

Ozdegerler ve Ozvektorler ile Imaj Bolmek, Gruplamak

Bir imaji nasıl gruplara ayırabiliriz? Ekran piksellerini çizit (graph) düğümleri olarak gösterebiliriz, ve bu çiziti yakınsallık (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 uğraşıyoruz, bir gurup ve içinde olan pikseller arasında bağlantı, bir ilişki kurmak istiyoruz. Piksel ve herhangi bir gurup arasındaki ilişkiyi formül ile kağıt üzerine dökmeyi amaçlıyoruz. Matematik, sayılar arasında alâka kurma sanatıdır. Elimizde şimdilik bir algoritma olmasa bile, temsili olarak bir ilişki 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, W_{ij} , çizit üzerinde gösterilen i ve j düğümü arasındaki bağlantı ağırlığı olacak. x vektörünün içindeki her değer, çizitteki düğümlerin bu x gurubuna dâhil olma katsayısı olacak. Formülün sol tarafına göre, bu tanımları her i ve j değeri için yaparak sonuçlarını toplamış oluyoruz.

Dikkat, toplam sonucu tek bir sayı, 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 yapıyoruz, 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 önce, bir matematiksel sınır daha koymaya mecburuz. Maksimizasyon problemlerinde her sayıyı muazzam büyüklüklere getirerek formül sonucunu sürekli büyötmek 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:

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

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

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

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

$$2Aw = 2\lambda w$$

$$Aw = \lambda w$$

Ustteki son formül özdeğer (eigenvalue), özvektörler (eigenvector) formülüdür.

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

Bu son formül aşağıda

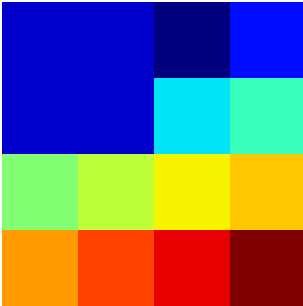
$$\lambda_{n-k} = \max_{x \perp x_{\lambda_n}, \dots, x_{\lambda_{n-k+1}}} \mathbf{x}^T \mathbf{A} \mathbf{x}$$

Ornek

Önce çok basit bir veri üzerinde gruplama yapalım, sol üst köşede aynı değerleri taşıyan bir resim yaratalım

```
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

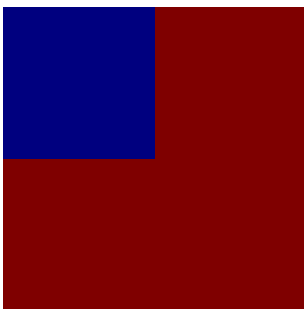
    V,D = np.linalg.eig(Img3)
    V = np.real(V)
    a = np.real(D[0])

    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 “düzlestirerek” bir vektor haline getirdik, ki bu vektorun elemanlari yeni bir yakınlilik matrisinin kenarlari olacakti. Sonra bu yeni elemanların her birini bir digeri ile karsilastirarak yakınliginı hesapladık, bunu piksel degerinin ne kadar yakınlık olduğuna bakarak karar verdik, `exp` bunun için kullanildi. Ayrıca yakınlık ve uzaklık kavramını tersine çevrildi, `exp` içinde eksi olması bundan, birbirine ”benzer” yani

değerleri birbirine yakın olan piksellerin farkları az olacaktır, fakat biz bu azlığı maksimizasyon problemi için bir fazlalığa çevirmek istiyoruz.

Bu noktada A matrisinin özdeğerlerini hesaplattık, ve geriye C, D geri geldi. Numpy özdeğerleri ve ona tekabül eden özvektörleri büyüklük sırasına dizerek geri getirir, bu sayede sıfıncı (ilk) D'ye bakarak en büyük özdeğere tekabül eden özvektörü alabildik.

Bu vektörün değerleri ise uyeliği en yüksek olan grubu içeriyordu. Ciplak gözle bakınca bu değerlerin hangi eşik (threshold) değerini almak gerektiğini gördük. Eşik altında kalan değerleri grup dışı olarak kabul ettik ve o değerlerin koordinatına 255 piksel değerini atadık, ki üstteki kırmızı renkli gözüken pikseller bu “küme dışı” değerleri temsil ediyor.

Not: Üstteki kodlama performans olarak biraz yavaş olabilir, alternatif olarak sadece birbirine yakın pikseller arası ilinti hesaplanarak ortaya çıkacak seyrek (sparse) matris üzerinde seyrek özvektör hesabi daha hızlı sonuç verebilir.

Kaynaklar

[1] Sarkar ve Boyer makalesi ”Değişimlerin 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)