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,

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

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,

$$\exp\big(\sum_{k} Y^0_{ki} + \sum_{j} Y^0_{ij} + Y^0_{ii} - (\sum_{k} Y^1_{ki} + \sum_{j} Y^1_{ij} + Y^1_{ii}) \big)$$

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

$$= \exp\big(0 - (\sum_{k} Y^1_{ki} + \sum_{j} Y^1_{ij} + Y^1_{ii})\big)$$

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

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

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

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

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 (hucreyi) icin, diger degiskenleri baz alip digerinin kosulsal olasiligini hesapliyoruz, ayni anda bu olasiligi kullanarak bir sayi uretimi yapiyoruz. Uretimin yapilmasi icin `np.random.rand`'dan gelen 0 ve 1 arasindaki bir uniform rasgele sayiyi gecip gecmeme irdelemesi yeterli. Bir olasiligi bir uretilen rasgele degiskene bu sekilde cevirebilirsiniz. Bu ilginc bir numara, ayrica Simulasyon adli bir matematik dali tamamen bu islerle ugrasir. Yani MCMC daha cetrefil dagilimlar icin kullanilir tabii, ama baz dagilimlar, mesela Normal, Ustel (Exponential), Bernoulli gibi dagilimlarin hep uretilme teknikleri vardir.

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 hesaplanmasi icin aynen $< x_i x_j >_{P(x;W)}$ icin oldugu gibi tum mumkun x'ler uzerinden bir toplam gerekir. Bunu bilmedigimize gore, 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 bakarsa 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.

Yeni test verisini dagilima gecelim,

```
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 ]
```

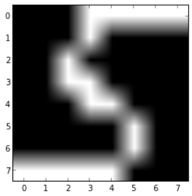
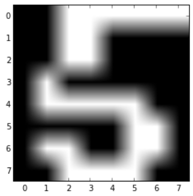
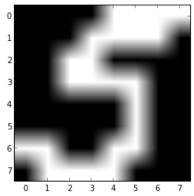
Goruntu Tanima

Elimizde el yazisi tanima algoritmalari icin kullanilan bir veri seti var. Veride 0,5,7 harflerinin goruntuleri var. Mesela 5 icin bazi ornek goruntuler,

```
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 goruntuleri tanimak icin BM kullanalim. Egitim ve test olarak veriyi ikiye ayiracagiz, ve egitim seti her etiketin W'sini ogrenmek icin kullanilacak. Daha sonra test setinde her veri noktalarini her BM'ye (ucune) ayri ayri gecip o test verisinin "olasiligini" soracagiz, ve hangi BM daha yuksek olasilik donduruyorsa etiket olarak onu kabul edecegiz. Hangi BM daha yuksek olasilik donduruyorsa, o BM "bu verinin benden gelme olasiligi yuksek" diyor demektir, ve etiket o BM olmalidir.

```
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_st
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
```

KNN ile ayni sonucu aldik, ki bu aslinda oldukca temiz / basit bir veri seti icin fena degildir.

Biraz Hikaye

Boltzman Makinalariyla ilgilenmemizin ilginc bir hikayesi var. Aslinda BM'den haberimiz yoktu, ayrica mevcut isimizde 0/1 iceren ikisel verilerle cok hasir nesirdik, ve bu tur verilerde ikisel iliskiler (cooccurence) hesabi iyi sonuclar verir, ki bu hesap basit bir matris carpimi ile elde edilir.

```python
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 bakilirsa 2. satir 3. kolon 3 degerini tasiyor cunku 2. ve 3. degiskenlerin ayni anda 1 olma sayisi tam olarak 3. Sonra dedik ki, acaba bu bilgiyi veri uzerinde hesaplayip bir kenara koysak bir dagilim gibi kullanamaz miyiz, sonra yeni veri noktasini bu "dagilima sorabiliriz", olasiligi nedir? Biraz matris carpim cambazligi sonrasi, yeni veri noktasi icin sonuc

```python
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 ortasindaki 4,3,3,4 sayilarinin toplaminin yarisi olduguna dikkat. Yani yani x carpimi iliski matrisinin sadece kendini ilgilendiren kismini cekip cikartti, yani 2. ve 3. degisenleri arasindaki iliskiyi 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 verisal 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-frac1zw-exp-bigl-frac12-xt-w-x-bigr-to-sigmoid/`

[4] `http://math.stackexchange.com/questions/1080504/calculating-the-sum-frac12-sum-xt-sigma-x-for-all-x-in-0-1-n`