

[Download notebook](#)

Temel Bileşen Analizi (Principal Component Analysis -PCA-)

PCA yontemi boyut azaltan yontemlerden biridir, takip edilmeden (unsupervised) isleyebilir. Ana fikir veri noktalarının izdusumunun yapilacagi yonler bulmaktir ki bu yonler baglaminda noktalarin arasindaki degisinti / varyans (variance) en fazla olsun, ve ozellikle hangi degiskenlerin varyansinin daha fazla oldugu ortaya ciksın. Ornek olarak alttaki grafige bakalim,PCA yontemi boyut azaltan yontemlerden biridir, takip edilmeden (unsupervised) isleyebilir. Ana fikir veri noktalarının izdusumunun yapilacagi yonler bulmaktir ki bu yonler baglaminda noktalarin arasindaki degisinti / varyans (variance) en fazla olsun, yani noktalar en "yaygin" sekilde olsunlar. Boylece birbirinden daha uzaklasan noktalarin mesela daha rahat kumelenebilecegini umabiliriz.. Ayrica bir diger amac hangi degiskenlerin varyansinin daha fazla olmasinin anlasilmasiyla, o degiskenlerin daha faydali olabileceginin ortaya cikmasi. Ornek olarak alttaki grafige bakalim,

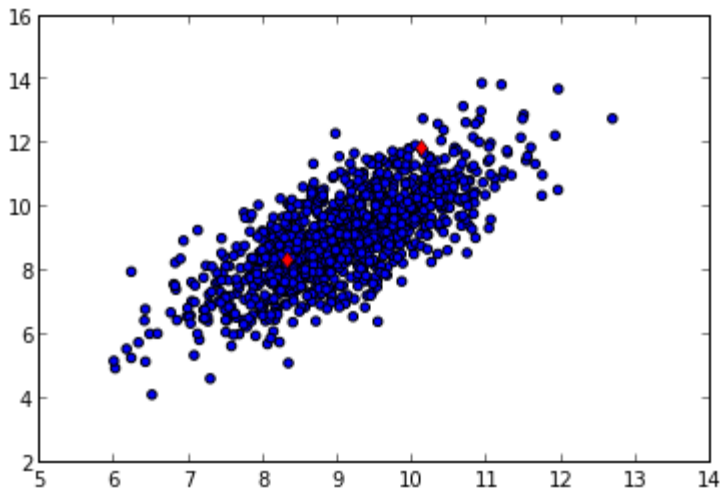
```
In [4]: from pandas import *
data = read_csv("testSet.txt", sep="\t", header=None)
data[:10]
```

Out[4]:

	0	1
0	10.235186	11.321997
1	10.122339	11.810993
2	9.190236	8.904943
3	9.306371	9.847394
4	8.330131	8.340352
5	10.152785	10.123532
6	10.408540	10.821986
7	9.003615	10.039206
8	9.534872	10.096991
9	9.498181	10.825446

```
In [47]: scatter(data.ix[:,0],data.ix[:,1])
          plot(data.ix[1,0],data.ix[1,1], 'rd')
          plot(data.ix[4,0],data.ix[4,1], 'rd')
```

```
Out[47]: [<matplotlib.lines.Line2D at 0xb52880c>]
```

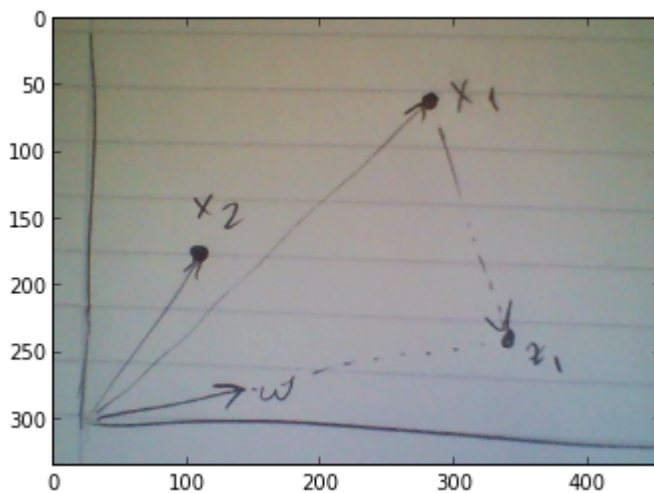


PCA ile yapmaya çalıştığımız öyle bir yön bulmak ki, x veri noktalarının tamamının o yöne izdüşümü yapılırca sonuç olacak, "izdüşümü yapılmış" z 'nin varyansı en büyük olsun. Bu bir maksimizasyon problemidir. Fakat ondan önce x nedir, z nedir bunlara yakından bakalım.

Veri x ile tüm veri noktaları kastedilir, fakat PCA probleminde genellikle bir "vektörün diğeri üzerine" yapılan izdüşümü, "daha optimal bir w yönü bulma", ve "o yöne doğru izdüşüm yapmak" kelimeleri kullanılır. Demek ki veri noktalarını bir vektör olarak görmeliyiz. Eğer üstte kırmızı ile işaretlenen iki noktayı alırsak (bu noktalar verideki 1. ve 4. sıradaki noktalar),

```
In [11]: img=imread("proj1.png")
          plt.imshow(img)
```

```
Out[11]: <matplotlib.image.AxesImage at 0x94cbf4c>
```



gibi bir görüntüden bahsediyoruz. Hayali bir w kullandık, ve x_1 veri noktası, w üzerine izdüşüm yapılarak z_1 'i ortaya çıkartıyor. Genel olarak ifade edersek, bir nokta için

$$z = w^T x$$

Yapmaya calistigimiz varyansi maksimize etmek demistik. Ozel bir noktayi referans alirsak, $Var(z_1)$ maksimize edilmeli. Bunu soyler soylemez x 'i ve z 'yi rasgele degiskenler olarak gordugumuzu belirtmis oluyoruz. Ardi ardina alet cantasindan temsili numaralari kullaniyoruz - x 'i bir yandan vektor yapiyoruz, bir yandan bir dagilimdan gelen zar atisi gibi goruyoruz, problemi cozmemize ne yardım edecekse onu surekli devreye sokuyoruz. Devam edelim, rasgele degisken deyince, demek ki x, z dagilimlardan gelecektir, bu dagilimlarin cok boyutlu normal (multivariate normal) oldugunu kabul edebiliriz. Peki eger $x, N(x, \Sigma)$ gibi cok boyutlu normal dagilimdan geliyorsa, $w^T x$ nedir?

Bu yeni "seyin" beklentisi ve varyansina bakalim,

$$E[w^T x] = w^T E[x] = w^T \mu$$

$$Var(w^T x) = w^T Var(x) w = w^T \Sigma w$$

Ustteki sonuclarin boyutlarına dikkat: $w^T \mu$ durumunda $1 \times N \cdot N \times 1 = 1 \times 1$, $w^T \Sigma w$ durumunda ise $1 \times N \cdot N \times N \cdot N \times 1 = 1 \times 1$. Iki durumda da tek boyutlu skalar degerler elde ettik. Demek ki w^T ile carpim sonrası bir $N(w^T \mu, w^T \Sigma w)$ dagilimi elde ederiz ve bu dagilim bir tek boyutlu bir dagilimdir. Yani w yonundeki izdusumm bize tek boyutlu bir Gaussian dagilimini verecektir. Bu sonuc aslında cok sasirtici olmasa gerek, tum veri noktalarını alip, baslangici orijin noktasında olan vektorlere cevrip ayni yone isaret edecek sekilde duzenliyoruz, bu vektorleri tekrar nokta olarak dusunursek, tabii ki ayni yonu gosteriyorlar, bilahere ayni çizgi üzerindeki noktalara donusuyorlar. Ayni çizgi üzerinde olmak ne demek? Tek boyuta inmis olmak demek.

Bastaki amacimiza donersek, $Var(z_1)$ 'i maksimize etmek ayni anda $Var(w_1^T \Sigma w_1)$ 'i maksimize etmek demektir.

Ufak bir sorun $w_1^T \Sigma w_1$ 'i surekli daha buyuk w_1 'lerle sonsuz kadar buyutebilirsiniz. Bize ek bir kisiltama sarti daha lazim, bu sart $\|w\| = 1$ olabilir, yani w 'nin norm'u 1'den daha buyuk olmasin. Boylece optimizasyon w 'nin buyuklugu üzerinde taklalar atmayacak, sadece yon bulmak ile ilgilenecek, iyi, zaten biz w 'nin yonu ile ilgileniyoruz. Aradigimiz ifadeyi yazalim, ve ek siniri Lagrange ifadesi olarak ekleyelim, ve yeni bir L ortaya cikartalim,

$$L(w_1, \lambda) = w_1^T \Sigma w_1 - \lambda(w_1^T w_1 - 1)$$

Niye eksiden sonraki terim o sekilde eklendi? O terim oyle sekilde secildi ki, $\partial L / \partial \lambda = 0$ alinca $w_1^T w_1 = 1$ geri gelsin / ortaya ciksın [2, sf 340], bu Lagrange'in dahice bulusu. Bunu kontrol edebilirsiniz, λ 'ya gore turev alirken w_1 sabit olarak yokolur, parantez icindeki ifadeler kalir ve sifira esitlenince orijinal kisiltama ifadesi geri gelir. Simdi

$$\max_{w_1} L(w_1, \lambda)$$

Turevi w_1 'e gore alirsak, ve sifira esitlersek,

$$2w_1 \Sigma - 2\lambda w_1 = 0$$

$$2w_1 \Sigma = 2\lambda w_1$$

$$\Sigma w_1 = \lambda w_1$$

Ustteki ifade ozdeger, ozvektor ana formulune benzemiyor mu? Evet. Eger w_1 , Σ 'nin ozvektoru ise ve esitligin sagindaki λ ona tekabul eden ozdeger ise, bu esitlik dogru olacaktir.

Peki hangi ozdeger / ozvektor maksimal degeri verir? Unutmayalim, maksimize etmeye calistigimiz sey $w_1^T \Sigma w_1$ idi

Eger $\Sigma w_1 = \lambda w_1$ yerine koyarsak

$$w_1^T \lambda w_1 = \lambda w_1^T w_1 = \lambda$$

Cunku $w_1^T w_1$ 'nin 1 olacagi sartini koymustuk. Neyse, maksimize etmeye calistigimiz deger λ cikti, o zaman en buyuk λ kullanirsak, en maksimal varyansi elde ederiz, bu da en buyuk ozdegerin ta kendisidir.

Demek ki izdusum yapılacak "yon" en buyuk ozdegere tekabul eden ozvektor olarak secilirse, temel bileşenlerden en

onemlisini hemen bulmus olacagiz.

Cebirin geri kalani w_2, w_3 icin devam eder, bu turetmenin detaylarini [1] ve [2] gibi kaynaklarda bulabilirsiniz. Fakat ulasilan sonuc en bileşenlerin onem sirasinin aynen ozdegerlerin buyukluk sirasina tekabul ediyor olmasi.

```
In [15]: means = mean(data, axis=0)
meanless_data = data - means
cov_mat = cov(meanless_data, rowvar=0)
eigs,eigv = linalg.eig(mat(cov_mat))
eig_ind = argsort(eigs)
eig_ind
```

```
Out[15]: array([0, 1])
```

Ustteki sonuc sort sonrasi elde edilen indis degerleri, sort en kucukten en buyuge dogru dizimi yapmistir, en buyuk ozdegerin indisi en sondadir. Raslantisal bir sekilde en buyuk olan deger en sagda cikti ama tersi de olabilirdi, neyse, bu ozdeger, vektorleri ekrana basalim,

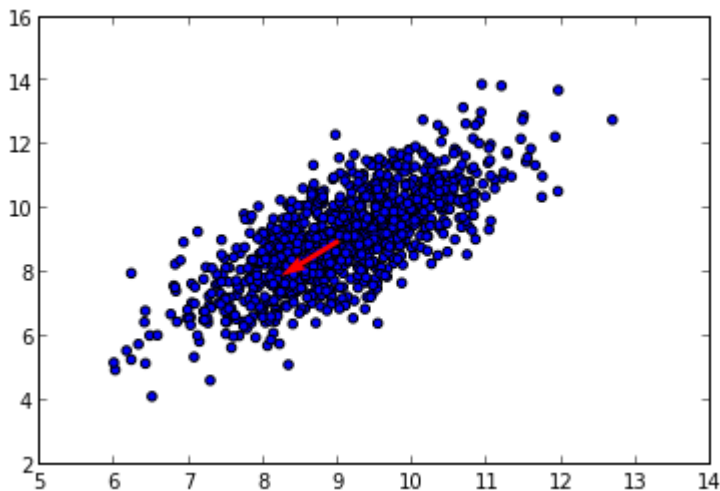
```
In [34]: print eigs[1],eigv[:,1].T
print eigs[0],eigv[:,0].T

2.89713495618 [[-0.52045195 -0.85389096]]
0.366513708669 [[-0.85389096  0.52045195]]
```

En buyuk olan yonu quiver komutunu kullanarak orijinal veri seti uzerinde gosterelim,

```
In [35]: scatter(data.ix[:,0],data.ix[:,1])
quiver(9,9,eigv[1,1],eigv[0,1],scale=10,color='r') # merkez 9,9, kabaca
secildi
```

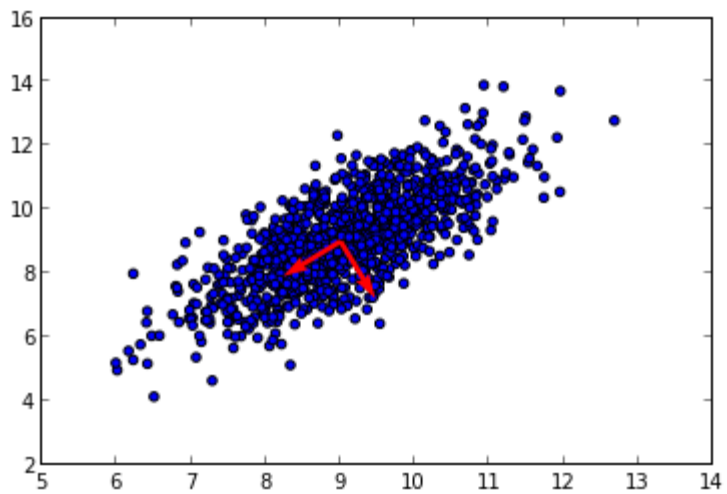
```
Out[35]: <matplotlib.quiver.Quiver at 0xbb6254c>
```



Goruldugu gibi bu yon hakikaten dagilimin, veri noktalarinin en cok yayilmis oldugu yon. Demek ki PCA yontemi dogru sonucu buldu. Her iki yonu de cizersk,

```
In [36]: scatter(data.ix[:,0],data.ix[:,1])
         quiver(9,9,eigv[1,0],eigv[0,0],scale=10,color='r')
         quiver(9,9,eigv[1,1],eigv[0,1],scale=10,color='r')
```

Out[36]: <matplotlib.quiver.Quiver at 0xc194c6c>



Bu ikinci yon birinciye dik olmaliydi, ve o da bulundu. Aslında iki boyut olunca baska secenek kalmiyor, 1. yon sonrasi ikincisi baska bir sey olamazdi, fakat cok daha yuksek boyutlarda en cok yayilimin oldugu ikinci yon de dogru sekilde geri getirilecekti.

[1] Alpaydin, E., Introduction to Machine Learning, 2nd Edition

[2] Strang, G., Linear Algebra and Its Applications, 4th Edition

[3] <http://www.stat.columbia.edu/~fwood/Teaching/w4315/Spring2010/PCA/slides.pdf>