

Kisitli Boltzmann Makinalari (Restricted Boltzmann Machines -RBM-)

Bir RBM icinde ikisel (binary) degerler tasiyan gizli (hidden) h degiskenler, ve yine ikisel gorunen (visible) degiskenler v vardir. Z aynen once gordugumuz Boltzman Makinalarinda (BM) oldugu gibi normalizasyon sabitidir.

$$p(x, h; W) = \exp(-E(x, h))/Z$$

E tanimina “enerji” olarak ta atif yapilabiliyor.

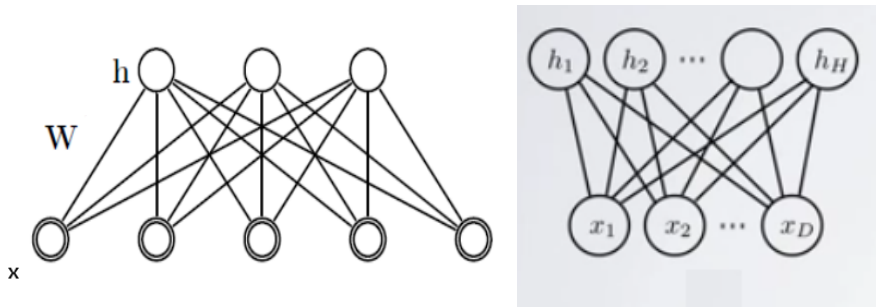
$$E(x, h) = -h^T W x - c^T x - b^T h$$

BM’lerden farkli olarak RBM taniminda c, b degiskenleri var. Bu degiskenler yanlilik (bias) icin, yani veri icindeki genel egilimi saptamaları icin modele konulmustur. Ayrica $h^T W x$ terimi var, bu BM’deki $x^T W x$ biraz farkli, gizli degiskenler, h üzerinden x ’ler arasinda baglanti yapiyor. Bir baska ilginc farklilik BM ile tum x ogeleri birbirine baglanabiliyordu, RBM ile daha az (ya da fazla) olabilecek h katmaninda baglantilar paylasiliyor. Ozellikle azaltma durumunda RBM ozellik alanini azaltarak bir tur genellemeyi gerceklestirebiliyor.

Formul alttaki gibi de acilabilir,

$$= - \sum_j \sum_k W_{j,k} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j$$

RBM’lerin alttaki gibi resmedildigini gorebilirsiniz.



h, x degiskenleri olasilik teorisinden bilinen rasgele degiskenlerdir, yani hem x ’e hem de h ’e “zar attirabiliriz” / bu degiskenler uzerinden orneklem toplayabiliriz.

Ayrica, RBM’ler aynen BM’ler gibi bir olasilik yogunluk fonksiyonu uzerinden tanimlanirlar, onceki formilde gordugumuz gibi, tum mumkun degerleri uzerinden entegralleri (ya da toplamlari) alinca sonuc 1 olur, vs.

RBM’lerin “kisitli” olarak tanimlanmalarinin sebebi gizli degiskenlerin kendi aralarinda, ve gorunen degiskenlerin kendi aralarinda direk baglantiya izin verilmemis olmasidir, bu bakimdan “kisitlanmislardır”. Baglantilara, W uzerinden

sadece gizli ve gorunen degiskenler (tabakalar) arasinda izin verilmistir. Bu tabii ki matematiksel olarak bazi kolayliklar sagliyor.

Devam edelim, ana formulden hareketle cebirsel olarak sunlar da dogrudur,

$$\begin{aligned}
 p(x, h; W) &= \exp(-E(x, h))/Z \\
 &= \exp(h^T W x + c^T x + b^T h)/Z \\
 &= \exp(h^T W x) \exp(c^T x) \exp(b^T h)/Z
 \end{aligned} \tag{2}$$

cunku bir toplam uzerindeki exp, ayri ayri exp'lerin carpimi olur. Ayni mantikla, eger ana formulu matris / vektor yerine ayri degiskenler olarak gormek istersek,

$$p(x, h; W) = \frac{1}{Z} \prod_j \prod_k \exp(W_{jk} h_j x_k) \prod_k \exp(c_k x_k) \prod_j \exp(b_j h_j)$$

Notasyonu kolaylastirmak amaciyla b, c terimlerini W icine absorbe edebiliriz, $x_0 = 1$ ve $h_0 = 1$ degerlerini mecbur tutarsak ve $w_{0,:} = c$ ve $w_{:,0} = b$ dersek, yani W 'nin sifirinci satirinin tamaminin c oldugunu, sifirinci kolonunun tamaminin b oldugunu kabul edersek RBM ana formulunu tekrar elde etmis oluruz, fakat artik

$$\begin{aligned}
 E(x, h) &= -h^T W x \\
 &= - \sum_j \sum_k W_{j,k} h_j x_k
 \end{aligned}$$

ve

$$p(x, h; W) = \exp(h^T W x)/Z$$

yeterli olacaktır. Bir diger kolaylik x, h yerine tek degisken kullanmak,

Eger $y \equiv (x, h)$ olarak alirsak,

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

Aslinda acik konusmak gerekirse “enerji” gibi kavramlarla ugrasmak, ya da icinde eksi terimler iceren bir grup degiskenin tekrar eksisini almak ve eksilerin etkisinin notralize etmis olmak yerine bastan (2)'deki ifadeyle yola cikmak daha

kisadir. Icinde enerji olan aciklamalari biraz da literaturde gorulebilecek anlamlara aciklik getirmek icin yaptik.

Neyse, h uzerinden marjinalize edersek,

$$\begin{aligned} P(x; W) &= \sum_h \frac{1}{Z(W)} \exp \left[\frac{1}{2} y^T W y \right] \\ P(x; W) &= \frac{1}{Z(W)} \sum_h \exp \left[\frac{1}{2} y^T W y \right] \end{aligned} \quad (1)$$

Ve $Z(W)$

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

(1) denkleminde bolumunden sonraki kisma $Z_x(W)$ dersek, sanki ayni exp denkleminin x 'ler uzerinden marjinalize edilmiş hali olarak gosterebiliriz onu, ve boylece daha kısa bir formül kullanabiliriz,

$$P(x; W) = \frac{1}{Z(W)} \underbrace{\sum_h \exp \left[\frac{1}{2} y^T W y \right]}_{Z_x(W)}$$

O zaman

$$P(x; W) = \frac{Z_x(W)}{Z(W)}$$

elde ederiz. Veri uzerinden maksimum olurluk icin, yine log uzerinden bir hesap yapariz, BM icin yapmistik bunu,

$$\begin{aligned} \mathcal{L} &= \ln \left(\prod_{n=1}^N P(x^n; W) \right) = \sum_{n=1}^N \ln P(x^n; W) \\ &= \sum_{n=1}^N \ln \frac{Z_{x^{(n)}}(W)}{Z(W)} = \sum_{n=1}^N (\ln Z_{x^{(n)}} - \ln Z) \\ \frac{\partial \mathcal{L}}{\partial w_{ij}} &= \sum_{n=1}^N \left(\frac{\partial \ln Z_{x^{(n)}}}{\partial w_{ij}} - \frac{\partial \ln Z}{\partial w_{ij}} \right) \end{aligned} \quad (3)$$

Parantez icindeki 1. turevi alalim,

$$\begin{aligned}
\frac{\partial \ln Z_{x^{(n)}}}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}} \ln \left[\sum_h \exp \left(\frac{1}{2} y^{n\top} W y^n \right) \right] \\
&= \frac{1}{Z_{x^{(n)}}} \left[\sum_h \frac{\partial}{\partial w_{ij}} \exp \left(\frac{1}{2} y^{n\top} W y^n \right) \right] \\
&= \frac{1}{Z_{x^{(n)}}} \left[\sum_h \exp \left(\frac{1}{2} y^{n\top} W y^n \right) \frac{\partial}{\partial w_{ij}} y^{n\top} W y^n \right] \\
&= \frac{1}{Z_{x^{(n)}}} \sum_h \exp \left(\frac{1}{2} y^{n\top} W y^n \right) y_i y_j \\
&= \sum_h \frac{1}{Z_{x^{(n)}}} \exp \left(\frac{1}{2} y^{n\top} W y^n \right) y_i y_j
\end{aligned}$$

$Z_{x^{(n)}}$ 'nin ne olduğunu hatırlarsak, exp ifadesinin h üzerinden marjinalize edilmiş hali,

$$= \sum_h \frac{\exp \left(\frac{1}{2} y^{n\top} W y^n \right)}{\sum_h \exp \left(\frac{1}{2} y^{n\top} W y^n \right)} y_i y_j$$

Eğer bolumun üstünü ve altını Z ile bösek,

$$= \sum_h \frac{\exp \left(\frac{1}{2} y^{n\top} W y^n \right) / Z}{\sum_h \exp \left(\frac{1}{2} y^{n\top} W y^n \right) / Z} y_i y_j$$

Üst kısım $P(y; W)$ yani $P(x, h; W)$ alt kısım $P(x; W)$ olmaz mı? Evet! Ve,

$$P(h|x^n; W) = \frac{P(x^n, h; W)}{P(x^n; W)}$$

olduguna göre,

$$= \sum_h P(h|x^n; W) y_i y_j$$

elde ederiz. Bunu da $\langle y_i y_j \rangle_{P(h|x^n; W)}$ olarak yazabiliriz.

Şimdi parantez içindeki 2. türevi alalım, yani $\frac{\partial \ln Z}{\partial w_{ij}}$,

$$\frac{\partial \ln Z}{\partial w_{ij}} = \sum_{h,x} \frac{1}{Z} \exp \left(\frac{1}{2} y^\top W y \right) y_i y_j = \sum_{h,x} P(y; W) y_i y_j$$

ki bu son ifadeyi de $\langle y_i y_j \rangle_{P(y;W)}$ olarak yazabiliriz. Tamamini, yani (3) ifadesini, artık soyle yazabiliriz,

$$\sum_{n=1}^N \left(\frac{\partial \ln Z_{x^{(n)}}}{\partial w_{ij}} - \frac{\partial \ln Z}{\partial w_{ij}} \right) = \sum_{n=1}^N \langle y_i y_j \rangle_{P(h|x^n;W)} - \langle y_i y_j \rangle_{P(y;W)} \quad (4)$$

Bu formulu de BM icin yaptigimiz gibi bir gradyan guncelleme formulune donusturebiliriz. Guncelleme formulunun hangi hesaplari gerektirdigine gelince; İlk terim tum h 'ler uzerinden, ikincisi ise tum x , h 'ler uzerinden bir olasilik hesabi ve ornekleme gerektirecek. Bu durum cetin hesap (intractable) denen bir durum, ve daha once BM icin bu problemi Gibbs orneklemesi ile cozmustuk. Ayni cozumu burada da uygulayabiliriz, fakat belki daha iyi bir yaklasim su olacak.

CD Yontemi (Contrastive Divergence)

RBM'leri egitmek icin kullanılan en populer yontem CD yontemidir. Bu yontemi gecmeden once bazi matematiksel kolayliklari bilmek gerekli.

RBM grafigine bakarsak, eger x biliniyor ise bu h degiskenlerini bagimsiz hale getirir (kosullu olasilik kurali), ve ayni sekilde h biliniyor ise x bagimsiz hale gelir. Bunu gorsel olarak bile anlamak cok kolay, elimizle tum x 'leri kapatalim mesela ve h dugumlerine bakalim, aralarinda hicbir baglanti yoktur degil mi? Ayni sekilde h kapatınca x 'ler "baglantısız" hale gelir.

Bu bagimsizlikten yola cikarak, daha once BM icin yaptigimiz gibi, olasiliklar su basit formullere donusur,

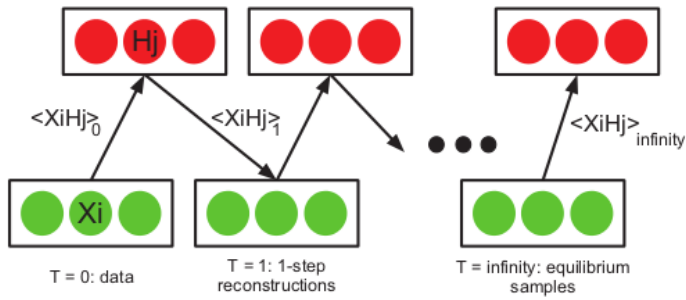
$$p(h_i|x) = \sigma \left(\sum_{j=1}^m w_{ij} x_j \right)$$

$$p(x_i|h) = \sigma \left(\sum_{j=1}^n w_{ij} h_j \right)$$

ve tabii ki $\sigma(x) = 1/(1 + e^{-x})$. Daha once 1 olma olasiligini nasil ornekleme cevirecegimizi de gormustuk zaten.

Simdi CD'nin ne olduguna geelim. Eger RBM icin gereken orneklemeyi klasik Gibbs ile yaparsak orneklemeyi "yeterince uzun sure" yapmak gerekir ki dagilimin olasi noktaları gezilmissin olsun. Fakat, ozellikle yuksek boyutlu durumlarda, tum x, h kombinasyonlarını dusunursek bu cok buyuk bir alandır. Bunun yerine, ve ustteki bagimsizlik formullerinden hareketle CD yontemi bulunmustur, bu yonteme gore ornekleme verinin *kendisinden* baslatilir (kiyasla Gibbs rasgele bir noktadan), dongunun mesela ilk adiminda x^0 (ki bu tum verinin tamamı), o baz alınarak $p(h^0|x^0)$ hesaplanır (ustteki sigmoid, onun uzerinden h^0 ornekleme). Sonra h^0 baz alınir ve x^1 , bu boyle devam eder. Boylece mumkun h ve x 'ler gezilmissin olur.

Literatürde şu şekildeki resim bolca görülebilir,



Bu yöntem pur Gibbs örneklemesine kıyasla çok daha hızlı işler ve iyi sonuçlar verir. Teorik olarak niye işlediği [1,2] makalelerinde bulunabilir. CD aslında (4) hedef formülünü değil başka bir hedefi optimize ediyor, fakat sonuç (4)'ün yapmak istediğine yakın. Kod [3] baz alınarak, şu şekilde,

```
import numpy as np
import itertools

class RBM:

    def __init__(self, num_hidden, learning_rate, max_epochs, num_visible=10):
        self.num_hidden = num_hidden
        self.num_visible = num_visible
        self.learning_rate = learning_rate
        self.norm_dict = {}
        self.weights = 0.1 * np.random.randn(self.num_visible, self.num_hidden)
        self.weights = np.insert(self.weights, 0, 0, axis = 0)
        self.weights = np.insert(self.weights, 0, 0, axis = 1)
        self.max_epochs = max_epochs

    def fit(self, data):
        num_examples = data.shape[0]

        data = np.insert(data, 0, 1, axis = 1)

        for epoch in range(self.max_epochs):
            pos_hidden_activations = np.dot(data, self.weights)
            pos_hidden_probs = self._logistic(pos_hidden_activations)
            pos_hidden_states = pos_hidden_probs > \
                np.random.rand(num_examples, self.num_hidden + 1)

            tmp = np.array(pos_hidden_states).astype(float)
            pos_visible_states = self.run_hidden(tmp[:,1:])

            pos_associations = np.dot(data.T, pos_hidden_probs)

            neg_visible_activations = np.dot(pos_hidden_states, self.weights.T)
            neg_visible_probs = self._logistic(neg_visible_activations)
            neg_visible_probs[:,0] = 1 # Fix the bias unit.
            neg_hidden_activations = np.dot(neg_visible_probs, self.weights)
```

```

neg_hidden_probs = self._logistic(neg_hidden_activations)

neg_associations = np.dot(neg_visible_probs.T, neg_hidden_probs)

self.weights += self.learning_rate * \
    ((pos_associations - neg_associations) / num_examples)

error = np.sum((data - neg_visible_probs) ** 2)

def run_hidden(self, data):

    num_examples = data.shape[0]

    visible_states = np.ones((num_examples, self.num_visible + 1))

    data = np.insert(data, 0, 1, axis = 1)

    visible_activations = np.dot(data, self.weights.T)
    visible_probs = self._logistic(visible_activations)
    visible_states[:, :] = visible_probs > \
        np.random.rand(num_examples, self.num_visible + 1)

    visible_states = visible_states[:, 1:]
    return visible_states

def run_visible(self, data):

    num_examples = data.shape[0]

    hidden_states = np.ones((num_examples, self.num_hidden + 1))

    data = np.insert(data, 0, 1, axis = 1)

    hidden_activations = np.dot(data, self.weights)
    hidden_probs = self._logistic(hidden_activations)
    hidden_states[:, :] = hidden_probs > \
        np.random.rand(num_examples, self.num_hidden + 1)
    hidden_states = hidden_states[:, 1:]
    return hidden_states

def _logistic(self, x):
    return 1.0 / (1 + np.exp(-x))

import cPickle, numpy as np, gzip, rbm
from sklearn import neighbors
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LogisticRegression

from sklearn.base import BaseEstimator
from sklearn.base import TransformerMixin

class SKRBM(BaseEstimator, TransformerMixin):
    def __init__(self, n_components=1, learning_rate=0.1, n_iter=10):
        self.n_components = n_components

```

```

        self.learning_rate = learning_rate
        self.n_iter = n_iter

    def transform(self, X):
        return self.rbm_.run_visible(X)

    def fit(self, X, y=None):
        self.rbm_ = rbm.RBM(num_hidden=self.n_components, learning_rate=self.learning_rate)
        self.rbm_.fit(X)
        return self

X = np.loadtxt('../stat/stat_mixbern/binarydigits.txt')
Y = np.ravel(np.loadtxt('../stat/stat_mixbern/bindigitlabels.txt'))

from sklearn.cross_validation import KFold
scores = []
cv = KFold(n=len(X), n_folds=3)
for train, test in cv:
    X_train, Y_train = X[train], Y[train]
    X_test, Y_test = X[test], Y[test]
    r = SKRBM(n_components=30, learning_rate=0.3, n_iter=300)
    r.fit(X_train)
    clf = LogisticRegression(C=100)
    clf.fit(r.transform(X_train), Y_train)
    res3 = clf.predict(r.transform(X_test))
    scores.append(np.sum(res3==Y_test) / float(len(Y_test)))

print np.mean(scores)

# 98

```

! python test_rbmfold.py

0.979797979798

```

import cPickle, numpy as np, gzip
from sklearn import neighbors
from sklearn import svm
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LogisticRegression

X = np.loadtxt('../stat/stat_mixbern/binarydigits.txt')
Y = np.ravel(np.loadtxt('../stat/stat_mixbern/bindigitlabels.txt'))

from sklearn.cross_validation import KFold
scores = []
cv = KFold(n=len(X), n_folds=3)
for train, test in cv:
    X_train, Y_train = X[train], Y[train]
    X_test, Y_test = X[test], Y[test]
    clf = neighbors.KNeighborsClassifier()
    clf.fit(X_train, Y_train)
    scores.append(clf.score(X_test, Y_test))

```



```
print np.mean(scores)
```

```
# 97
```

```
! python test_knnkfold.py
```

```
0.970291146762
```

Kaynaklar

[1] Hinton, G., Training Products of Experts by Minimizing Contrastive Divergence

[2] Louppe, G., Collaborative filtering, Scalable approaches using restricted Boltzmann machines, Master Tezi, 2010

[3] <https://github.com/echen/restricted-boltzmann-machines>