ディープラーニングライブラリ Mocafe開発記

TOMOKAZU MORIYAMA

【開発目的と開発対象】

1．ディープラーニングのアルゴリズムを
　　ゼロから実装して体得する

2．画像認識の基礎を体感する

ディープラーニングライブラリを
プロトタイプ開発

第１弾

☆自己符号化器

☆ディープボルツマンマシン

【陣容と開発技術】

１．陣容

森山１人 (開発2016年4月～8月＋試験9月)

２．開発技術

Ｃ／Ｃ＋＋11 (標準ライブラリのみ)

・Gnu C／C++4.9
(gnu/g++コマンド＋標準ライブラリ)
・VC++2013 (clコマンド＋標準ライブラリ)
・CUDA7.5 (nvccコマンド＋標準ライブラリ)
・MinGW (gnu/g++コマンド＋標準ライブラリ)
・Makefile
・bashスクリプト／Microsoft bat形式

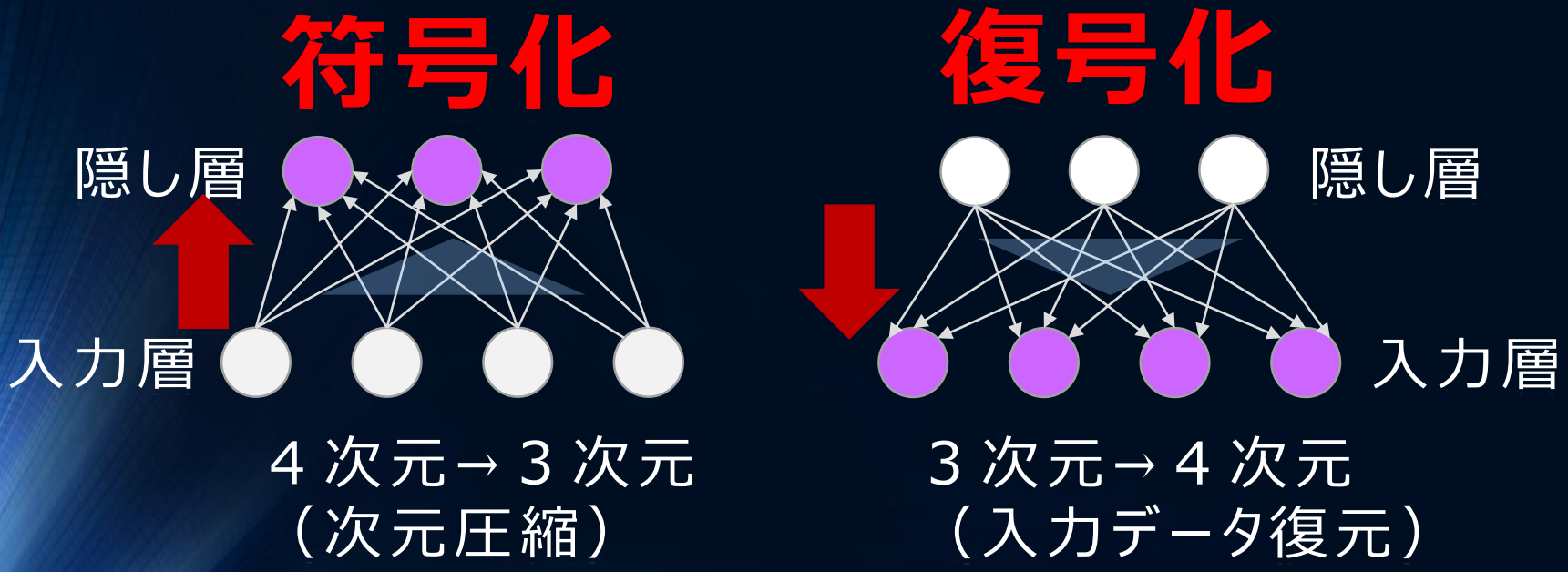
【陣容と開発技術】

３．動作環境

- x86／64環境
- NVIDIA GPU (K40･･･sm_35)
- Windows7／Windows10
- CentOS7
- bashシェル／ターミナル
- Microsoft コマンドプロンプト

【自己符号化器とは？】

入力層から隠し層に出力(符号化)し
隠し層から入力層に出力(復号化)した際に
元の入力データを復元するように
ネットワークを訓練する方法を使用したニューラルネット

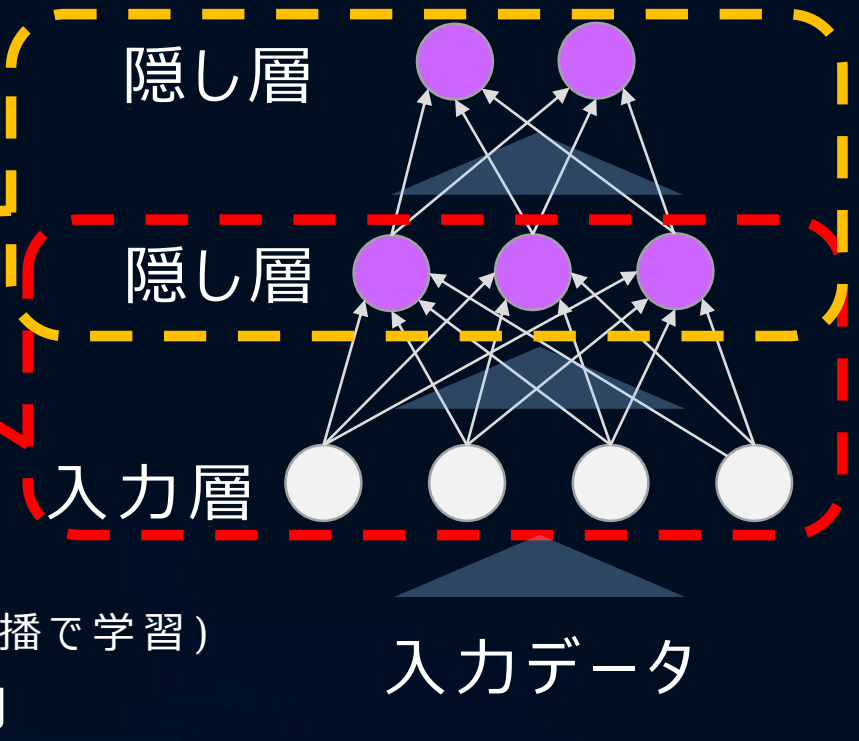


元の入力との誤差の最小化

【自己符号化のメリット】

ニューラルネットの事前学習にて隠し層ごとに学習し
ネットワークパラメータの良い初期値を得ることができる

層ごとに
自己符号化で
学習

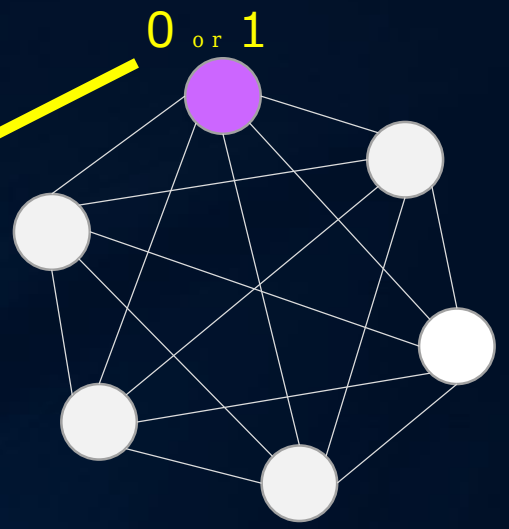


自己符号化は事前学習
(自己符号化器はその後に誤差逆伝播で学習)

教師なし学習

【ボルツマンマシンとは？】

ニューラルネットのユニットの値を２値(0or1)とし
確率を使用して状態を決定するマシン



その他のユニットの状態を元に
確率を使用して値を決定

【制約ボルツマンマシンとは？】

隠れ変数をもつボルツマンマシン、ユニット間結合に制約

ニューラルネットの隠し層の値を２値(0or1)とし
順伝播にて各層の出力の際に確率を使用

同一層同士で
ユニット間の
結合なし

$$z = f(\Sigma wx + b)$$

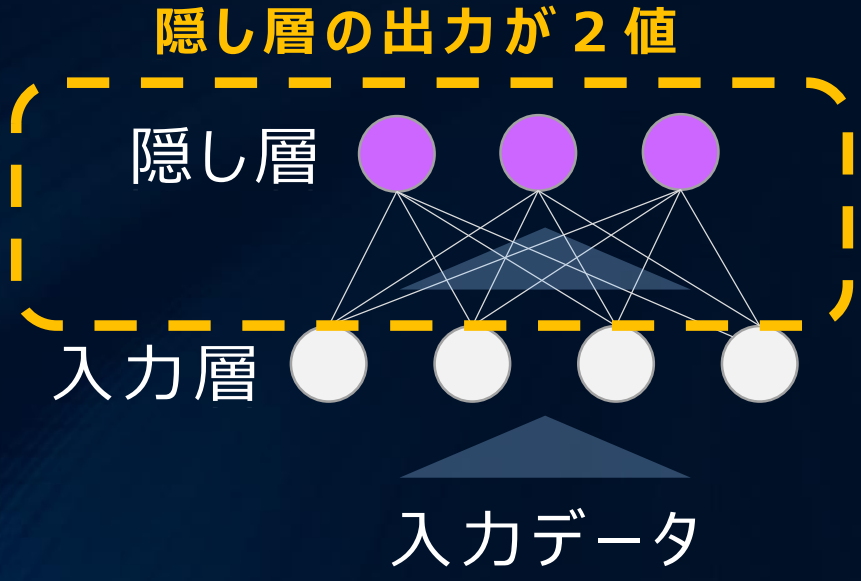
x：前層のユニットの出力

w：ユニットを接続するコネクタの重み

b：バイアス値

f()：活性化関数　・・・シグモイド関数　etc…

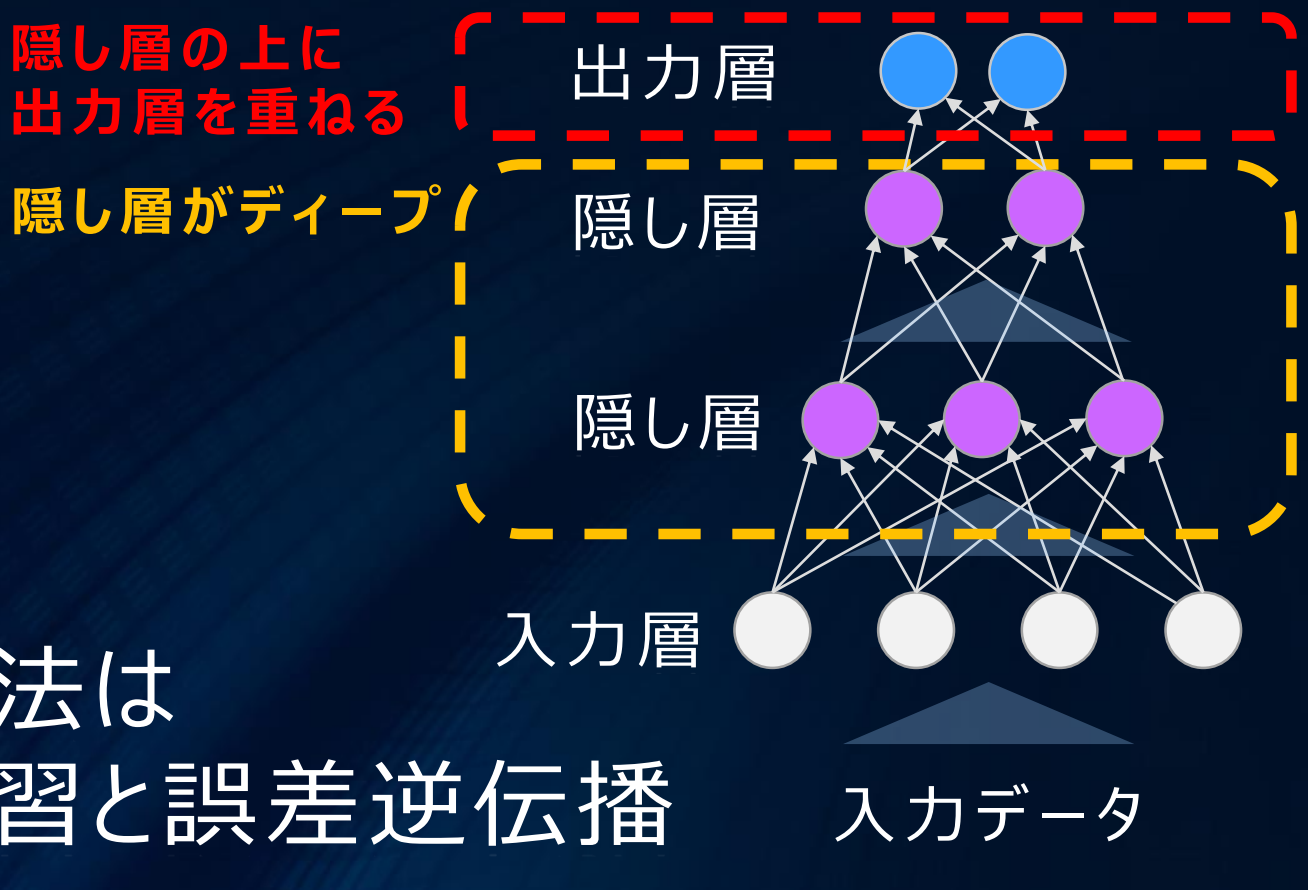
z：ユニット出力を決定する確率値



０～１のランダム値を使用して
確率値を超えるか否かで出力を決定

【ディープボルツマンマシンとは？】

制約ボルツマンマシンをディープに積み重ね
最上層に出力層を重ねたもの



学習方法は
事前学習と誤差逆伝播

【ボルツマンマシンの事前学習】

手法(教師なし学習)

１．ギブスサンプリング

入力層の値に0～1の**ランダム値**を使用

ｖを入力層、ｈを隠し層とすると

$$v_0 \rightarrow h_0 \rightarrow v_1 \rightarrow h_1 \rightarrow v_2 \rightarrow h_2 \cdots v_T \rightarrow h_T$$

とサンプリングを繰り返し

自己符号化と同様に**誤差を最小化**するように
ネットワークパラメータを修正

２．コントラスティブ・ダイバージェンス

入力層の値に**実際の訓練データ**を使用

（その他はギブスサンプリングと同様）→反復回数が少なくて済む

【誤差逆伝播の概要】

手法(教師あり学習)

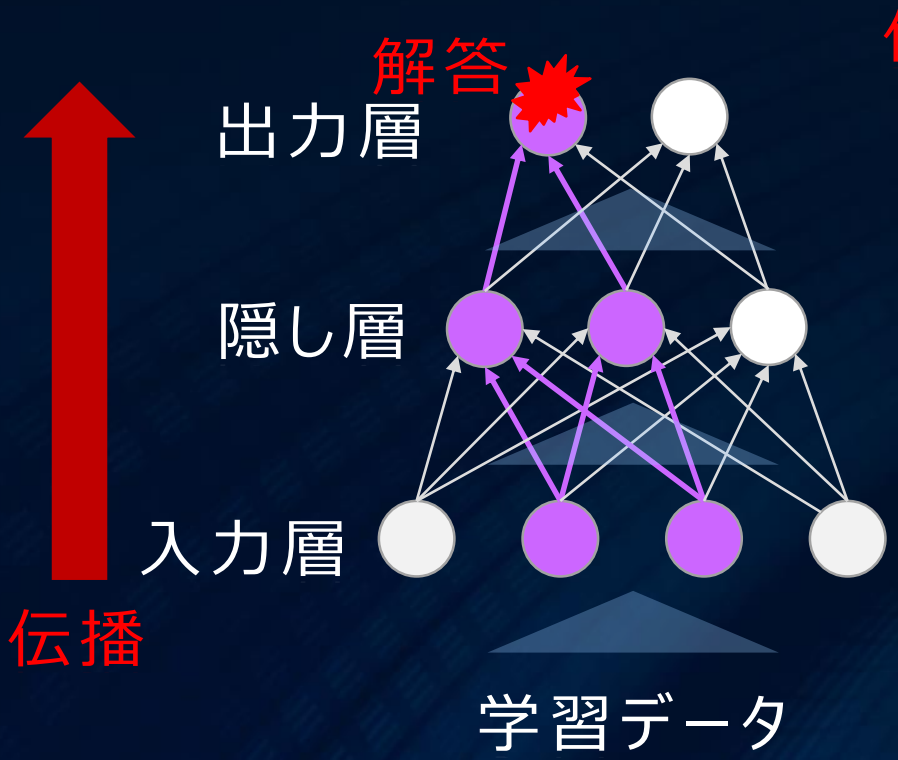
ニューラルネットに共通の学習方法
(自己符号化器、ディープボルツマンマシンに共通)

正解ラベル付きの学習データを使用し
順伝播した結果の出力と正解ラベルを比較

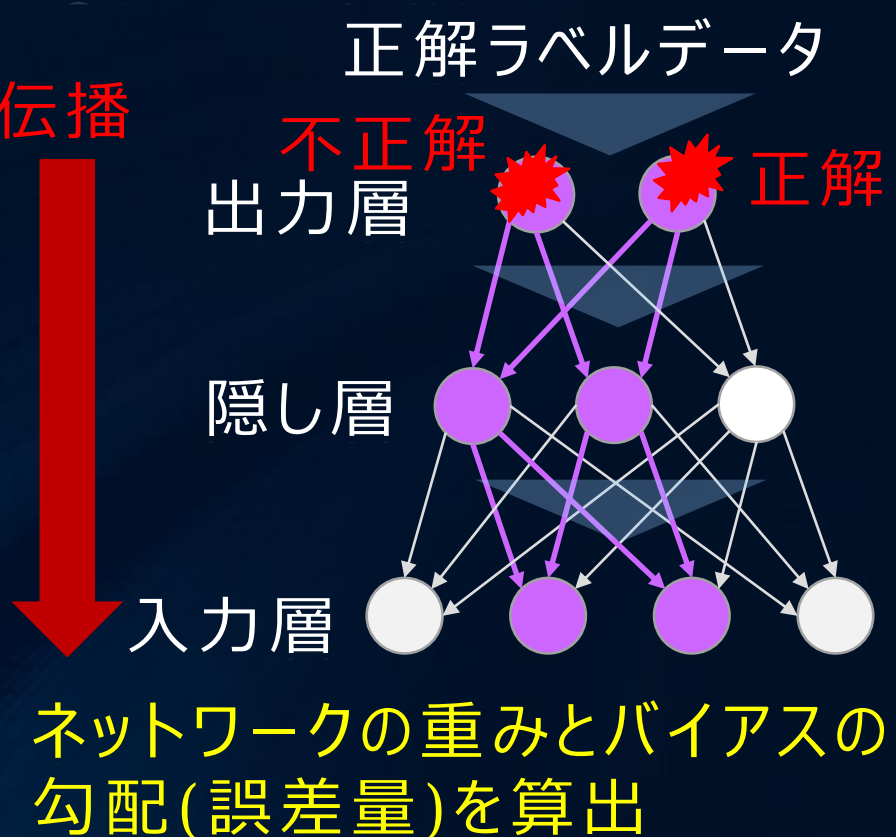
正解と実際の出力の差分を誤差として
ネットワークを逆伝播して各パラメータの誤差を求め
誤差量を元にネットワークパラメータを更新

【誤差逆伝播の手順】

①順伝播



②誤差逆伝播



重み量、バイアス量更新

【ネットワーク構成パラメータ】

１．共通
- 各層のユニット数
- 隠し層の層数
- 活性化関数
- ネットワークパラメータ初期値倍率
- 入力層値倍率
- 出力層の回帰／二値分類／多クラス分類
- 使用GPU番号

２．自己符号化器
- 自己符号化器の種類(現時点ではノーマルのみ)

３．ディープボルツマンマシン
- ディープボルツマンマシンの種類
  ガウシアンベルヌーイ、ベルヌーイ、RELU

【ネットワーク学習パラメータ】

１．共通

- epocs数(全体の学習回数)
- 学習回数(1epocsごとの回数)・・・事前学習、誤差逆伝播個別指定
- 学習率(固定値、AdaGrad、RMSProp、AdaDelta、Adam)
  ・・事前学習、誤差逆伝播個別指定
- モーメンタム率
- ドロップアウト率
- 重み減衰率
- 重み上限値
- スパース正則化値
- 学習サンプリング方法(オンライン、ミニバッチ、バッチ、SGD)

２．ディープボルツマンマシンのみ

- 事前学習方法／サンプリング回数、繰り返し回数
  (ギブスサンプリング、コントラスティブ・ダイバージェンス、
  持続的コントラスティブダイバージェンス)

【性能試験】

１．試験環境

- Windows7(64bit)
- マシン HP Compaq8200 Elite USDT PC
- CPU Intel i5-2400S 2.5GHz
- メモリ 4G Byte
- Microsoft コマンドプロンプト

２．試験内容

MNIST(手書き数字画像)データセットによる検証

学習データ60000件、交差検証データ10000件
手書き数字画像と０～９までの数字の正解ラベルを
使用して学習、検証を行った(最高正解率の結果のみ表示)

【性能試験】

３．試験結果

アルゴリズム／ネットワークパラメータごとの試験結果（正解率、処理時間）

アルゴリズム	層数	隠し層Unit数	活性化関数	学習回数	学習率	サンプリング方法	正解率(時間)
開発したAutoencoder	3	500	RELU	100	0.001	online	98.17%（95時間54分）
Theanets (Autoencoder)	3	500	RELU	100	0.001	online	89.00%（２時間１８分）
開発したDBM	3	500	TANH	100	0.001	online	97.84%（103時間20分）
(同上)	3	100	SIGMOID	100	0.001	online	95.44%（２８時間３８分）
Scikit-learn (DBM)	3	100	SIGMOID	100	0.001	online	90.08%（３時間４０分）

（時間はＣＰＵのみを使用したＰＣでの計測値）

【今後の改善について】

１．現時点ではGPUを使用した方が遅い
　　・・・速度改善をすべき

２．学習状況が分かりづらい
　　・・・学習状況の可視化をすべき

３．未実装の基本的なアルゴリズムが存在
　　・・・アルゴリズムを追加実装すべき

第２弾
- ☆畳込みニューラルネット
- ☆再帰ニューラルネット
- ☆マックスアウト自己符号化器