AA^T U = U \Lambda$$

Son ifadeye yakından bakalım, U 'nun tek bir kolonuna, u_k diyelim, odaklanacak olursak, üstteki ifadede bu sadece kolona yönelik nasıl bir eşitlik çıkartabilirdik? Şöyle çıkartabilirdik,

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

Bu ifade tanıdık geliyor mu? Özdeğer / özvektör klasik yapısına eriştik. Üstteki eşitlik sadece ve sadece eğer u_k , AA^T 'nin özvektörü ve σ^2 onun özdeğeri ise geçerlidir. Bu eşitliği tüm U kolonları için uygulayabileceğimize göre demek ki U 'nun kolonlarında AA^T 'nin özvektörleri vardır, ve AA^T 'nin özdeğerleri A 'nin tekil değerlerinin karesidir.

Bu muthis bir buluş. Demek ki AA^T 'nin özvektörlerini hesaplamak için A üzerinde SVD uygulayarak U 'yu bulmamız yeterli, kovaryans matrisini hesaplamak bile gerekmiyor! AA^T özdeğerleri üzerinde büyüklük karşılaştırması için ise A 'nin tekil değerlerine 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 ilginç bir ornege bakalım. Bir araştırmacı belli yıllar arasındaki NY Times makalelerinde her yazıda hangi kelimenin kaç kere çıktığının verisini toplamış [1,2,3], bu veri 4000 kusur kelime, her satır (yazı) için bir boyut (kolon) olarak kaydedilmiştir. Bu veri nytimes.csv üzerinde ek bir normalize işleminden sonra, onun üzerinde boyut indirgeme yapabiliriz.

Veri setinde her yazı ayrıca ek olarak sanat (arts) ve müzik (music) olarak etiketlenmiş, ama biz PCA kullanarak bu etiketlere hiç bakmadan, verinin boyutlarını azaltarak acaba verinin “ayrılabilir” hale indirgenip indirgenemediğine bakalacağız. Sonra etiketleri veri üstüne koyup sonucun doğruluğunu kontrol edeceğiz.

Bakmak derken veriyi (en önemli) iki boyuta indirgeyip sonucu grafikleyeceğiz. İlla 2 olması gerekmez tabii, 10 boyuta indirgeyip (ki 4000 kusur boyuttan sonra bu hala muthis bir kazanım) geri kalanlar üzerinde mesela bir kümeleme algoritması kullanabilirdik.

Ana veriyi yükleyip birkaç satırını ve kolonlarını göstereyim.

```

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 kullanim DataFame 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 toplamının karekokunu aliriz ve satirdaki tum degerler bu karekok ile bolunur. Buna oklitsel (euclidian) normalizasyon denebilir.

Not: Oklitsel norm alirken toplamın hemen ardından cok ufak bir 1e-16 degeri eklememize dikkat cekelim, bunu toplamın sifir olma durumu icin yapıyoruz, ki sonra sifir la 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 atladi
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 yapalım

```
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])
```

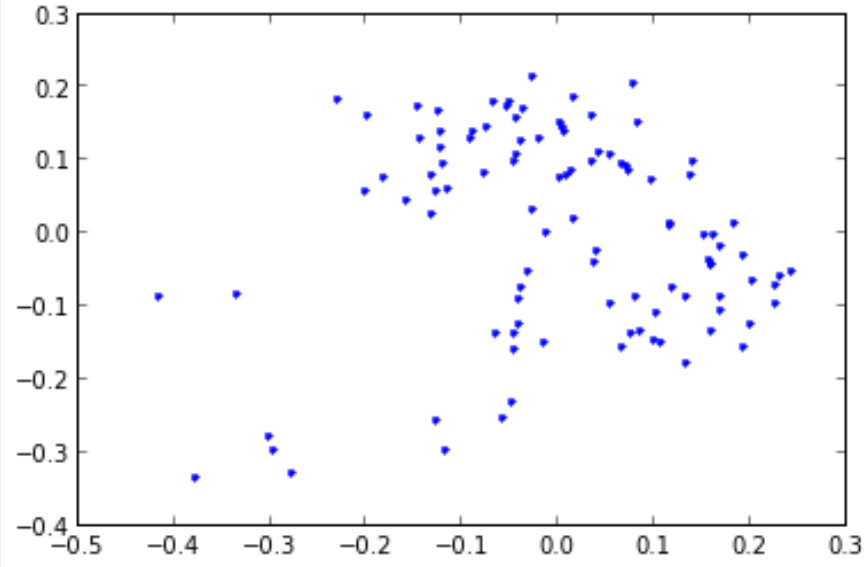
```
u.shape
```

```
(4430, 102)
```

SVD'nin verdiği 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],'.')
```

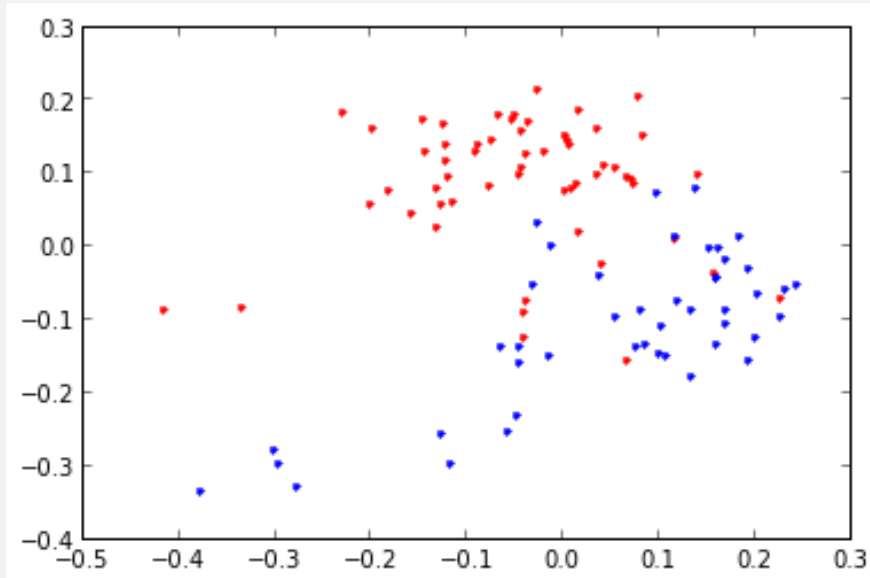
```
[<matplotlib.lines.Line2D at 0xb03bb2c>]
```



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 0xb58eccc>]



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 hakkında 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.