

Boltzman Makinalari (Rasgele Hopfield Aglari)

Alttaki ifade bir Boltmann dagilimini gosterir,

$$P(x; W) = \frac{1}{Z(W)} \exp \left[ \frac{1}{2} x^T W x \right] \quad (3)$$

ki  $x$  cok boyutlu ve  $-1, +1$  degerleri iceren bir vektor,  $W$  simetrik ve caprazinda (diagonal) sifir iceren bir matristir,  $n \times d$  boyutlarindaki bir veri icin  $d \times d$  boyutlarinda olacaktir. Boltzmann Makinalari (BM), Kisitli Boltzmann Makinalari (Restricted Boltzmann Machines) kavramina gecis yapmadan once iyi bir durak nok-tasi.

BM  $W$  icinde aslinda tum degiskenlerin ikisel iliskisini icerir.  $W$  cok degiskenli Gaussian dagilimindaki  $\Sigma'$ 'da oldugu gibi ikisel baglantilari saptar. Veriden  $W$ 'yu ogrenmek icin olurlugu hesaplamak lazim. Olurluk (likelihood)

$$\prod_{n=1}^N P(x^{(n)}; W) = \frac{1}{Z(W)} \exp \left[ \frac{1}{2} x^{(n)T} W x^{(n)} \right]$$

Log olurluk

$$\mathcal{L} = \ln \left( \prod_{n=1}^N P(x^{(n)}; W) \right) = \sum_{n=1}^N \left[ \frac{1}{2} x^{(n)T} W x^{(n)} - \ln Z(W) \right] \quad (1)$$

Birazdan  $\frac{\partial \mathcal{L}}{\partial w_{ij}}$  turevini alacagiz, o sirada  $\ln Z(W)$ 'nin turevi lazim, daha dogrusu  $Z(W)$ 'yi nasil turevi alinir hale getiririz?

$Z(W)$  normalizasyon sabiti olduguna gore, dagilimin geri kalaninin sonsuzlar uzerinden entegrali (ya da toplami) normalizasyon sabitine esittir,

$$Z(W) = \sum_x \exp \left[ \frac{1}{2} x^T W x \right]$$

$$\ln Z(W) = \ln \left[ \sum_x \exp \left( \frac{1}{2} x^T W x \right) \right]$$

Log bazli turev alinca log icindeki hersey oldugu gibi bolume gider, ve log icin-dekinin turevi alinirak bolume koyulur. Fakat log icine dikkatli bakarsak bu za-ten  $Z(W)$ 'nin tanimidir, boylece denklemi temizleme sansi dogdu, bolume hemen  $Z(W)$  deriz, ve turevi log'un icine uygulariz,

$$\frac{\partial}{\partial w_{ij}} \ln Z(W) = \frac{1}{Z(W)} \left[ \sum_x \frac{\partial}{\partial w_{ij}} \exp \left( \frac{1}{2} x^T W x \right) \right]$$

$$\frac{\partial}{\partial w_{ij}} \exp\left(\frac{1}{2} \mathbf{x}^T \mathbf{W} \mathbf{x}\right) = \frac{1}{2} \exp\left(\frac{1}{2} \mathbf{x}^T \mathbf{W} \mathbf{x}\right) \frac{\partial}{\partial w_{ij}} \mathbf{x}^T \mathbf{W} \mathbf{x} \quad (2)$$

(2)'in icindeki bolumu acalim,

$$\frac{\partial}{\partial w_{ij}} \mathbf{x}^T \mathbf{W} \mathbf{x} = x_i x_j$$

Simdi (2)'ye geri koyalim,

$$\begin{aligned} &= \frac{1}{2} \exp\left(\frac{1}{2} \mathbf{x}^T \mathbf{W} \mathbf{x}\right) x_i x_j \\ \frac{\partial}{\partial w_{ij}} \ln Z(\mathbf{W}) &= \frac{1}{Z(\mathbf{W})} \left[ \sum_{\mathbf{x}} \frac{1}{2} \exp\left(\frac{1}{2} \mathbf{x}^T \mathbf{W} \mathbf{x}\right) x_i x_j \right] \\ &= \frac{1}{2} \sum_{\mathbf{x}} \frac{1}{Z(\mathbf{W})} \exp\left(\frac{1}{2} \mathbf{x}^T \mathbf{W} \mathbf{x}\right) x_i x_j \\ &= \frac{1}{2} \sum_{\mathbf{x}} P(\mathbf{x}; \mathbf{W}) x_i x_j \end{aligned}$$

Ustteki son ifadede bir kisaltma kullanalim,

$$\sum_{\mathbf{x}} P(\mathbf{x}; \mathbf{W}) x_i x_j = \langle x_i, x_j \rangle_{P(\mathbf{x}; \mathbf{W})} \quad (4)$$

Artik  $\ln Z(\mathbf{W})$ 'nin turevini biliyoruz. O zaman tum log olurlugun turevine (1) donebiliriz,

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_{ij}} &= \sum_{n=1}^N \left[ \frac{\partial}{\partial w_{ij}} \frac{1}{2} \mathbf{x}^{(n)T} \mathbf{W} \mathbf{x}^{(n)} - \frac{\partial}{\partial w_{ij}} \ln Z(\mathbf{W}) \right] \\ &= \sum_{n=1}^N \left[ \frac{1}{2} x_i^{(n)} x_j^{(n)} - \frac{\partial}{\partial w_{ij}} \ln Z(\mathbf{W}) \right] \\ &= \sum_{n=1}^N \left[ \frac{1}{2} x_i^{(n)} x_j^{(n)} - \frac{1}{2} \langle x_i x_j \rangle_{P(\mathbf{x}; \mathbf{W})} \right] \end{aligned}$$

1/2 sabitlerini atalim,

$$= \sum_{n=1}^N \left[ \mathbf{x}_i^{(n)\top} \mathbf{x}_j^{(n)} - \langle \mathbf{x}_i \mathbf{x}_j \rangle_{P(\mathbf{x}; W)} \right]$$

Eger

$$\langle \mathbf{x}_i \mathbf{x}_j \rangle_{\text{Data}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_i^{(n)\top} \mathbf{x}_j^{(n)}$$

olarak alırsak, esitliğin sağ tarafı verisel kovaryansı (empirical covariance) temsil eder. Düzenleyince,

$$N \cdot \langle \mathbf{x}_i \mathbf{x}_j \rangle_{\text{Data}} = \sum_{n=1}^N \mathbf{x}_i^{(n)\top} \mathbf{x}_j^{(n)}$$

şimdi esitliğin sağ tarafı uc üstteki formüle geri koyulabilir,

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = N \left[ \langle \mathbf{x}_i \mathbf{x}_j \rangle_{\text{Data}} - \langle \mathbf{x}_i \mathbf{x}_j \rangle_{P(\mathbf{x}; W)} \right]$$

Her ne kadar  $N$  veri noktası sayısını gösteriyor olsa da, üstteki ifade bir gradyan güncelleme formülü olarak ta görülebilir, ve  $N$  yerine bir güncelleme sabiti alınabilir. Gradyan güncelleme olarak görülebilir çünkü  $w_{ij}$ 'ye göre türev aldık, o zaman bizi  $\mathcal{L}$ 'in minimumuna götürecek  $w$  adımları üstte görüldüğü gibidir.

(4)'te görülen  $\langle \mathbf{x}_i \mathbf{x}_j \rangle_{P(\mathbf{x}; W)}$ 'in anlamı nedir? Bu ifade mümkün tüm  $\mathbf{x}$  değerleri üzerinden alınıyor ve ikisel ilişkilerin olasılığını “mevcut modele” göre hesaplıyor. Yani bu ifade de bir korelasyon hesabıdır, sadece veriye göre değil, tüm mümkün değerler ve model üzerinden alınır. Bu hesabi yapmak oldukça zordur, fakat yaklaşıksal olarak Monte Carlo yöntemi ile hesaplanabilir. Nihayet MC ve MCMC metodlarının kullanılma sebebini görmeye başlıyoruz; bu metodlar zaten asiri yüksek boyutlu, analitik çözümü olmayan, hesaplanamaz (intractable) entegraller (ya da toplamlar) için keşfedilmiştir.

Yani bu ifadeyi hesaplamak için Monte Carlo simülasyonu kullanacağız. Tüm değerleri teker teker ziyaret etmek yerine (ki bu çok uzun zaman alırdı) mevcut modele en olası  $\mathbf{x}$  değerleri “ürettireceğiz”, ve bu değerleri alıp sanki gerçek veriymiş gibi sayısal korelasyonlarını hesaplayacağız. Eğer veriler dağılımın en olası noktalarından geliyorsa, elimizde veri dağılımı “iyi” temsil eden bir veri setidir. Daha sonra bu korelasyon hesabını değeri gerçek veri korelasyonunun dan çıkartıp bir sabit üzerinden gradyan adımı atmamız mümkün olacak.

Gibbs Orneklemesi (Sampling)

Gibbs orneklemesinin detayları için *Monte Carlo, Entegraller, MCMC* yazısına danışılabilir. Boltzmann dağılımından örneklem almak için bize tek bir değişken (hucre)

haricinde diger hepsinin bilindigi durumun olasilik hesabi lazim, yani kosulsal olasilik  $P(x_i = 1|x_j, j \neq i)$ . Yani  $x$  uzerinde, biri haric tum ogelerin bilindigi durumda bilinmeyen tek hucre  $i$ 'nin 1 olma olasilik degeri,

$$P(x_i = 1|x_j, j \neq i) = \frac{1}{1 + e^{-\alpha_i}}$$

ve,

$$\alpha_i = \sum_j w_{ij}x_j$$

Bu kosulsal olasiligin temiz / basit bir formul olmasi onemli, ustteki gorulen bir sigmoid fonksiyonu bu turden bir fonksiyondur... Bu fonksiyonlar hakkında daha fazla bilgi *Lojistik Regresyon* yazisinda bulunabilir.

Ama, ana formul (3)'ten bu noktaya nasil eristik? Bu noktada biraz turetme yapmak lazim.  $x$  vektörü icinde sadece  $x_i$  ogesinin 1 olmasini  $x^b$  olarak alalim. Once kosulsal dagilimda "verili" olan kısmi elde etmek lazim. O uzaman

$$P(x_j, j \neq i) = P(x^0) + P(x^1)$$

Bu bir marjinalizasyon ifadesi, tum olasi  $i$  degerleri uzerinde bir toplam alinca geri kalan  $j$  degerlerinin dagilimini elde etmis oluruz.

$$P(x_i = 1|x_j, j \neq i) = \frac{P(x^1)}{P(x^0) + P(x^1)}$$

cunku  $P(A|B) = P(A, B)/P(B)$  bilindigi gibi, ve  $P(x^1)$  icinde  $x_1 = 1$  setini iceren tum veriler uzerinden.

Esitligin sag tarafinda  $P(x^1)$ 'i bolen olarak gormek daha iyi, ayrıca ulasmak istedigimiz  $1/(1 + e^{-\alpha_i})$  ifadesinde  $+1$ 'den kurtulmak iyi olur, boylece sadece  $e^{-\alpha_i}$  olan esitligi ispatlariz. Bunun her iki denklemde ters ceviri 1 cikartabiliriz,

$$\begin{aligned} 1/P(x_i = 1|x_j, j \neq i) &= \frac{P(x^0) + P(x^1)}{P(x^1)} \\ &= 1 + \frac{P(x^0)}{P(x^1)} \end{aligned}$$

Bir cikartirsak,  $\frac{P(x^0)}{P(x^1)}$  kalir. Bu bize ulasmak istedigimiz denklemde  $e^{-\alpha_i}$  ibaresini birakir. Artik sadece  $\frac{P(x^0)}{P(x^1)}$ 'in  $e^{-\alpha_i}$ 'e esit oldugunu gostermek yeterli.

$$\frac{P(x^0)}{P(x^1)} = \exp(x^{0T} W x^0 - x^{1T} W x^1)$$

Simdi  $x^T W x$  gibi bir ifadeyi indisler bazinda acmak icin sunlari yapalim,

$$x^T W x = \sum_{k,j} x_k x_j w_{kj}$$

Ustteki cok iyi bilinen bir acilim. Eger

$$\sum_{k,j} \underbrace{x_k x_j w_{kj}}_{Y_{kj}} = \sum_{k,j} Y_{kj}$$

alirsak birazdan yapacagimiz islemler daha iyi gorulebilir. Mesela  $k = i$  olan durumu dis toplamdan disari cekebiliriz

$$= \sum_{k \neq i} \sum_j Y_{kj} + \sum_j Y_{ij}$$

Daha sonra  $j = i$  olan durumu ic toplamdan disari cekebiliriz,

$$= \sum_{k \neq i} \left( \sum_{j \neq i} Y_{kj} + Y_{ki} \right) + \sum_j Y_{ij}$$

Ic dis toplamlari birlestirelim,

$$\begin{aligned} &= \sum_{k \neq i, j \neq i} Y_{kj} + \sum_{k \neq i} Y_{ki} + \sum_j Y_{ij} \\ &= \sum_{k \neq i, j \neq i} Y_{kj} + \sum_k Y_{ki} + \sum_j Y_{ij} + Y_{ii} \end{aligned}$$

Ustteki ifadeyi  $\exp(x^{0T} W x^0 - x^{1T} W x^1)$  icin kullanirsak,

$$\exp \left( \sum_k Y_{ki}^0 + \sum_j Y_{ij}^0 + Y_{ii}^0 - \left( \sum_k Y_{ki}^1 + \sum_j Y_{ij}^1 + Y_{ii}^1 \right) \right)$$

$\sum_{k \neq i, j \neq i} Y_{kj}$  teriminin nereye gittigi merak edilirse, bu ifade i'ye dayanmadigi icin bir eksi bir arti olarak iki defa dahil edilip iptal olacakti.

$$= \exp \left( 0 - \left( \sum_k Y_{ki}^1 + \sum_j Y_{ij}^1 + Y_{ii}^1 \right) \right)$$

$W$ 'nin simetrik matris oldugunu dusunursek,  $\sum_k Y_{ki}^1$  ile  $\sum_j Y_{ij}^1$  ayni ifadedir,

$$= \exp \left( - \left( 2 \sum_j Y_{ij}^1 + Y_{ii}^1 \right) \right)$$

$W$  sifir caprazli bir matristir, o zaman  $Y_{ii}^1 = 0$ ,

$$= \exp \left( 2 \sum_j Y_{ij}^1 \right) = \exp(-2a_i)$$

Orijinal dagilim denkleminde  $1/2$  ifadesi vardi, onu basta islemlere dahil etmemistik, edilseydi sonuc  $\exp(-a_i)$  olacakti.

```
import numpy as np
```

```
class Boltzmann:
```

```
    def __init__(self, n_iter=100, eta=0.1, sample_size=100, init_sample_size=10):
        self.n_iter = n_iter
        self.eta = eta
        self.sample_size = sample_size
        self.init_sample_size = init_sample_size
```

```
    def sigmoid(self, u):
        return 1. / (1. + np.exp(-u));
```

```
    def draw(self, Sin, T):
        """
        Bir Gibbs gecisi yaparak dagilimdan bir orneklem al
        """
        D=Sin.shape[0]
        S=Sin.copy()
        rand = np.random.rand(D,1)
        for i in xrange(D):
            h=np.dot(T[i,:],S)
            S[i]=rand[i]<self.sigmoid(h);
        return S
```

```
    def sample(self, T):
        N=T.shape[0]
        # sigmoid(0) her zaman 0.5 olacak
        s=np.random.rand(N)<self.sigmoid(0)
        # alttaki dongu atlama / gozonune alinmayacak degerler icin
        for k in xrange(self.init_sample_size):
            s=self.draw(s,T)
        S=np.zeros((N,self.sample_size))
        S[:,0]=s
        # simdi degerleri toplamaya basla
        for i in xrange(1,self.sample_size):
            S[:,i]=self.draw(S[:,i-1],T)
        return S.T
```

```

def normc(self, X):
    """
    normalizasyon sabitini dondur
    """
    def f(x): return np.exp(0.5 * np.dot(np.dot(x, self.W), x))
    S = 2*self.sample(self.W)-1
    # sozluk icinde anahtar tekil x degeri boylece bir
    # olasilik degeri sadece bir kere toplanir
    res = dict((tuple(s), f(s)) for s in S)
    return np.sum(res.values())

def fit(self, X):
    W=np.zeros((X.shape[1],X.shape[1]))
    W_data=np.dot(X.T,X)/X.shape[1];
    for i in range(self.n_iter):
        if i % 10 == 0: print 'Iteration', i
        S = self.sample(W)
        S = (S*2)-1
        W_guess=np.dot(S.T,S)/S.shape[1];
        W += self.eta * (W_data - W_guess)
        np.fill_diagonal(W, 0)
    self.W = W
    self.C = self.normc(X)

def predict_proba(self, X):
    return np.diag(np.exp(0.5 * np.dot(np.dot(X, self.W), X.T))) / self.C

```

Fonksiyon `draw` icinde, tek bir veri satiri icin ve sirayla her degisken (hucre) icin, diger degiskenleri baz alip digerinin kosulsal olasiligini hesapliyoruz, ve sonra bu olasiligi kullanarak bir sayi uretimi yapiyoruz. Uretimin yapılması için `np.random.rand`'dan gelen 0 ve 1 arasındaki bir (duz) uniform rasgele sayiyi gecip gecmeme irdelemesi yeterli. Bir Bernoulli olasilik hesabini uretilen bir rasgele degiskene bu sekilde cevirebilirsiniz. Bu niye isler? Ustte belirttigimiz irdelemeyi rasgele degisken olarak kodlarsak (ki bu da bir Bernoulli rasgele degiskeni olur), ve uniform rasgele degisken  $U$  olsun,

$$Y = \begin{cases} 1 & U < p \\ 0 & U \geq p \end{cases}$$

Bu durumda  $P(X = 1) = P(U < p) = p$  olurdu. Neden? Cunku ustte bir surekli (continuous) bir uniform yarattik, ve  $P(U < p) = F_u(p) = p$ .

Devam edelim; Cagri `sample` ise `draw`'u kullanarak pek cok veri satirini iceren ve dagilimi temsil eden bir orneklem yaratmakla sorumlu. Bunu her orneklem satirini baz alarak bir sonrakini urettirerek yapiyor, boylelikle MCMC'nin dagilimi "gezmesi" saglanmis oluyor.

### Normalizasyon Sabiti

Bu sabitin hesaplanması için aynen  $\langle x_i x_j \rangle_{p(x;W)}$  için olduğu gibi tüm mümkün  $x$ 'ler üzerinden bir toplam gerekir. Bu toplamın hesaplanmasının çok zor olduğu

icin, yine MCMC'ye basvuracagiz. Tek fark alinan orneklemi (3) formulune gecegiz, ve bir olasilik hesabi yapacagiz, ve bu olasiliklar toplayacagiz. Tabii ayni  $x$ 'i (eger tekrar tekrar uretilirse -ufak bir ihtimal ama mumkun-) tekrar tekrar toplamamak icin hangi  $x$ 'lerin uretildigini bir sozluk icinde hatirlayacagiz, yani bir  $x$  olasiligi sadece bir kere toplanacak.

Simdi ufak bir ornek uzerinde BM'i isletelim.

```
import boltz
A = np.array([\
[0.,1.,1.,1],\
[1.,0.,0,0],\
[1.,1.,1.,0],\
[0, 1.,1.,1.],\
[1, 0, 1.,0]\
])
A[A==0]==-1

clf = boltz.Boltzmann(n_iter=50,eta=0.01,sample_size=200,init_sample_size=50)
clf.fit(A)
print 'W'
print clf.W
print 'normalizasyon sabiti', clf.C

Iteration 0
Iteration 10
Iteration 20
Iteration 30
Iteration 40
W
[[ 0.      -0.065 -0.06  -0.055]
 [-0.065   0.      0.17   0.105]
 [-0.06    0.17    0.     -0.09 ]
 [-0.055   0.105 -0.09    0.     ]]
normalizasyon sabiti 16.4620358997
```

Sonuc  $W$  ustte goruldugu gibi. Ornek veriye bakarsak 2. satir 3. kolonda arti bir deger var, 1. satir 4. kolonda eksi deger var. Bu bekledigimiz bir sey cunku 2. ve 3. degiskenlerin arasinda bir korelasyon var,  $x_2$  ne zaman 1/0 ise  $x_3$  te 1/0. Fakat  $x_1$  ile  $x_4$  ters bir korelasyon var, birbirlerinin zitti degerlere sahipler.

Simdi yeni test verisini dagilima “soralim”,

```
test = np.array([\
[0.,1.,1.,1],\
[1.,1.,0,0],\
[0.,1.,1.,1]\
])
print clf.predict_proba(test)

[ 0.0730905  0.05692294  0.0730905 ]
```

Goruntu Tanimi

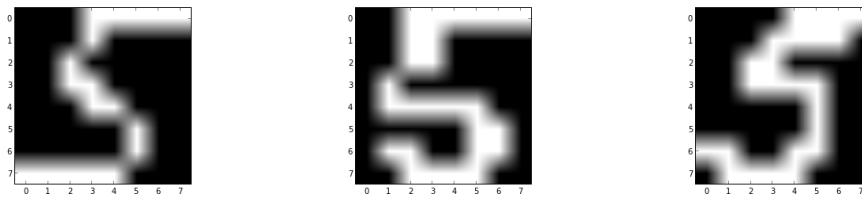


Elimizde el yazisi tanima algoritmaları için kullanılan bir veri seti var. Veride 0,5,7 harflerinin görüntüleri var. Mesela 5 için bazı örnek görüntüler,

```
Y = np.loadtxt('../stat/stat_mixbern/binarydigits.txt')
label = np.ravel(np.loadtxt('../stat/stat_mixbern/bindigitlabels.txt'))
Y5 = Y[label==5]
plt.imshow(Y5[0,:].reshape((8,8),order='C'), cmap=plt.cm.gray)
plt.savefig('boltzmann_01.png')

plt.imshow(Y5[1,:].reshape((8,8),order='C'), cmap=plt.cm.gray)
plt.savefig('boltzmann_02.png')

plt.imshow(Y5[2,:].reshape((8,8),order='C'), cmap=plt.cm.gray)
plt.savefig('boltzmann_03.png')
```



Bu görüntüleri tanımak için BM kullanalım. Eğitim ve test olarak veriyi ikiye ayıracağız, ve eğitim seti her etiket için  $W$ 'sini öğrenmek için kullanılacak. Daha sonra test setinde her veri noktalarını her üç BM'ye ayrı ayrı "sorup" o test verisinin o BM'ye göre olasılığını alacağız, ve hangi BM daha yüksek olasılık donduruyorsa etiket olarak onu kabul edeceğiz. Hangi BM daha yüksek olasılık donduruyorsa, o BM "bu verinin benden gelme olasılığı yüksek" diyor demektir, ve etiket o olacaktır.

```
from sklearn import neighbors
import numpy as np, boltz
from sklearn.cross_validation import train_test_split

Y = np.loadtxt('../stat/stat_mixbern/binarydigits.txt')
labels = np.ravel(np.loadtxt('../stat/stat_mixbern/bindigitlabels.txt'))
X_train, X_test, y_train, y_test = train_test_split(Y, labels, test_size=0.4, random_state=0)
X_train[X_train==0]=-1
X_test[X_test==0]=-1

clfs = {}
for label in [0,5,7]:
    x = X_train[y_train==label]
    clf = boltz.Boltzmann(n_iter=30,eta=0.05,sample_size=500,init_sample_size=100)
    clf.fit(x)
    clfs[label] = clf

res = []
for label in [0,5,7]:
    res.append(clfs[label].predict_proba(X_test))

res3 = np.argmax(np.array(res).T,axis=1)
```

```
res3[res3==1] = 5
res3[res3==2] = 7
print 'Boltzmann Makinasi', np.sum(res3==y_test) / float(len(y_test))
```

```
clf = neighbors.KNeighborsClassifier()
clf.fit(X_train,y_train)
res3 = clf.predict(X_test)
print 'KNN', np.sum(res3==y_test) / float(len(y_test))
```

```
!python testbm.py
```

```
Iteration 0
Iteration 10
Iteration 20
Iteration 0
Iteration 10
Iteration 20
Iteration 0
Iteration 10
Iteration 20
Boltzmann Makinasi 0.975
KNN 0.975
```

Sonuc yuzde 97.5, oldukca yuksek, ve KNN metodu ile ayni sonucu aldik, ki bu aslında oldukca temiz / basit bir veri seti için fena değil.

### Biraz Hikaye

Boltzman Makinalarıyla ilgilenmemizin ilginç bir hikayesi var. Aslında bu metodtan haberimiz yoktu, ayrıca mevcut isimizde 0/1 içeren ikisel verilerle çok hasır nesirdik, ve bu tür verilerde ikisel ilişkiler (cooccurence) hesabi iyi sonuçlar verir, ki bu hesap basit bir matris carpimi ile elde edilir.

```
import numpy as np
A = np.array([\
[0.,1.,1.,0],\
[1.,1.,0, 0],\
[1.,1.,1.,0],\
[0, 1.,1.,1.],\
[0, 0, 1.,0]\
])
c = A.T.dot(A).astype(float)
print c

[[ 2.  2.  1.  0.]
 [ 2.  4.  3.  1.]
 [ 1.  3.  4.  1.]
 [ 0.  1.  1.  1.]]
```

Burada bakılırsa 2. satır 3. kolon 3 değerini taşıyor çünkü 2. ve 3. değişkenlerin aynı anda 1 olma sayısı tam olarak 3. Sonra acaba bu bilgiyi veri üzerinde hesaplayıp bir kenara koysak bir dağılım gibi kullanamaz mıyız, sonra yeni veri noktasını bu “dağılıma sorabiliriz” diye düşündük. Biraz matris carpim cambazlığı sonrası, yeni veri noktası için

```
x = np.array([0,1,1,0])
print np.dot(np.dot(x.T,c), x) / 2

7.0
```

gibi sonuclar alabildigimizi gorduk; Bu degerin iliski matrisinin tam ortasin-daki 4,3,3,4 sayilarinin toplamının yarisi olduguna dikkat edelim. Yani  $x$  carpimi iliski matrisinin sadece kendini ilgilendiren kismini cekip cikartti, yani 2. ve 3. degisenleri arasindaki iliskiye toplayip aldi.

Buradan sonra, “acaba bu bir dagilim olsa normalizasyon sabiti ne olurdu?” sorusuna geldik, ki [4] sorusu buradan cikti ve bu soruya bomba bir cevap geldi. Sonra diger okumalarimiz sirasinda Boltzmann Dagilimina ulastik, bu dagilimin ek olarak bir exp tanimi var (ki turev alimi sirasinda bu faydali), ve tabii ogrenim icin daha net bir matematigi var. Biz de maksimum olurluk ile [4]’teki fikrin sayisal kovaryansa ulastirip ulastirmayacagini merak ediyorduk, BM formunda verisel kovaryans direk elde ediliyor. Boylece BM konusuna girmis olduk.

Fakat daha iyi haber BM’in, Kisitli BM (RBM) icin bir ziplama tahtasi olmasi, zaten RBM’den sonra Derin Ogrenim (Deep Learning) konusu geliyor, cunku DO birden fazla RBM’lerin ust uste konmus hali.

[1] Information Theory, Inference and Learning Algorithms, D. MacKay, sf. 523

[2] <http://nbviewer.ipython.org/gist/aflaxman/7d946762ee99daf739f1>

[3] <http://math.stackexchange.com/questions/1095491/from-pxw-fraclzw-exp-bigl-fracl2-xt-w-x-bigr-to-sigmoid/>

[4] <http://math.stackexchange.com/questions/1080504/calculating-the-sum-fracl2-sum-xt-sigma-x-for-all-x-in-0-1-n>