Boltzman Makinalari (Rasgele Hopfield Aglari)

Alttaki ifade bir Boltmann dagilimini gosterir,

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

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. Bolzmann Makinalari (BM), Kisitli Bolzmann Makinalari (Restricted Bolzmann Machines) kavramina gecis yapmadan once iyi bir durak noktasi.

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

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

Log olurluk

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

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,

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

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

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

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

$$\frac{\partial}{\partial w_{ij}} \exp\big(\frac{1}{2}x^T W x\big) = \frac{1}{2}\exp\big(\frac{1}{2}x^T W x\big)\frac{\partial}{\partial w_{ij}} x^T W x \tag{2}$$

(2)'in icindeki bolumu acalim,

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

Simdi (2)'ye geri koyalim,

$$= \frac{1}{2}\exp\big(\frac{1}{2}x^T W x\big) x_i x_j$$

$$\frac{\partial}{\partial w_{ij}} \ln Z(W) = \frac{1}{Z(W)}\Big[\sum_x \frac{1}{2}\exp\big(\frac{1}{2}x^T W x\big) x_i x_j\Big]$$

$$= \frac{1}{2}\sum_x \frac{1}{Z(W)} \exp\big(\frac{1}{2}x^T W x\big) x_i x_j$$

$$= \frac{1}{2}\sum_x P(x;W) x_i x_j$$

Ustteki son ifadede bir kisaltma kullanalim,

$$\sum_x P(x;W) x_i x_j = < x_i, x_j >_{P(x;W)} \tag{4}$$

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

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_{n=1}^{N}\Big[\frac{\partial}{\partial w_{ij}}\frac{1}{2}x^{(n)^T} W x^{(n)} - \frac{\partial}{\partial w_{ij}} \ln Z(W)\Big]$$

$$= \sum_{n=1}^{N}\Big[\frac{1}{2}x_i^{(n)^T} x_j^{(n)} - \frac{\partial}{\partial w_{ij}} \ln Z(W)\Big]$$

$$= \sum_{n=1}^{N}\Big[\frac{1}{2}x_i^{(n)^T} x_j^{(n)} - \frac{1}{2} < x_i x_j >_{P(x;W)}\Big]$$

$1/2$ sabitlerini atalim,

$$= \sum_{n=1}^{N} \left[x_i^{(n)^T} x_j^{(n)} - < x_i x_j >_{P(x;W)} \right]$$

Eger

$$< x_i x_j >_{Data} = \frac{1}{N} \sum_{n=1}^{N} x_i^{(n)^T} x_j^{(n)}$$

olarak alirsak, esitligin sag tarafi verisel kovaryansi (empirical covariance) temsil eder. Duzenleyince,

$$N \cdot < x_i x_j >_{Data} = \sum_{n=1}^{N} x_i^{(n)^T} x_j^{(n)}$$

simdi esitligin sag tarafi uc ustteki formule geri koyulabilir,

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = N \big[< x_i x_j >_{Data} - < x_i x_j >_{P(x;W)} \big]$$

Her ne kadar N veri noktasi sayisini gosteriyor olsa da, ustteki ifade bir gradyan guncelleme formulu olarak ta gorulebilir, ve N yerine bir guncelleme sabiti alinabilir. Gradyan guncelleme olarak gorulebilir cunku w_{ij}'ye gore turev aldik, o zaman bizi $\mathcal{L}$'in minimumuna goturecek w adimlari ustte goruldugu gibidir.

(4)'te gorulen $< x_i x_j >_{P(x;W)}$'in anlami nedir? Bu ifade / carpim tum mumkun x degerleri uzerinden aliniyor ve ikisel iliskilerin olasiligini "mevcut modele" gore hesapliyor. Yani bu ifade de bir korelasyon hesabidir, sadece veriye gore degil, tum mumkun degerler ve model uzerinden alinir. Bu ifadeyi hesaplamak icin Monte Carlo simulasyonu kullanacagiz. Tum degerler uzerinden gecmek yerine mevcut modele x degerleri "urettirecegiz", ve bu degerleri alip sanki gercek veriymis gibi sayisal korelasyonlarini hesaplayacagiz. Sonra bu degeri gercek verinin korelasyonunundan cikartip bir sabit uzerinden gradyan adimi atmak mumkun olacak.

Gibbs Orneklemesi (Sampling)

Gibbs orneklemesinin detaylari icin *Monte Carlo, Entegraller, MCMC* yazisina danisilabilir. Bolzmann dagilimindan orneklem almak icin bize tek bir degisken (hucre) haricinde diger hepsinin bilindigi olasilik hesabi lazim, yani $P(x_i = 1 | x_j, j \neq i)$. Ana dagilim fonksiyonu baz alinarak, yeni veri x uzerinde o x uzerinde biri haric tum ogelerin bilindigi durumda bilinmeyen tek hucre i icin 1 olma olasilik degeri,

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

ve,

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

Bu kosulsal olasiligin ne kadar temiz oldugu onemli, ustteki gorulen bir sigmoid fonksiyonudur nihayetinde... Bu fonksiyonlar hakkinda daha fazla bilgi *Lojistik Regresyon* yazisinda bulunabilir.

Ama, ana formul (3)'ten bu noktaya nasil eristik?

x vektoru icinde sadece x_i ogesinin b olmasini x^b olarak alalim. Once kosulsal dagilimda "verili" olan kismi elde etmek lazim. O zaman

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

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, ayrica ulasmak istedigimiz $1/1+e^{-a_i}$ ifadesinde $+1$'den kurtulmak iyi olur, boylece sadece e^{-a_i} olan esitligi ispatlariz. Bunun her iki denklemde ters cevirip 1 cikartabiliriz,

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

$$= 1 + \frac{P(x^0)}{P(x^1)}$$

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

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

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

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

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,

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

Ic dis toplamlari birlestirelim,

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

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

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

$$\exp\Big(\sum_k Y^0_{ki} + \sum_j Y^0_{ij} + Y^0_{ii} - \big(\sum_k Y^1_{ki} + \sum_j Y^1_{ij} + Y^1_{ii} \big) \Big)$$

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

$$= \exp\Big(0 - \big(\sum_k Y^1_{ki} + \sum_j Y^1_{ij} + Y^1_{ii} \big) \Big)$$

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

$$= \exp\Big(- \big(2 \sum_j Y^1_{ij} + Y^1_{ii} \big) \Big)$$

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

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

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):
        """
        draw - perform single Gibbs sweep to draw a sample from distribution
        """
        N=Sin.shape[0]
        S=Sin.copy()
        rand = np.random.rand(N,1)
        for i in xrange(N):
            h=np.dot(T[i,:],S)
            S[i]=rand[i]<self.sigmoid(h);
        return S

    def sample(self, T):
        N=T.shape[0]
        s=np.random.rand(N)<self.sigmoid(0)
        for k in xrange(self.init_sample_size):
            s=self.draw(s,T)
        S=np.zeros((N,self.sample_size))
        S[:,0]=s
        for i in xrange(1,self.sample_size):
            S[:,i]=self.draw(S[:,i-1],T)
        return S.T

    def normc(self, X):
        """
        Return the normalization constant
        """
        def f(x): return np.exp(0.5 * np.dot(np.dot(x,self.W), x))
        S = 2*self.sample(self.W)-1
        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
```

```
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

np.random.seed(0)
clf = boltz.Boltzmann(n_iter=30,eta=0.05,sample_size=100,init_sample_size=50)
clf.fit(A)
print 'W'
print clf.W
print 'normalizasyon sabiti', clf.C

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

```
Iteration 0
Iteration 10
Iteration 20
W
[[ 0.     0.05  -0.075 -0.375]
 [ 0.05   0.     0.05   0.45 ]
 [-0.075  0.05   0.     0.325]
 [-0.375  0.45   0.325  0.   ]]
normalizasyon sabiti 20.1580365398
[ 0.11319955  0.05215146  0.11319955]
```

```
Y = np.loadtxt('../../stat/stat_mixbern/binarydigits.txt')
print Y.shape
```

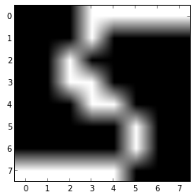
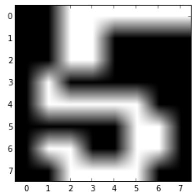
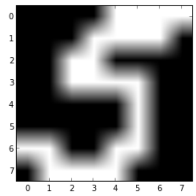
```
(100, 64)
```

```
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')
```

```
! python testbm.py
```

```
Iteration 0
Iteration 10
Iteration 20
Iteration 0
Iteration 10
Iteration 20
Iteration 0
Iteration 10
Iteration 20
0.975
KNN 0.975
```

Information Theory, Inference and Learning Algorithms, D. MacKay

http://nbviewer.ipython.org/gist/aflaxman/7d946762ee99daf739f1

http://math.stackexchange.com/questions/1095491/from-pxw-frac1zw-exp-bigl-frac12-xt-w-x-bigr-to-sigmoid/