Ozdegerler ve Ozvektorler ile Imaj Bolmek, Gruplamak

Bir imaji nasil gruplara ayirabiliriz? Ekran piksellerini çizit (graph) düğümleri olarak gösterebiliriz, ve bu ciziti yakinsallik (affinity) matrisine çevirebiliriz.

Bu matris üzerinde öyle işlemler yapalım ki, elimize Wx denen bir vektör geçsin; bu vektörün 1..N üyeleri, 1..N piksellerinin x gurubuna üyelik katsayısı olsun. Katılım değerleri en fazla olan vektör (gurup), ekran üzerindeki en büyük nesne demektir.

Matematiksel olarak şöyle bir formül kuralım. Basta sadece temsil etmeye ugraşıyoruz, bir gurup ve içinde olan pikseller arasında bağlantı, bir iliski kurmak istiyoruz. Piksel ve herhangi bir gurup arasındaki ilişkiyi formül ile kağıt üzerine dökmeyi amacliyoruz. Matematik, sayılar arasında alâka kurma sanatıdır. Elimizde şimdilik bir algoritma olmasa bile, temsili olarak bir iliski kurmak mümkün.

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j = \mathbf{x}^T\mathbf{A}\mathbf{x}$$

Bu formüle göre, Wij, cizit üzerinde gösterilen i ve j düğümu arasındaki bağlantı ağırlığı olacak. x vektörünün içindeki her değer, cizitteki düğümlerin bu x gurubuna dâhil olma katsayısı olacak. Formülun sol tarafına göre, bu tanımları her i ve j değeri için yaparak sonuçlarını toplamis oluyoruz.

Dikkat, toplam sonucu tek bir sayi, yani bir skalar. Nelerin birbiri ile carpildigi optimizasyon icin cok onemli, i ve j arasindaki agirligi, i'nin uyelik agirligi ve j'nin uyelik agirligi ile carpiyoruz, bunlari tum diger kombinasyonlar icin yapiyoruz, ama bu carpimlari topluyoruz. Carpim daha fazla buyutur, ve maksimizasyon icin bu buyukluk daha on planda olacaktir. Ve bu toplâmın 'en büyük' olduğu yer, görüntü üzerindeki en büyük nesnenin olduğu yerdir! Yâni elimizde bir matematiksel maksimizasyon problemi var.

Caprimi tekrar kontrol edelim

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Diyelim ki $i = 2$, $j = 1$. O zaman a_2, b_{21} ve c_1'in birbiriyle carpilmasi gerekir. Hakikaten caprimi elle kontrol edersek bunun oldugunu goruruz. Icinde $a_1 \cdot b_{21}$ caprimini iceren terim, sonra c_1 ile caprilacaktir.

Bir ek daha; maksimizasyon işlemine atlamadan once, bir matematiksel sınır daha koymaya mecburuz. Maksimizasyon problemlerinde her sayıyı muazzam büyüklüklere getirerek formül sonucunu sürekli büyütmek mümkün olabilirdi. Buna bir sınır getirmek için, sağ tarafta, A'nin yanına çarpan olarak x vektörünün norm'u (yani uzunluğu) 1 olsun diyrouz. Altta, bu tanımın açılmış hali var. (Not:

X vektörünün norm'u = X'in devriği çarpı X). Lagrange formulu soyle gosterilebilir:

$$w^T A w - \lambda(w^T w - 1)$$

$$w^T A w - \lambda(w^T w - 1) = 0$$

$$\frac{d}{dw} w^T A w - \lambda(w^T w - 1) = 0$$

$$2Aw - 2\lambda w = 0$$

$$2Aw = 2\lambda w$$

$$Aw = \lambda w$$

Ustteki son formul ozdeger (eigenvalue), ozvektorler (eigenvector) formuludur.

Rayleigh-Ritz kuramına göre, yukarıdaki formülün maksimum x vektörü A matrisinin maksimum özdeğerine tekabül eden özvektör olacaktir. Dügümler birbirine ne kadar iyi bağlıysa, bir gurubun içsel baglantısı ve 'gurupluğu' o kadar iyi olacaktir.

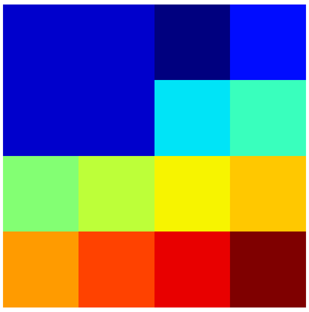
Bu son formül aşağıda

$$\lambda_{n-k} = \max_{x \perp x_{\lambda_n}, \ldots, x_{\lambda_{n-k+1}}} \mathbf{x}^T \mathbf{A} \mathbf{x}$$

Ornek

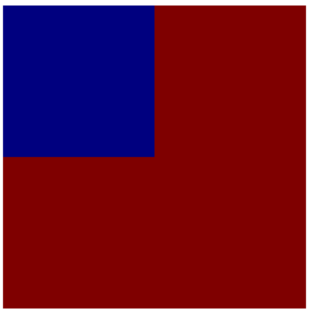
Once cok basit bir veri uzerinde gruplama yapalim, sol ust kosede ayni degerleri tasiyan bir resim yaratalim

```
Img = np.array([[3,3,0,6],
                [3,3,15,18],
                [22,25,28,30],
                [32,36,39,43]])

plt.imsave('simple.png',Img)
```



```
import itertools

def segment(Img, threshold):
    Img2 = Img.copy()
    n = Img2.shape[0]
    Img3 = Img2.flatten(order='F')
    nn = Img3.shape[0]

    f = lambda (i,j): np.exp(-((Img3[i]-Img3[j])**2))
    res=np.fromiter(itertools.imap(f, itertools.product(range(nn),range(nn))),
                    dtype=np.float)

    beta = 1.
    Img3 = np.exp(-beta * res / np.std(res))

    Img3 = Img3.reshape((nn,nn))
    print "Img3",Img3.shape

    V,D = np.linalg.eig(Img3)
    V = np.real(V)
    a = np.real(D[0])

    a = np.reshape(a, (n,n))
    np.savetxt('/tmp/out.txt',a)
    Img2[a<threshold] = 255. # mavi
    np.savetxt('/tmp/out2.txt',Img2)
    plt.imsave('eigseg1.png',Img2)

segment(Img, threshold = 1e-1)
```

```
Img3 (16, 16)
```



Kodda `imread` ile imaji okuduk, elimize 4x4 boyutunda bir matris gecti. Bu matrisi once "duzlestirerek" bir vektor haline getirdik, ki bu vektorun elemanlari yeni bir yakinlilik matrisinin kenarlari olacakti. Sonra bu yeni elemanlarin her birini bir digeri ile karsilastirark yakinligini hesapladik, bunu piksel degerinin ne kadar yakin olduguna bakarak karar verdik, `exp` bunun icin kullanildi. Ayrica yakinlik ve uzaklik kavramini tersine cevrildi, `exp` icinde eksi olmasi bundan, birbirine "benzer" yani degerleri birbirine yakin olan piksellerin farklari az olacaktir, fakat biz bu azligi maksimizasyon problemi icin bir fazlaliga cevirmek istiyoruz.

Bu noktada A matrisinin ozdegerlerini hesaplattik, ve geriye `C,D` geri geldi. Numpy ozdegerleri ve ona tekabul eden ozvektorleri buyukluk sirasina dizerek geri getirir, bu sayede sifirinci (ilk) `D`'ye bakarak en buyuk ozdegere tekabul eden ozvektoru alabildik.

Bu vektorun degerleri ise uyeligi en yuksek olan grubu iceriyordu. Ciplak gozle bakinca bu degerlerin hangi esik (threshold) degerini almak gerektigini gorduk. Esigin altinda kalan degerleri grup disi olarak kabul ettik ve o degerlerin kordinatina 255 piksel degerini atadik, ki ustteki kirmizi renkli gozuken pikseller bu "kume disi" degerleri temsil ediyor.

Not: Ustteki kodlama performans olarak biraz yavas olabilir, alternatif olarak sadece birbirine yakin pikseller arasi ilinti hesaplanarak ortaya cikacak seyrek (sparse) matris uzerinde seyrek ozvektor hesabi daha hizli sonuc verebilir.

Kaynaklar

[1] Sarkar ve Boyer makalesi "Değisimlerin Sayısal Ölçümünü Özellik Organizasyonu Kullanarak Yapmak: Özdeğerler ve Özvektörler". (Quantitative Measures of Change Based on Feature Organization: EigenValues and EigenVectors)

[2] Forsyth ve Ponce kitabı "Bilgisayar Görüşü, Yeni Yaklaşım (Computer Vision, A Modern Approach)