多層制限ボルツマンマシンの実装

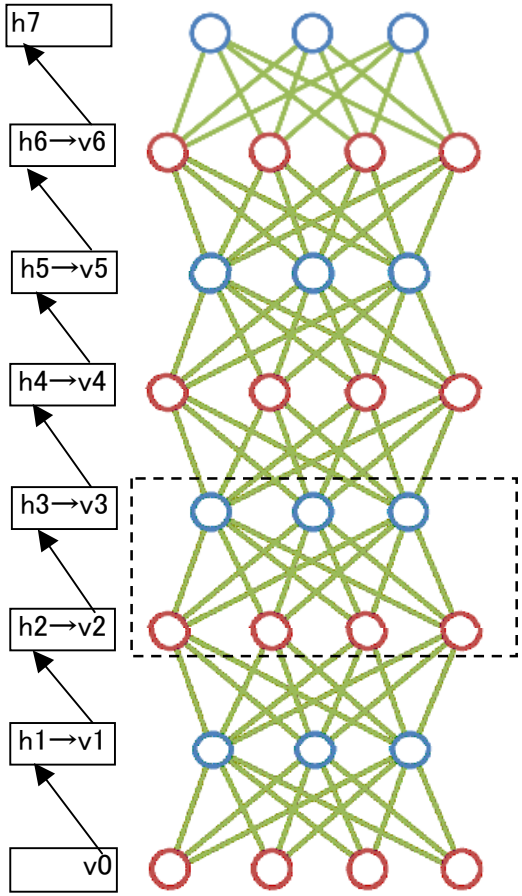
Implement For Deep RBM By C

2015/8/5

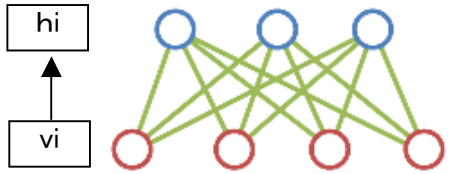
by @mabonki0725

RBMの数理

① 多層制限ボルツマンマシン(Deep RBM)
多層のネットワークは制限ボルツマンモデルの繰返と考え、下層から1段毎に処理していく



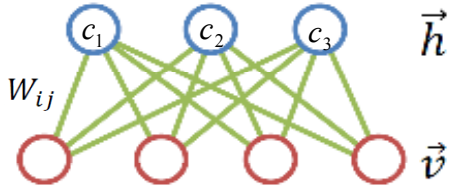
制限ボルツマンマシンモデル



注記
制限なしのボルツマンマシンモデル
は補足資料2 参照

RBMの数理

② 制限ボルツマンマシンモデル
ネットワークが上下にのみ結線しているネットワークモデルを制限ボルツマンマシンモデル(RBM)と云う



vとhは0か1を採る次の確率分布とする

$$p(v_i = 1|\mathbf{h}) = \text{sigmoid}\left(b_i + \sum_j h_j w_{ij}\right)$$

$$p(h_j = 1|\mathbf{v}) = \text{sigmoid}\left(c_j + \sum_i v_i w_{ij}\right)$$

$$E(h,v) = \boxed{-\sum_j b_j v_j} - \sum_i c_i h_i - \sum_{ij} v_j h_i w_{ij}$$

隠れ層hのみバイアス値Cを設ける

$$p(\mathbf{v},\mathbf{h}) = \frac{\exp(-E(\mathbf{v},\mathbf{h}))}{Z}$$

ボルツマン分布

v の 観測データがN件数ある場合、p(v,h) の最適な連結間の重み W と バイアス C を求めるには
最大対数尤度 max L となる WとC を求めればよい

$$L(\mathcal{D};\theta) \equiv \sum_{\mathbf{v}} q(\mathbf{v}) \log p(\mathbf{v})$$

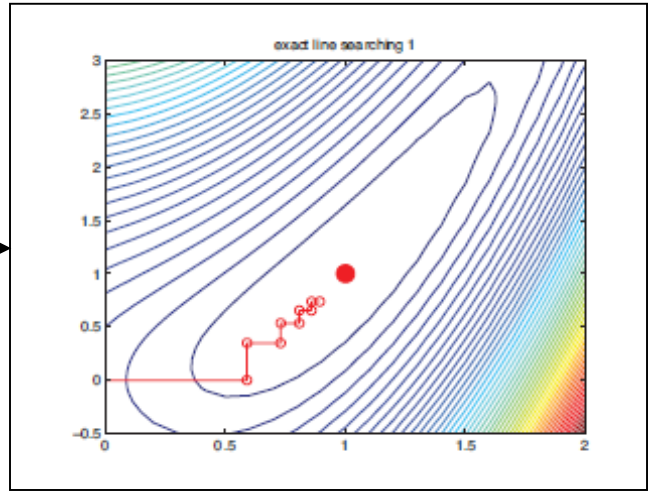
$$\log p(\mathbf{v}) = \log \sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h}) = \log \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v},\mathbf{h})\right) - \log Z$$

ここで
θ =[Ci Wij]
D : vのデータ
q(v) : 観測データの分布

RBMの数理

③ 制限ボルツマンマシンの最尤推定
最大対数尤度を算出するには、対数尤度Lをθ=[Wij Ci]で微分して確率的勾配法でθを徐々に更新して求めることを考える
（上下層のみの制約ボルツマンマシンなので、対数尤度関数は凹であると考えられる）

$$\theta' = \theta + \eta * \Delta L(D, \theta)$$



確率的勾配法

そこでθで部分した対数尤度は以下で与えら得る。

$$\frac{\partial \log p(v)}{\partial \theta} = \boxed{\frac{\partial}{\partial \theta}\left\{\log \sum_h \exp(-E(v,h))\right\}} - \boxed{\frac{\partial}{\partial \theta}\log Z}$$

$$\boxed{\frac{\partial}{\partial \theta}\left\{\log \sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))\right\}} = \frac{\frac{\partial}{\partial \theta}\left\{\sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))\right\}}{\sum_{\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))}$$

$$= \frac{\sum_h \exp(-E(v,h))\frac{\partial}{\partial \theta}E(v,h)}{\sum_h \exp(-E(v,h))} = \frac{Z\sum_{\mathbf{h}} p(\mathbf{v},\mathbf{h})\frac{\partial}{\partial \theta}E(\mathbf{v},\mathbf{h})}{Zp(\mathbf{v})}$$

$$= \boxed{-\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}}$$

$$\boxed{-\frac{\partial}{\partial \theta}\log Z} = -\frac{1}{Z}\frac{\partial}{\partial \theta}\sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v},\mathbf{h}))$$

$$= \boxed{\sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v},\mathbf{h})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}}$$

$$\frac{\partial \mathcal{L}(\mathcal{D},\theta)}{\partial \theta} = -\left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}\right\rangle_{\text{data}} + \left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta}\right\rangle_{\text{model}}$$

$$-\left\langle \frac{\partial E(v,h)}{\partial \theta}\right\rangle_{data} = -\sum_v q(v)\boxed{\sum_h p(h|v)\frac{\partial E(v,h)}{\partial \theta}} = -\frac{1}{N}\sum_j\sum_h p(h|v_j)\frac{\partial E(v_j,h)}{\partial \theta}$$

$$\left\langle \frac{\partial E(v,h)}{\partial \theta}\right\rangle_{model} = \sum_v q(v)\boxed{\sum_{vh} p(v,h)\frac{\partial E(v,h)}{\partial \theta}} = \frac{1}{N}\sum_j\sum_{jh} p(v_j,h)\frac{\partial E(v_j,h)}{\partial \theta}$$

θ=Wij の場合

$$\frac{\partial L(D,w_{ij})}{\partial w_{ij}} = -\frac{1}{N}\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}} + \sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v},\mathbf{h})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}}$$

$$= \frac{1}{N}\left\{\sum_h p(h|v_j)h_i v_j - \sum_{jh} p(v_j,h)h_i v_j\right\}$$

$$= \frac{1}{N}\left\{p(h=1|v_j)v_j - p(v_j,h=1)v_j\right\}$$

θ=Ci の場合

$$\frac{\partial L(D,c)}{\partial c_i} = \frac{1}{N}\left\{p(h=1|v_j) - p(v_j,h=1)\right\}$$

④CD-k(k Cycle Contrastive Divergence)法

$$-\left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right\rangle_{\text{data}} = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \boxed{p(\mathbf{h}|\mathbf{v})} \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \Big/ N$$

$$\left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right\rangle_{\text{model}} = \sum_{\mathbf{v}} \sum_{\mathbf{v},\mathbf{h}} \boxed{p(\mathbf{v},\mathbf{h})} \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \Big/ N$$

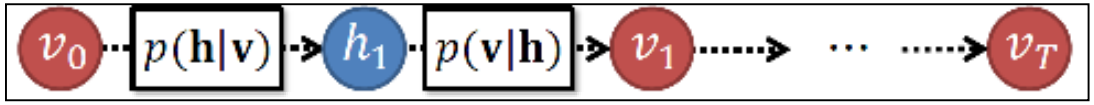
$$p(v_i = 1|\mathbf{h}) = \text{sigmoid}\left(b_i + \sum_j h_j w_{ij}\right)$$

$$\boxed{p(h_j = 1|\mathbf{v}) = \text{sigmoid}\left(c_j + \sum_i v_i w_{ij}\right)}$$

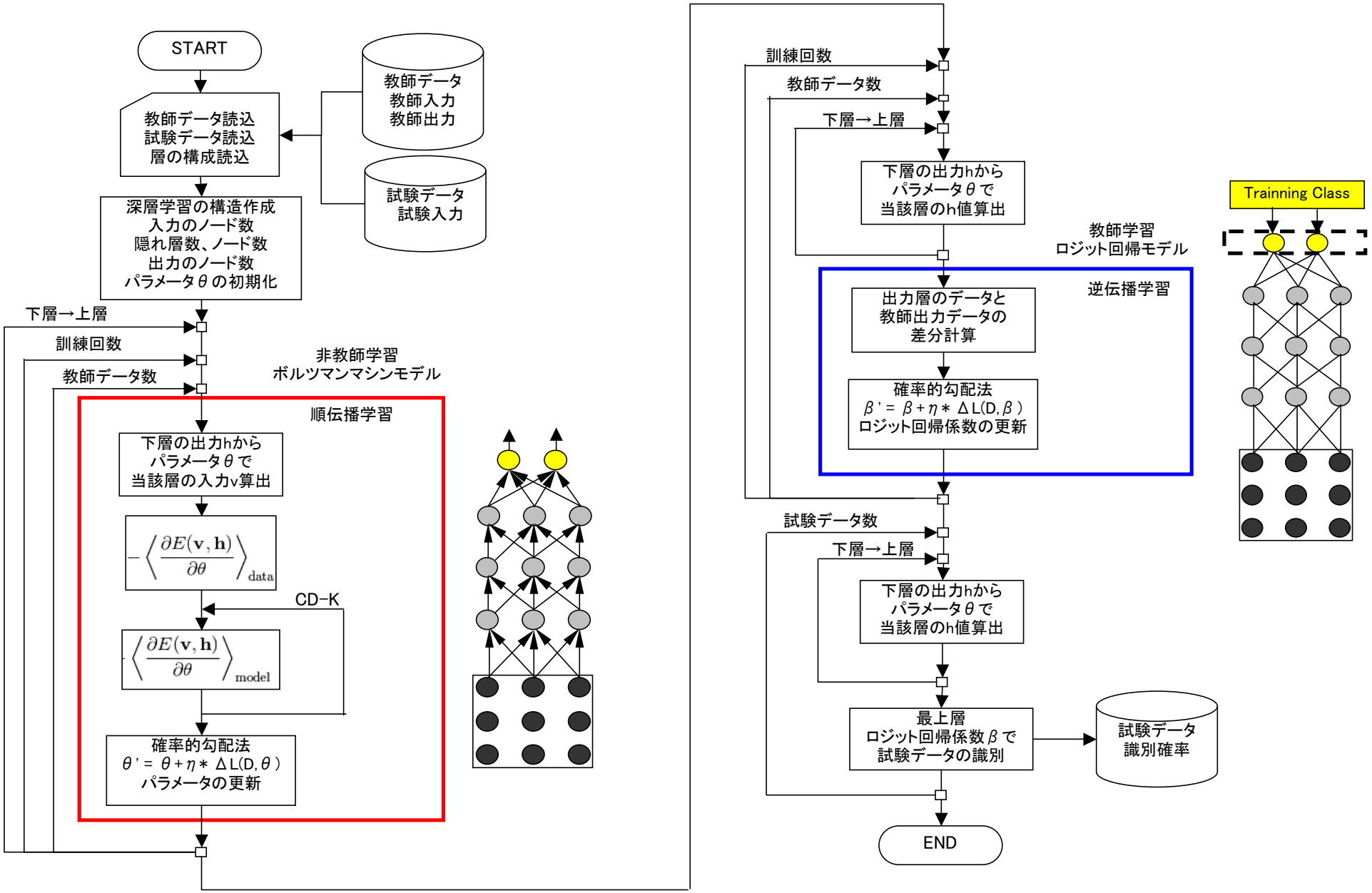
定義より算出できる

ここで P(v,h) は計算できないので、 バラメータを交互に繰り返すギブスサンプラーを行う
これをCD-k(k Cycle Contrastive Divergence)法という。

CD-k



全体フロー図


START
教師データ
教師入力
教師出力
試験データ
試験入力
教師データ読込
試験データ読込
層の構成読込
深層学習の構造作成
入力のノード数
隠れ層数、ノード数
出力のノード数
パラメータθの初期化
下層→上層
訓練回数
教師データ数
非教師学習
ボルツマンマシンモデル
順伝播学習
下層の出力hから
パラメータθで
当該層の入力v算出
$-\left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right\rangle_{\mathrm{data}}$
CD-K
$-\left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right\rangle_{\mathrm{model}}$
確率的勾配法
θ' = θ+η＊ ΔL(D,θ)
パラメータの更新
訓練回数
教師データ数
下層→上層
下層の出力hから
パラメータθで
当該層のh値算出
教師学習
ロジット回帰モデル
逆伝播学習
出力層のデータと
教師出力データの
差分計算
確率的勾配法
β' = β+η＊ ΔL(D,β)
ロジット回帰係数の更新
試験データ数
下層→上層
下層の出力hから
パラメータθで
当該層のh値算出
最上層
ロジット回帰係数βで
試験データの識別
試験データ
識別確率
END
Trainning Class

全体処理図

main(argc,argv)

Ⅰ 入力データを読込む

1.1 ファイルからデータを読む

1.2 引数から訓練データ行数と試験データ行数を得る

Ⅱ DBMIによる訓練と試験データの識別を行う

教師用データ　教師用識別　試験用データ　識別結果　データ数　教師データ数　入力ノード数　出力ノード数　隠れノード数　隠れ層数

test_dbn(train_x,　train_y,　test_x,　test_y,　nInst,　nTest,　nInp,　nOut,　nHide,　dHide);

2 層とノードの構成を作成

ネット　教師データ数　入力ノード数　隠れノード数　出力ノード数　隠れ層数

DBN_constrct(&dbn, train_N, n_ins, hidden_layer_sizes, n_outs, n_layers);

ノード構成を設定する。

入力ノード数、隠れ層の数、隠れ層のノード数、出力ノード数

層間の連結重み w[i][j]を初期値を一様分布で設定する

3 順伝播学習

ネット　教師データ　学習率　ギブス回数　トレーニング回数

DBN_pretrain(&dbn, *train_X, pretrain_lr,　k,　pretraining_epochs);

loop 入力層から出力層まで　当該層 i

loop トレーニング回数

loop 教師データレコード数 N

3_1入力層から当該層の前層まで、層間の連結重みで、当該層への入力値 layer_inputを計算する

3_2 次の関数で当該層の重みを確率的勾配法(学習率 lr)で更新する

当該層　当該層への入力データ　学習率　ギブス回数

RBM_contrastive_divergence(&(this->rbm_layers[i]), layer_input, lr,　k);

4 逆伝播学習

ネット　教師用データ　教師用識別　学習率　トレーニング回数

DBN_finetune(&dbn, *train_X, *train_Y, finetune_lr, finetune_epochs);

4_1 多クラスロジステック回帰係数を確率勾配法で算出

loop トレーニング回数

loop 訓練データレコード数 N

loop 入力層から出力層まで連結間の重みでノード値を計算

確率勾配法でロジット回帰係数を算出

5 試験データの予測

ネット　試験用データ　試験識別結果

DBN_predict(&dbn, test_X[i], test_Y[i]);

loop 試験データ数

loop 入力層から出力層まで

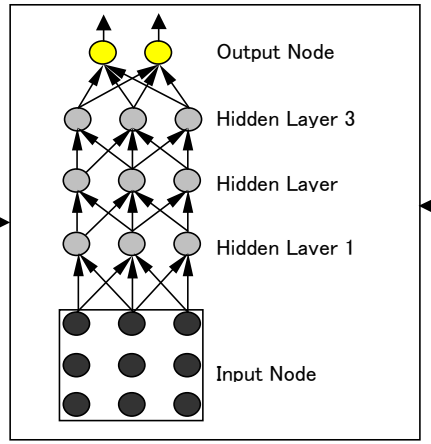
5_1入力層から最上層まで、連結間の連結重みで、出力層への入力値 layer_inputを計算する

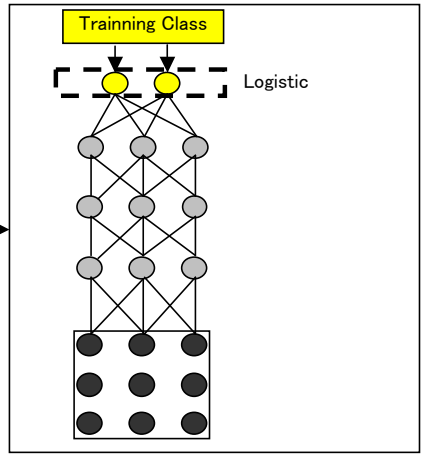
試験データの識別確率を計算する

ソフトマック関数で多クラス毎の確率を算出

当該層　識別確率

LogisticRegression_softmax(&(this->log_layer), y);





3 順伝播学習

3 順伝播

訓練データ　学習率　トレーニング回数

DBN_pretrain(&dbn, *train_X, pretrain_lr, k, pretraining_epochs);

loop 入力層から出力層まで　当該層 i

loop トレーニング回数

loop 訓練データレコード数 N

3_1入力層から当該層の前層まで、層間の連結重みで、当該層への入力値 layer_inputを計算する

3_2 次の関数で当該層の重みを確率的勾配法(学習率 lr)で更新する

RBM_contrastive_divergence(&(this->rbm_layers[i]), layer_input, lr, k);

$$\frac{\partial \mathcal{L}(\mathcal{D},\theta)}{\partial \theta} = -\left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right\rangle_{\text{data}} + \left\langle \frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \theta} \right\rangle_{\text{model}}$$

3_2_1 勾配をボルツマンモデルの微分式で計算する

3_2_1_1 微分式の前項(正値項)の算出

前層の値　p(h=1|v)　p(h=1|v)を閾値とする乱数

RBM_sample_h_given_v(this, input, ph_mean, ph_sample);

loop 当該層のノード数

3_2_1_1_1 p(h=1|v)の計算

p(h=1|v)　前層の値　当該層の重み 当該層のバイアス

mean[i] = RBM_propup(this, v0_sample, this->W[i], this->hbias[i]);

①前層の値に重み付けし総和する

loop 前層のノードの数 j

pre_sigmoid_activation += w[j] * v[j];

②バイアスを加算する

pre_sigmoid_activation += b;

③シグモイド関数で確率を求める

return sigmoid(pre_sigmoid_activation);

$$p(h_j = 1|\mathbf{v}) = \text{sigmoid}\left(c_j + \sum_i v_i w_{ij}\right)$$

3_2_1_1_2 p(h=1|v)を閾値とする乱数

sample[i] = binomial(1, mean[i]);

3_2_1_2 微分式の後項(負値項)をギブスサンプラーで算出CD-k（Constraing Divergency k cycle）

loop ギブスサンプラーの繰返し　k回

当該層の値　p(v|h)　同左の乱数　p(h|v)　同左の乱数

RBM_gibbs_hvh(this, ph_sample, nv_means, nv_samples, nh_means, nh_samples);

3 順伝播学習

3_2_1_2_1 当該層の値 h → 前層の値 v の逆算出

hの値　P(v|h)　p(v|h)の乱数

RBM_sample_v_given_h(this, nh_sample, nv_means, nv_samples);

3_2_1_2_1_1 p(h=1|v)の計算

p(v=1|h)　当該層の値　当該層のバイアス

```
mean[I]=RBM_propdown(this, h0_sample, i, this->vbias[i]);
    ①当該層の値に重み付けし総和する
    loop 当該層のノードの数 j
        pre_sigmoid_activation += this->W[j][i] * h[j];
    ②バイアスを加算する
    pre_sigmoid_activation += b;
    ③シグモイド関数で確率を求める
    return sigmoid(pre_sigmoid_activation);
```

$$p(v_i = 1|\mathbf{h}) = \text{sigmoid}\left(b_i + \sum_j h_j w_{ij}\right)$$

3_2_1_2_1_2 p(h=1|v)を閾値とする乱数

sample[i] = binomial(1, mean[i]);

3_2_1_2_2 層の入力 v → 層の出力 h の順算出

vの値　P(v|h)　p(v|h)の乱数

RBM_sample_h_given_v(this, nv_samples, nh_means, nh_samples);

3_2_1_2_2_1 p(v=1|h)の計算

p(h=1|v)　前層の値　当該層の重み　当該層のバイアス

```
mean[i] = RBM_propup(this, v0_sample, this->W[i], this->hbias[i]);
    ①前層の値に重み付けし総和する
    loop 前層のノードの数 j
        pre_sigmoid_activation += w[j] * v[j];
    ②バイアスを加算する
    pre_sigmoid_activation += b;
    ③シグモイド関数で確率を求める
    return sigmoid(pre_sigmoid_activation);
```

$$p(h_j = 1|\mathbf{v}) = \text{sigmoid}\left(c_j + \sum_i v_i w_{ij}\right)$$

3_2_1_2_2_2 p(v=1|h)を閾値とする乱数

sample[i] = binomial(1, mean[i]);

3_2_1_2 確率的勾配法での重みの更新

loop i 層のノードの数

loop j 前層のノード数

3_2_1_2_1 重みの更新

p(h|v)　v値　p(v,h)　v値(Gibbs)

this->W[i][j] += lr * (ph_mean[i] * input[j] - nh_means[i] * nv_samples[j]) / this->N;

$$\frac{\partial L(D, w_{ij})}{\partial w_{ij}} = -\frac{1}{N}\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}} + \sum_{\mathbf{v},\mathbf{h}} p(\mathbf{v},\mathbf{h})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial w_{ij}}$$
$$= \frac{1}{N}\left\{\sum_h p(h|v_j)h_i v_j - \sum_{ih} p(v_j, h)h_i v_j\right\}$$
$$= \frac{1}{N}\left\{p(h=1|v_j)v_j - p(v_j, h=1)v_j\right\}$$

3_2_1_2_2 バイアス項の更新

p(h|v)　p(v,h)

this->hbias[i] += lr * (ph_means[i] - nh_means[i]) / this->N;

$$\frac{\partial L(D,c)}{\partial c_i} = \frac{1}{N}\left\{p(h=1|v_j) - p(v_j, h=1)\right\}$$

4 逆伝播学習

4 逆伝播学習

訓練データ　教師データ　学習率　学習繰返回数

```
DBN_finetune(&dbn, *train_X, *train_Y, finetune_lr, finetune_epochs);
```

4_1 多クラスロジステック回帰係数を確率勾配法で算出

- loop トレーニング回数
  - loop 訓練データレコード数 N
    - loop 入力層から出力層まで
      - 4_1入力層から当該層の前層まで、層間の連結重みで、当該層への入力値 layer_inputを計算する

4_2 次の関数で出力層のロジット回帰係数を確率的勾配法(学習率 lr)で更新する

前層の値　出力層の教師データ　学習率

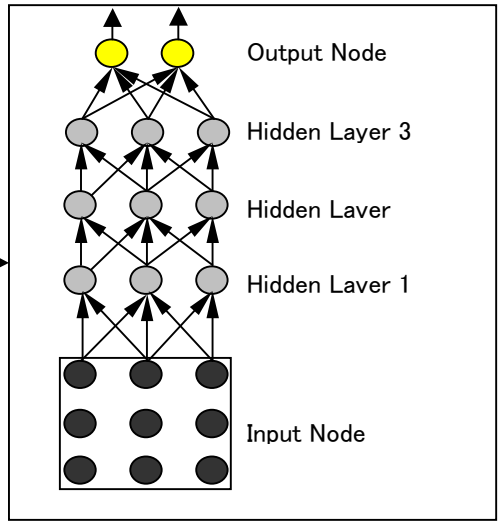
```
LogisticRegression_train(&(this->log_layer), layer_input, train_Y, lr);
```

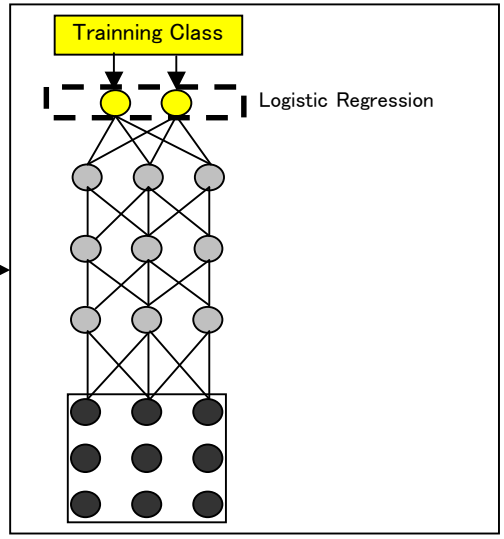
4_3 多クラスロジステック関数で確率計算

前層の値　指数化

```
LogisticRegression_softmax(this, p_y_given_x);
```

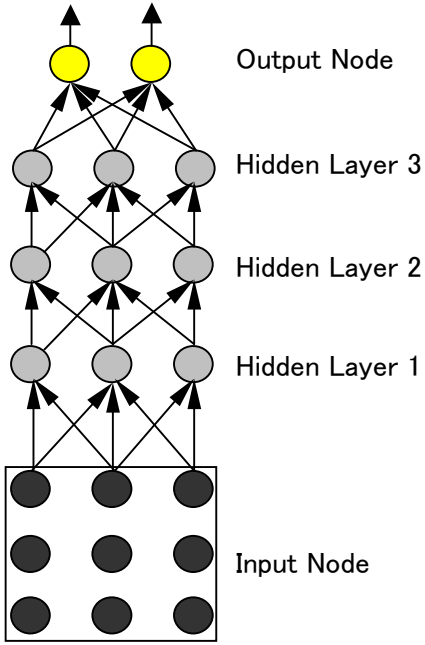
- loop 出力ノード数
  - dy[i] = y[i] − p_y_given_x[i];
  - loop 出力層の前層のノード数
    - ロジット回帰係数の更新
    - this->W[i][j] += lr * dy[i] * x[j] / this->N;
  - 定数項の更新
    - this->b[i] += lr * dy[i] / this->N;



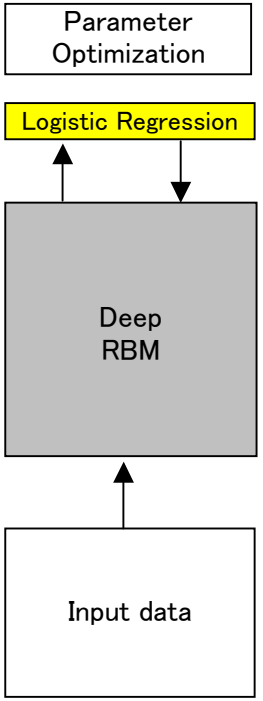


実装結果

Deep Learning for recognition about close or open figure

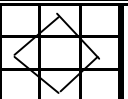
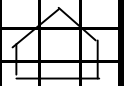
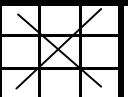
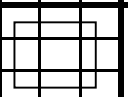
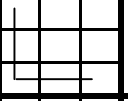
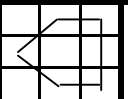
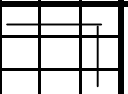
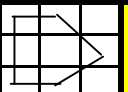
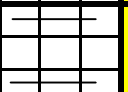
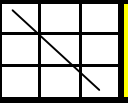


Deep Learning Model



※ RBN：Ristricted Boltzmann Net
CD　：Constrastive Divergence

Test Analysis

data	Input	pattern	close figure	open figure
Train	0 1 0 1 0 1 0 1 0		1	0
	0 1 0 1 0 1 1 1 1		1	0
	1 0 1 0 1 0 1 0 1		0	1
	1 1 1 1 0 1 1 1 1		1	0
	1 0 0 1 0 0 1 1 1		0	1
	0 1 1 1 0 1 0 1 1		1	0
	1 1 1 0 0 1 0 0 1		0	1
Prediction	1 1 0 1 0 1 1 1 0		0.846612	0.153388
	1 1 1 0 0 0 1 1 1		0.316327	0.683673
	1 0 0 0 1 0 0 0 1		0.237903	0.762097

補足資料1　ロジット回帰(SGD版)

download　http://www1.m.jcnnet.jp/mabonki/download.htm

一般にロジット回帰は多次元ニュートン法で解くが、SGD法だと5行でできる。

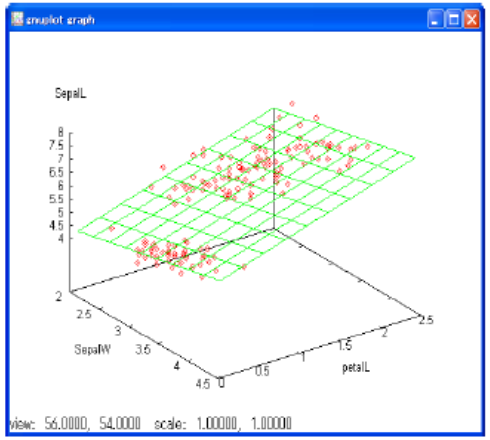
繰返計算

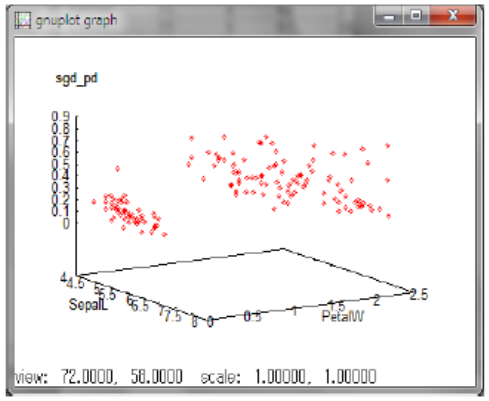
```
for(ii=0;ii<iter;ii++) {

  for(i=0;i<n;i++) {
    sum=w[0];
    for(j=0;j<m-1;j++) {
      sum += w[j+1]*x[i][j];
    }
#ifdef REG
    ww[0] = w[0] + u * (y[i] - sum);
    for(j=0;j<m-1;j++) {
      ww[j+1] = w[j+1] + u*(y[i]-sum)*x[i][j];
    }
#else
    g=(1.0/(1+exp(-sum))) ;
    ww[0] = w[0] + u * (y[i] - g);
    for(j=0;j<m-1;j++) {
      ww[j+1] = w[j+1] + u * (y[i] - g) * x[i][j];
    }
#endif
    /* 入れ替え */
    for(j=0;j<m;j++) {
      w[j]=ww[j];
    }
}
```

重回帰計算部分

ロジット回帰計算部分

重みの更新　教師区分　説明変数

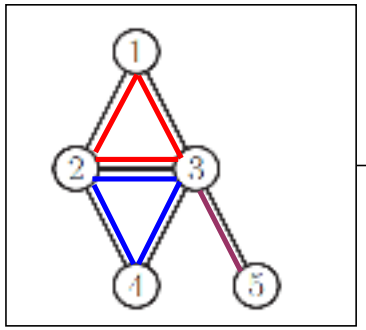




一般ボルツマンの学習　補足2

一般ボルツマンマシン

ノードとその連結（無方向）があり、ノード間の結合を**クリーク**単位で分割します。
クリーク内はボルツマン分布に従うと仮定した場合、全ノードの確率分布を計算します。



$$p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\psi_{123}(y_1,y_2,y_3)\psi_{234}(y_2,y_3,y_4)\psi_{35}(y_3,y_5)$$

$$Z = \sum_{\mathbf{y}} \psi_{123}(y_1,y_2,y_3)\psi_{234}(y_2,y_3,y_4)\psi_{35}(y_3,y_5)$$

クリーク内はΦ(y)とパラメータθの線形結合　注）Φはyの特徴関数(カーネル等)

$$\log \psi_c(\mathbf{y}_c) \triangleq \phi_c(\mathbf{y}_c)^T \boldsymbol{\theta}_c$$

全ノードのボルツマン確率分布

$$p(\mathbf{y}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\exp\left(\sum_c \boldsymbol{\theta}_c^T \phi_c(\mathbf{y})\right)$$

隠れ値 h がある場合のボルツマン確率分布

$$p(\mathbf{y},\mathbf{h}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})}\exp\left(\sum_c \boldsymbol{\theta}_c^T \phi_c(\mathbf{h},\mathbf{y})\right)$$

一般ボルツマンマシンのパラメータ推定

各ノード値のデータがN件ある場合、そのデータに合うパラメータを推定します。
推定には最大対数尤度を計算します。
最大値の算出には、対数尤度をパラメータで微分して、最大解を求めます。
ここでは確率的降下法を使ったアルゴリズムを示します。

$$\ell(\boldsymbol{\theta}) \triangleq \frac{1}{N}\sum_i \log p(\mathbf{y}_i|\boldsymbol{\theta}) = \frac{1}{N}\sum_i \left[\sum_c \boldsymbol{\theta}_c^T \phi_c(\mathbf{y}_i) - \log Z(\boldsymbol{\theta})\right]$$

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \frac{1}{N}\sum_i \left[\phi_c(\mathbf{y}_i) - \frac{\partial}{\partial \boldsymbol{\theta}_c}\log Z(\boldsymbol{\theta})\right]$$

$$\frac{\partial \log Z(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_c} = \mathbb{E}\left[\phi_c(\mathbf{y})|\boldsymbol{\theta}\right] = \sum_{\mathbf{y}} \phi_c(\mathbf{y})p(\mathbf{y}|\boldsymbol{\theta})$$

$$\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \left[\frac{1}{N}\sum_i \phi_c(\mathbf{y}_i)\right] - \mathbb{E}\left[\phi_c(\mathbf{y})\right]$$

隠れ値　h がある場合の対数尤度の微分式

$$\boxed{\frac{\partial \ell}{\partial \boldsymbol{\theta}_c} = \frac{1}{N}\sum_i \left\{\mathbb{E}\left[\phi_c(\mathbf{h},\mathbf{y}_i)|\boldsymbol{\theta}\right] - \mathbb{E}\left[\phi_c(\mathbf{h},\mathbf{y})|\boldsymbol{\theta}\right]\right\}}$$

確率的降下法によるボルツマンマシンの学習

Algorithm 19.1: Stochastic maximum likelihood for fitting an MRF

```
Initialize weights θ randomly;
k = 0, η = 1 ;
for each epoch do
    for each minibatch of size B do
        for each sample s = 1 : S do
            Sample y^{s,k} ~ p(y|θ_k) ;
        Ê(φ(y)) = (1/S) Σ_{s=1}^{S} φ(y^{s,k});
        for each training case i in minibatch do
            g_ik = φ(y_i) − Ê(φ(y)) ;
        g_k = (1/B) Σ_{i∈B} g_ik;
        θ_{k+1} = θ_k − η g_k;
        k = k + 1;
        Decrease step size η;
```