現場で使えるディープラーニング基礎講座

DAY4

SkillUP AI

前回の復習

- 前回は何を学びましたか？

学習の最適化 後半

目次

1. バッチ正規化とその類似手法
2. ドロップアウト
3. 荷重減衰

正則化

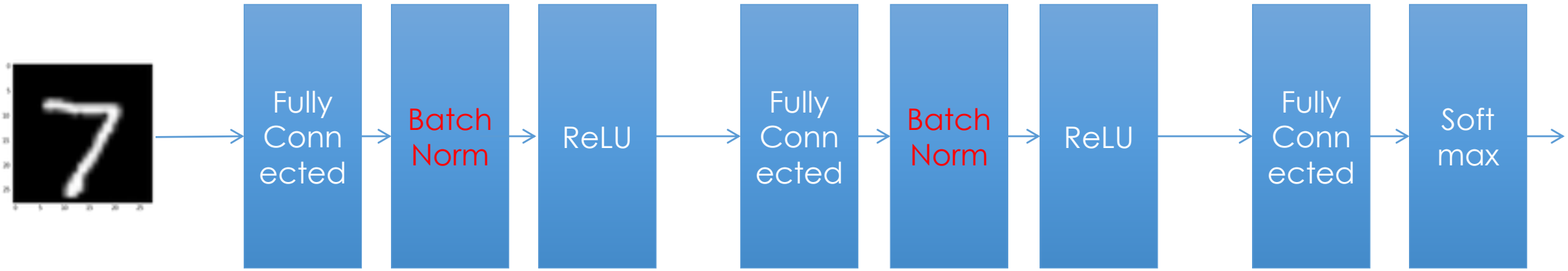
- 機械学習では、汎化性能の高いモデルが求められる。
- パラメータに制限をつけることで過学習を抑制する方法を正則化(regularization)という。
- 過学習が起きやすい条件
  - パラメータが多い。
  - 訓練データが少ない。

バッチ正規化とその類似手法

バッチ正規化(batch normalization)

- バッチ正規化とは各層のアクティベーション分布を適度な広がりを持つように調整する手法。活性化関数の手前に設置されることが多い。
- 学習の進行が速くなることが期待できる。
- 初期値に神経質にならなくてもよくなる。
- 過学習を抑制する効果がある。(ドロップアウトを使わなくてもよくなる)
- 原著論文：https://arxiv.org/pdf/1502.03167.pdf

バッチ正規化を含む全結合層NNの例

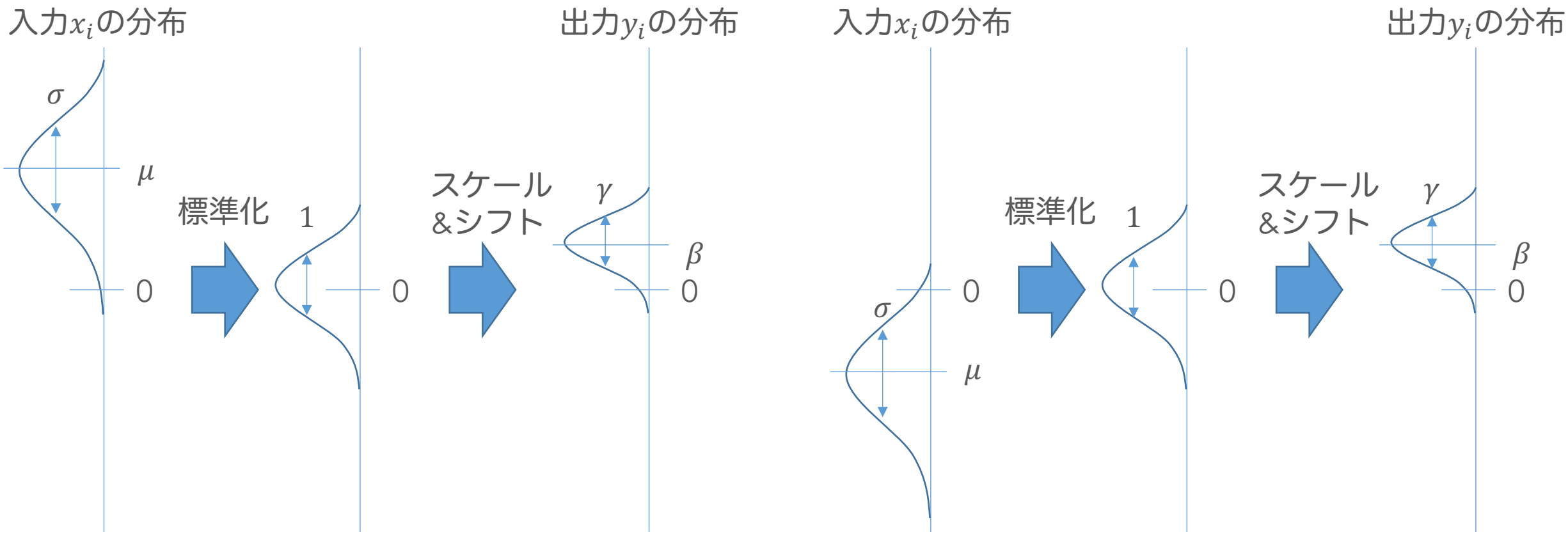


内部共変量シフト

- 共変量とは、なんらかのモデルに入力されるデータのこと。いわゆる説明変数のこと。
- 共変量シフトとは、与えられた入力に対する出力の生成規則は訓練時とテスト時で変わらないが、入力 (共変量) の分布が訓練時とテスト時で異なるという状況のこと。非定常データでよく観察される現象。
- ニューラルネットにおいては層間でもこれが起こる。あるL層の出力はL+1層の入力になるが、この入力は学習の過程において、重みが更新されるたびに全体的に移動する。これを内部共変量シフトという。
- バッチ正規化を用いると、この内部共変量シフトを抑えることができる。

バッチ正規化の概念図

- バッチ正規化の概念図を以下に示す。



入力x_iの分布がミニバッチ毎に異なっていても、同じ分布になるように変換される。

バッチ正規化の計算手順

バッチ正規化(batch normalization)レイヤの計算手順

ここでの説明は、バッチ正規化レイヤの入力層側に設置されている層の出力値のうちの1ノード分を対象にする。

[計算手順(学習時の順伝播計算)]

(1) 計算の対象をxとする

入力 : $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$
n : データ数=バッチサイズ

(2) 入力の平均値を求める

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

(3) 入力の分散を求める

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

(4) 入力を標準化する

各入力値について以下の処理を行う。numpyで計算する場合はベクトルとスカラーの演算が可能。

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$\epsilon : 1e-8$ (深層学習, Goodfellow, p.229)

(5) スケールし、平行移動させる

各入力値について以下の処理を行う。numpyで計算する場合はベクトルとスカラーの演算が可能。

$y_i = \gamma\hat{x}_i + \beta$

y_iが返り値になる。

γとβは、標準化されたxの分布を最適な分布に変換するための係数であり、学習の過程で最適化されていくパラメータ。

1つのミニバッチ内で計算される平均μと分散σ^2とは値が異なる。

バッチ正規化の計算

[計算手順(予測時の順伝播計算)]

基本的には、学習時の順伝播計算と同じだが、μとσ^2は、学習時に求めた移動平均値を使う

[計算手順(学習時の逆伝播計算)]

スライドの計算グラフを参照

[参考]

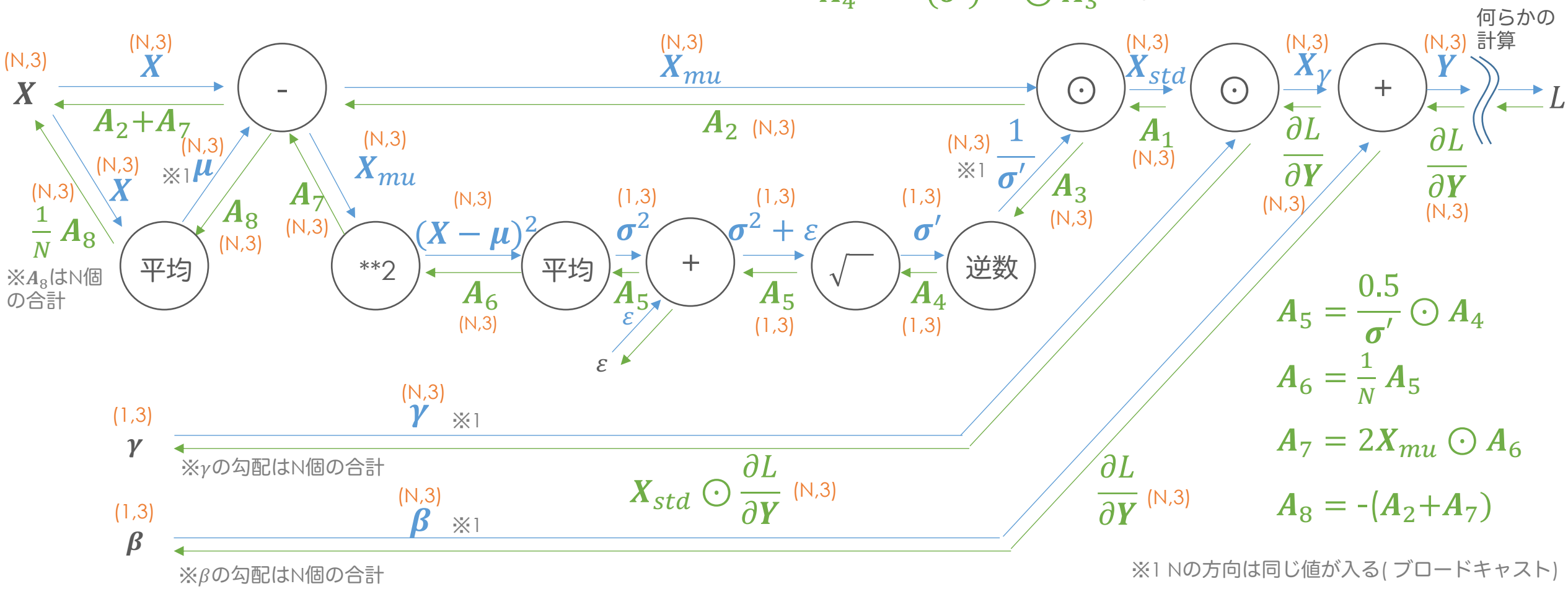
- 原著論文
  - https://arxiv.org/pdf/1502.03167.pdf
- ブログ
  - https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html

バッチ正規化の計算グラフ

- バッチ正規化の計算グラフを以下に示す。
  (入力が3次元、データ数がN次元の場合)

$$A_1 = \gamma \odot \frac{\partial L}{\partial Y}$$
$$A_2 = \frac{1}{\sigma'} \odot A_1$$
$$A_3 = X_{mu} \odot A_1$$
$$A_4 = -(\sigma')^{-2} \odot A_3 \quad ※A_3はN個の合計$$

$$X_{mu} = X - \mu$$
$$X_{std} = X_{mu} \odot \frac{1}{\sigma'}$$
$$X_\gamma = \gamma \odot X_{std}$$
$$Y = X_\gamma + \beta$$



$$A_5 = \frac{0.5}{\sigma'} \odot A_4$$
$$A_6 = \frac{1}{N} A_5$$
$$A_7 = 2X_{mu} \odot A_6$$
$$A_8 = -(A_2 + A_7)$$

※1 Nの方向は同じ値が入る(ブロードキャスト)

[演習] バッチ正規化(batch normalization)レイヤの実装

- 4_1_batch_normalization_layer_trainee.ipynb
  - バッチ正規化レイヤを実装しましょう。
- 4_2_batch_norm_test.ipynb
  - MNISTデータを用いて、バッチ正規化の効果を確認しましょう。

バッチ正規化とその類似手法

- バッチ正規化が登場して以降、その類似手法がいくつか提案されている。
- ここでは、以下の3つを紹介する。
- レイヤー正規化(layer normalization)
  - Jimmy Lei Ba , Jamie Ryan Kiros , Geoffrey E. Hinton. Layer Normalization. https://arxiv.org/pdf/1607.06450.pdf
- インスタンス正規化(instance normalization)
  - Dmitry Ulyanov, Andrea Vedaldi, Victor Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. https://arxiv.org/pdf/1607.08022.pdf
- グループ正規化(group normalization)
  - Yuxin Wu, Kaiming He. Group Normalization. https://arxiv.org/pdf/1803.08494.pdf

バッチ正規化とその類似手法

- レイヤー正規化
  - レイヤー内で正規化する。
  - 原著論文では、RNNへの適応方法も提案されている。
- インスタンス正規化
  - インスタンス(1チャンネル分の画像)毎に正規化する。
  - 1チャンネル分の画像で正規化することにより、画像のコントラストを取り除くことができる。
- グループ正規化
  - チャンネルをグループ化し、正規化する。
  - バッチ正規化は、バッチサイズを小さくすると、バッチ統計量(平均と分散) の推定精度が落ち、学習がうまくいかない。グループ正規化では、これが改善されている。

バッチ正規化とその類似手法

- 各手法は、正規化の対象が異なる。

全てCNNを想定

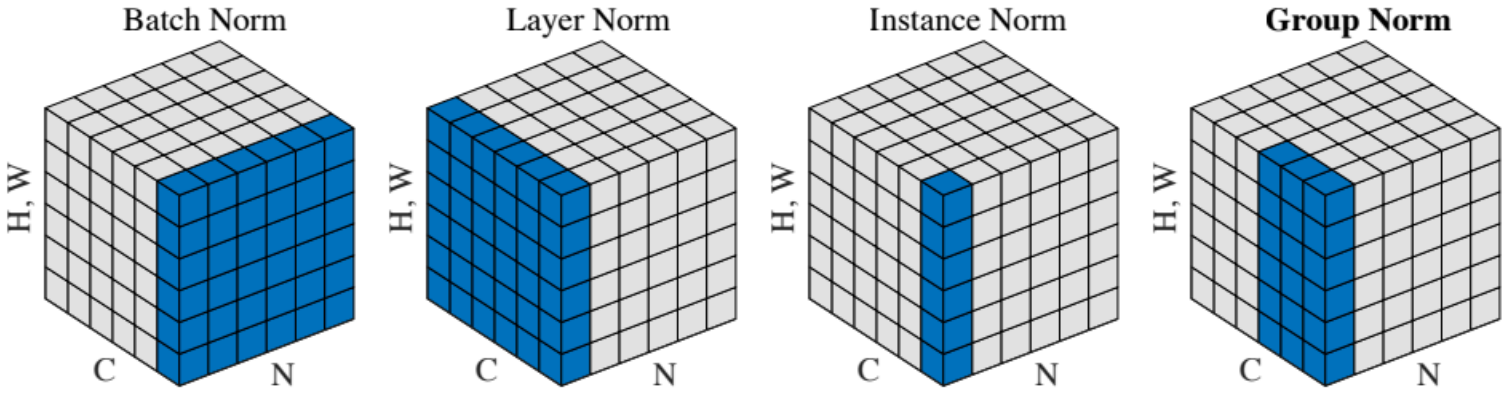


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

引用元： https://arxiv.org/pdf/1803.08494.pdf

バッチ正規化とその類似手法

- 性能比較結果の例を以下に示す。

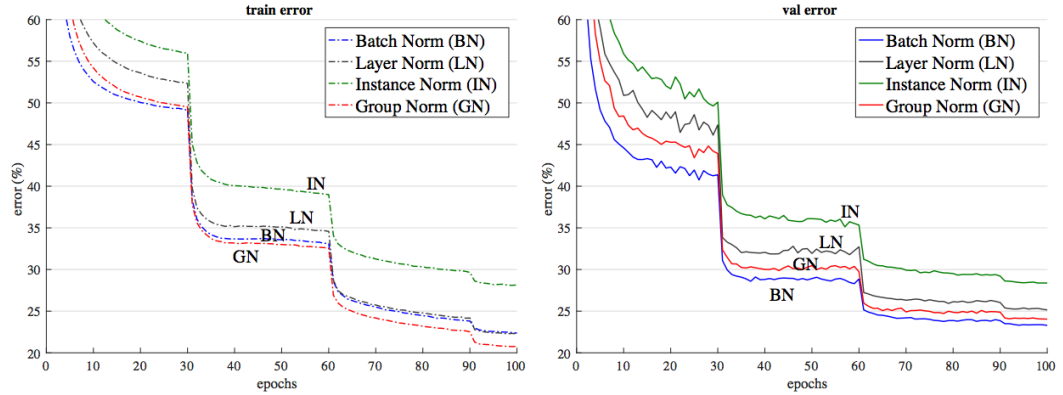


Figure 4. **Comparison of error curves** with a batch size of **32 images**/GPU. We show the ImageNet training error (left) and validation error (right) *vs*. numbers of training epochs. The model is ResNet-50.

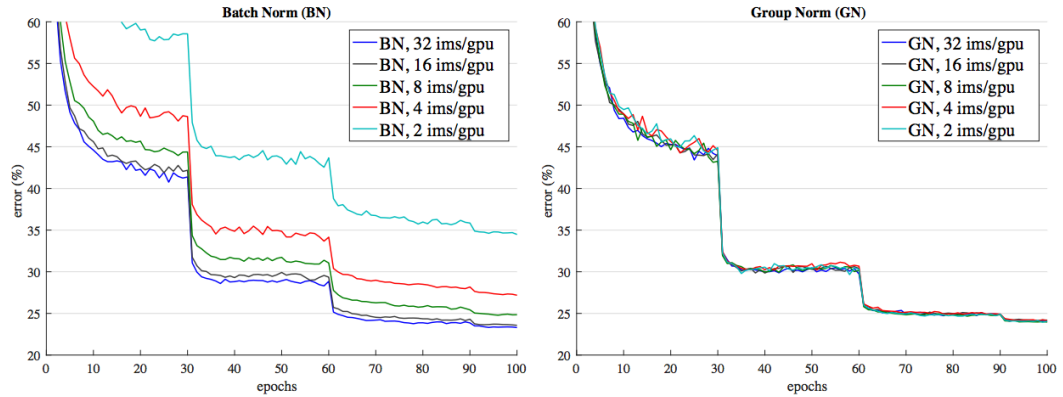


Figure 5. **Sensitivity to batch sizes**: ResNet-50's validation error of BN (left) and GN (right) trained with 32, 16, 8, 4, and 2 images/GPU.

グループ正規化(Group Norm)は、バッチ正規化(Batch Norm)と同等の精度がでており、また、バッチサイズの影響をほとんど受けていないことがわかる

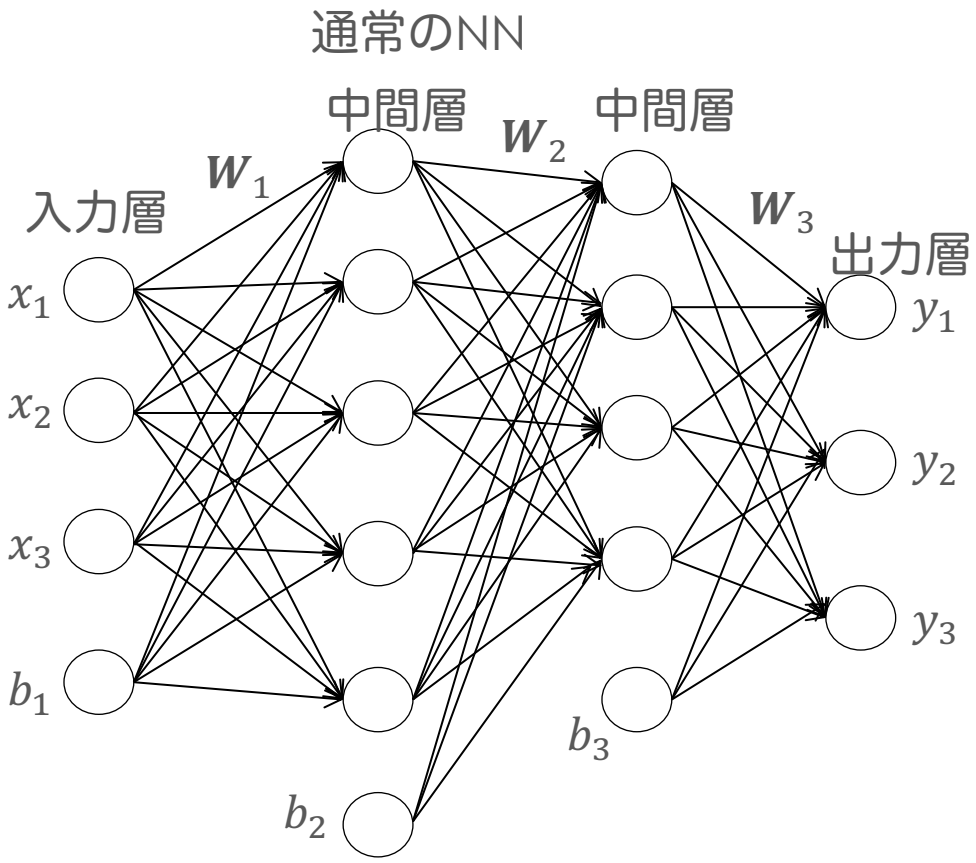
引用元：
https://arxiv.org/pdf/1803.08494.pdf

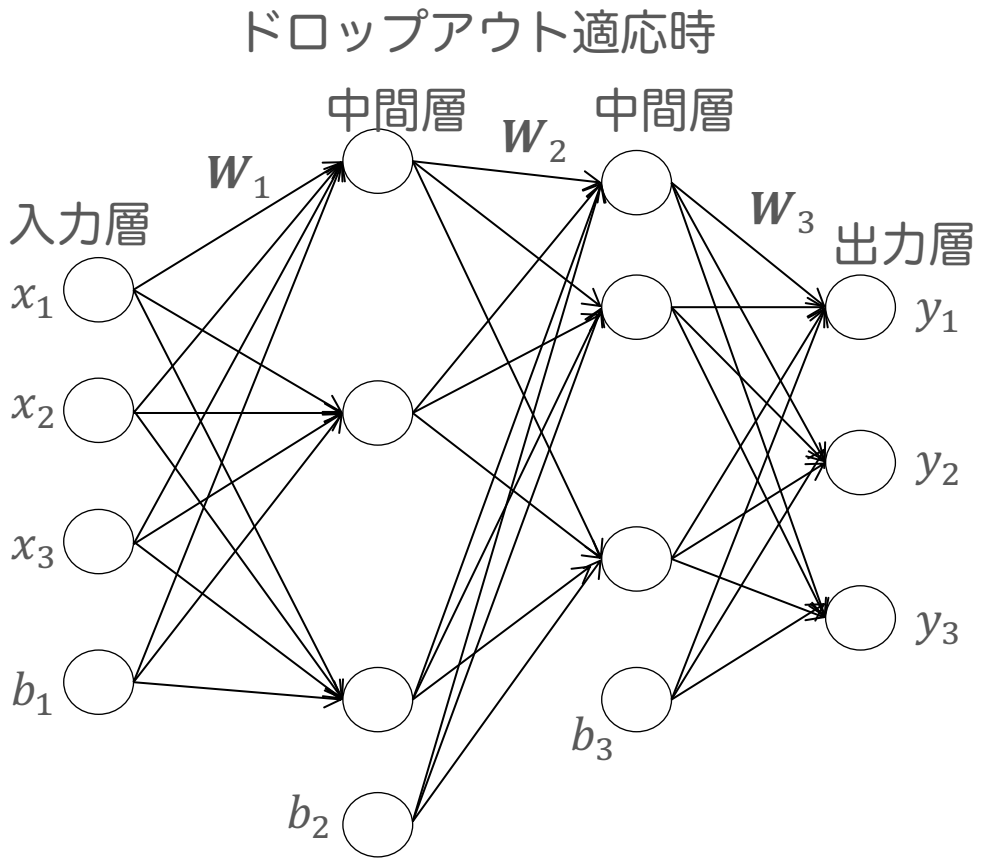
Any Questions?

ドロップアウト

ドロップアウト

- ドロップアウト(dropout)とは、ノードをランダムに消去しながら学習する方法。
- 過学習を抑制する効果がある。





ドロップアウト(dropout)の計算

- 学習時
  - ミニバッチ ごとに、消去するノードをランダムに選ぶ。
  - 順伝播計算と逆伝播計算を行う。
  - 順伝播計算で信号を通さなかったノードは、逆伝播でも信号を通さない。
- 予測時
  - すべてのノードを採用し、すべての重みにドロップアウトしなかった割合(1 – dropout_ratio)を一律に掛ける。
  - すべてのノードを採用すると、学習時に比べ結合後の値が大きくなるので、消去した割合を掛けることにより、値を調整している。

[演習] ドロップアウト(dropout)の実装

- 4_3_dropout_trainee.ipynb
  - ドロップアウト(dropout)を実装しましょう。

Any Questions?

荷重減衰

荷重減衰

- 荷重減衰(weight decay)は、過学習を抑制する方法の一つ。
- 重みの値が大きくなることにペナルティを与える。
- 重みの値が大きくなることによって過学習が起きることが多いので、その重みの値が大きくならないように制限する。
- 荷重減衰では、損失関数に全ての層の$\frac{1}{2}\lambda w_{ij}^2$を加える。いわゆるL2ノルムのこと。
- これにより、重みは、自身の大きさに比例したペナルティが加わるため、重みの値が小さくなっていく(減衰していく)。
- λはハイパーパラメータで、一般に、0.01~0.00001程度の範囲から選ぶ。
- この荷重減衰は、 wにだけ適応し、バイアスbには適応しないことが多い。バイアスbは総数が少ないため過適合を起こしにくいことと、ときに大きな値をとる必要があるため。

荷重減衰の計算式

損失関数

$$L = L' + \frac{1}{2}\lambda \sum_{l=1}^{layers} \sum_{i,\,j} (w_{ij}^{l})^2$$

$w_{i,j}^{l}$: l層目の重み行列$\mathbf{W}$のi,j成分
L : 正則化項を加えた後の損失
L' : 正則化項を加える前の損失
$layers$: 層番号
λ : 係数

重みの更新(SGDの場合)

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta(\frac{\partial L'}{\partial \mathbf{W}_t} + \lambda \mathbf{W}_t)$$

$\mathbf{W}$: 重み行列
L' : 正則化項を加える前の損失
η : 学習率
λ : 係数

$\boldsymbol{W}_t$が正の場合、 $\boldsymbol{\lambda W}_t$が引かれるため
$\boldsymbol{W}_{t+1}$はより0に近づく。

$\boldsymbol{W}_t$が負の場合、 $\boldsymbol{\lambda W}_t$が加算されるため
$\boldsymbol{W}_{t+1}$はより0に近づく。

実装する際は、上記の更新を勾配の補正によって考慮するのが簡単でよい

勾配の補正

$$\frac{\partial L}{\partial \mathbf{W}_t} = \frac{\partial L'}{\partial \mathbf{W}_t} + \lambda \mathbf{W}_t$$

$\mathbf{W}$: 重み行列
L' : 正則化項を加える前の損失
λ : 係数

参考：荷重減衰(weight decay)の仕組み

- 荷重減衰では、通常の損失関数に比べ、全ての重みが0に近づくように計算が進む。この時、仮に全ての重みが0になってしまうと、どんな入力を入れても出力が常にゼロになる意味のないモデルができあがってしまう。よって、荷重減衰で求めたい重みは、荷重減衰を適応しなかった場合と0との間になるはず。これは損失関数の式で考慮される。
- 荷重減衰の損失関数をみてみる。右辺第一項は通常の損失関数であり、これを基準に考える。右辺第二項は、重みの値がより小さくなるように働く。この時、重みが小さくなりすぎると、逆に右辺第一項の損失（例えば、2乗和誤差）が大きくなる。つまり、右辺第一項と右辺第二項のバランスによって、最終的な重みが決定されることになる。
- 入力データが標準化されている場合、最終的に決定された重みのうち、絶対値の大きな重みは出力に対する影響度が大きいものであり、逆に絶対値の小さな重みは出力に対する影響度が小さいものであると解釈することができる。

[演習] 荷重減衰に対応したNNの実装

- 4_4_weight_decay_trainee.ipynb
  - 荷重減衰に対応したNNを実装しましょう。

参考：早期終了

- 早期終了(early stopping)とは、一定のルールを満たせば学習を終了させるという方法。
- 過学習を抑える効果がある。
- 早期終了の計算手順例
  - 学習の各ステップにおいて、テスト用データを用いて検証誤差(test_loss)を算出する。
  - 検証誤差が前ステップの検証誤差よりも大きければ、1カウントする。
  - このカウントが指定回数以上になれば、学習を終了させる。
  - ここでの指定回数はハイパーパラメータ。

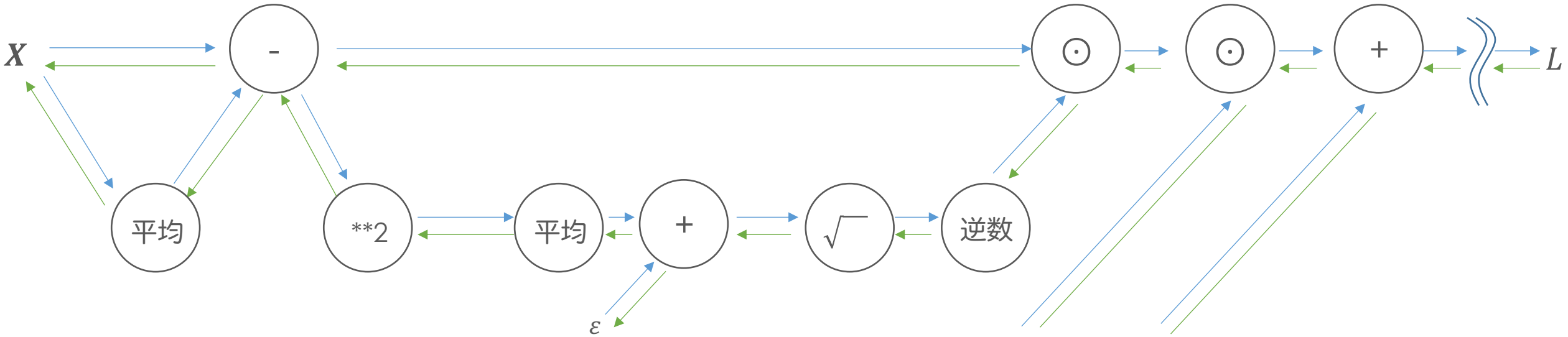
Any Questions?

[グループワーク] 荷重減衰適応時の計算グラフ

- 荷重減衰を考慮したニューラルネットワークの計算グラフを描いてみましょう。
- 2~3名のグループに分かれて、上記課題に取り組みましょう。(30分)
- 最後に、グループごとに発表していただきます。 (15分)

[グループワーク] バッチ正規化の計算グラフ

- 以下のバッチ正規化の計算グラフを完成させましょう。
- 「バッチ正規化の計算グラフ」のページが解答例になりますが、なるべくこのページを見ないで取り組みましょう。
- 2~3名のグループに分かれて、上記課題に取り組みましょう。(45分)
- 最後に、グループごとに発表していただきます。 (15分)



Any Questions?

ディープラーニングの様々なモデル

ディープラーニングの様々なモデル

確定的

確率的

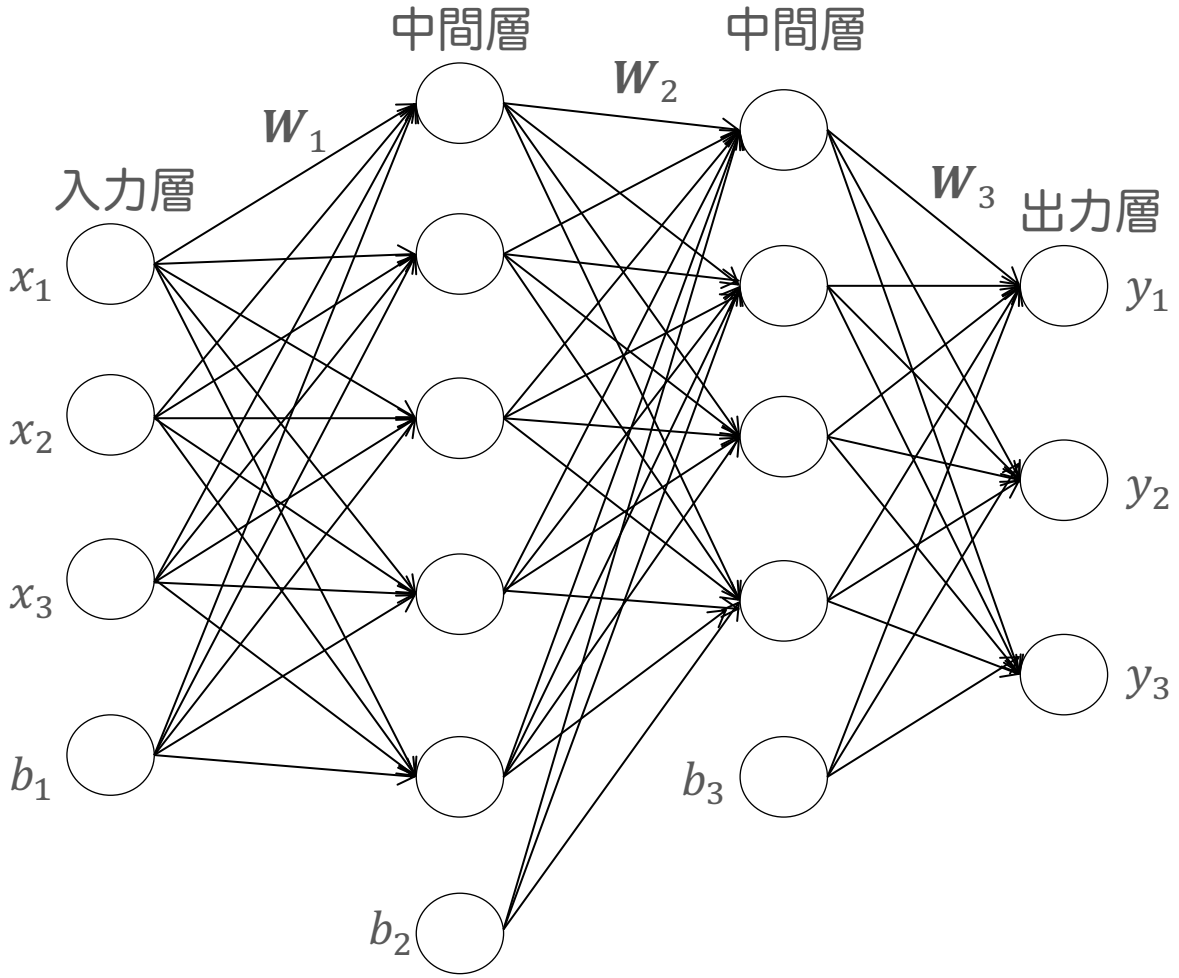
階層型	自己符号化器型	ボルツマンマシン型
全結合型ニューラルネットワーク fully connected neural network	自己符号化器 autoencoder	ボルツマンマシン Boltzmann machine
畳み込みニューラルネットワーク convolutional neural network	雑音除去自己符号化器 denoising autoencoder	制約ボルツマンマシン restricted Boltzmann machine
再帰型ニューラルネットワーク recurrent neural network	変分自己符号化器 variational autoencoder	深層信念ネットワーク deep belief network
	スパース自己符号化器 sparse autoencoder	深層ボルツマンマシン deep Boltzmann machine

参考：『深層学習』(神嶌など、近代科学社)

全結合型ニューラルネットワークの計算グラフ

- いわゆる普通のニューラルネットワーク

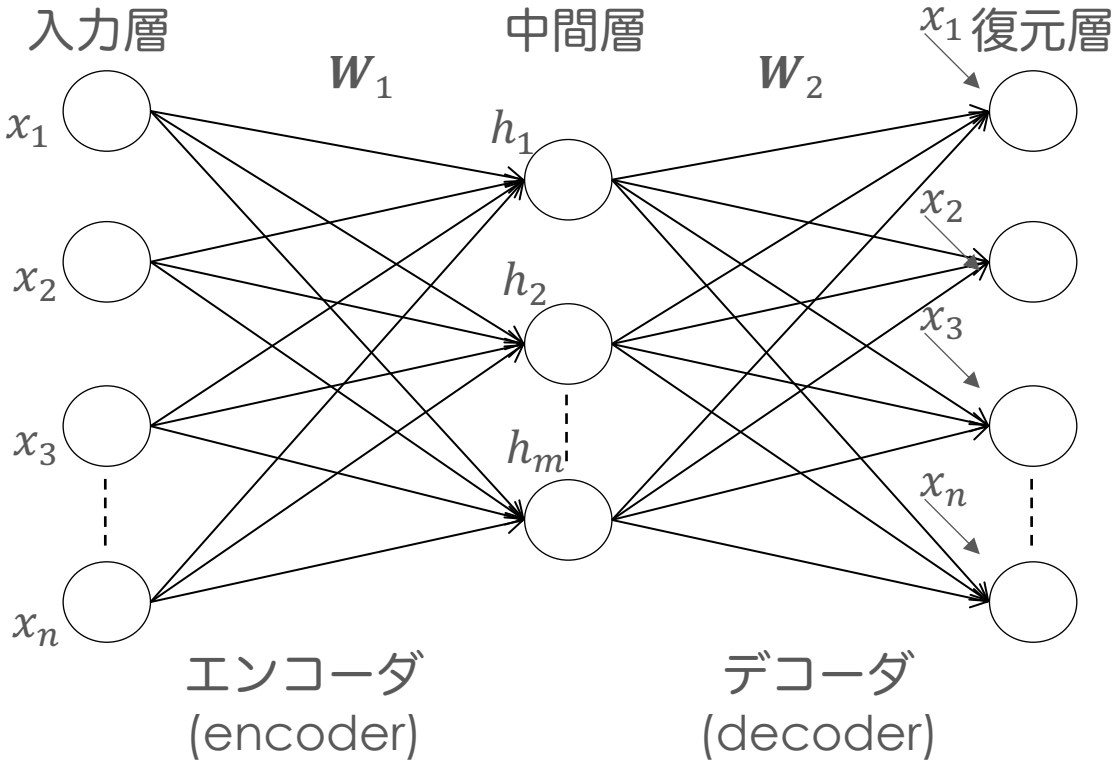
全結合型ニューラル
ネットワークの例



自己符号化器型の計算グラフ

- 自己符号化器型のニューラルネットワークでは、入力層と復元層に同じデータを入れて学習を行う。
- 学習によって求められた重み$\boldsymbol{W}$や中間層の出力$\boldsymbol{h}$に価値があり、それらが次の計算に利用される。

自己符号化器の例

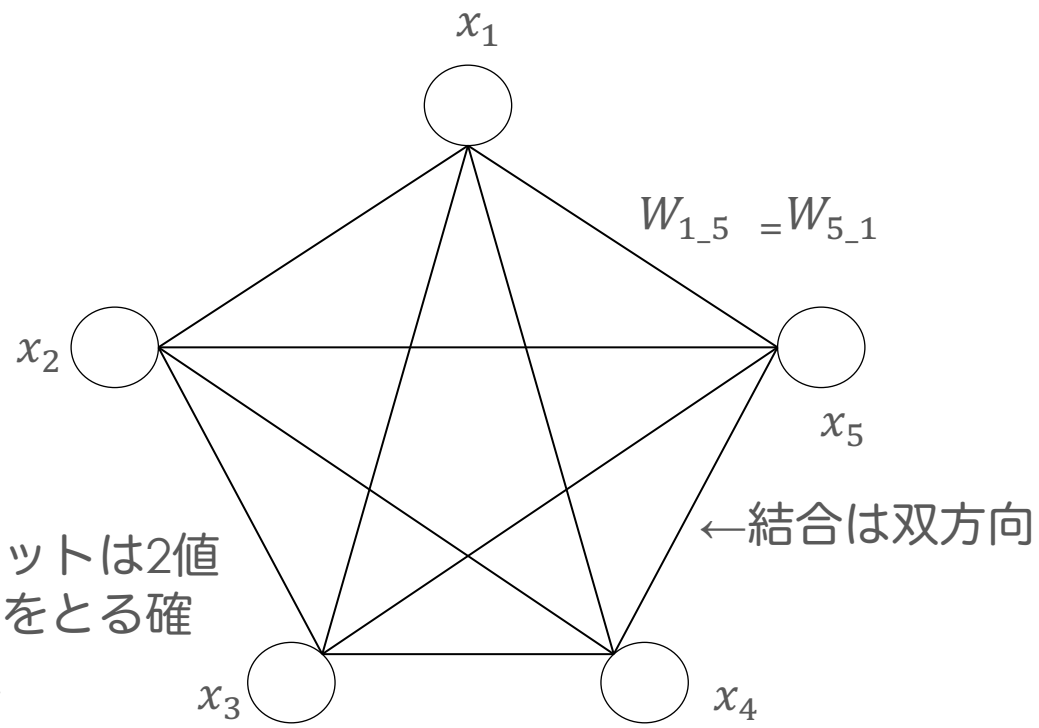


自己符号化器とは

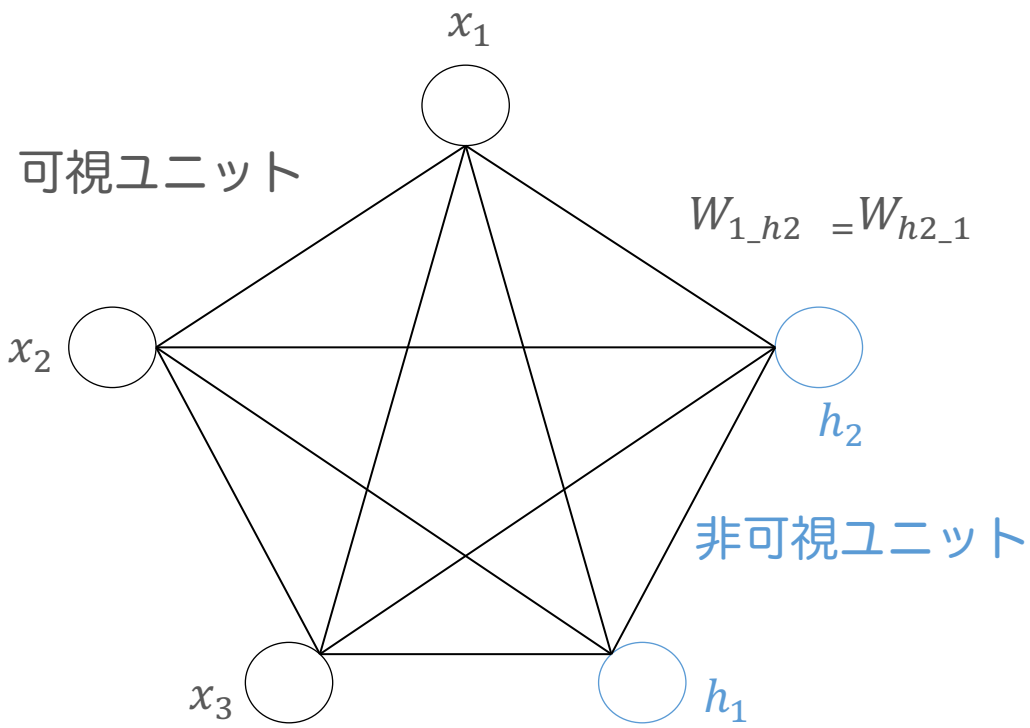
- 自己符号化器(autoencoder)とは、エンコーダとデコーダを組み合わせたネットワークを組み、ノード間の重みを学習する方法
- 特徴抽出器(次元圧縮)、ノイズ除去、階層型NNの初期パラメータの取得、復元誤差を利用した異常検知などに利用される。
- スパース自己符号化器は、自己符号化器に正則化項を加えて学習させる方法。効率的にユニット数を少なくできる。
- 雑音除去自己符号化器は、ランダムなノイズを付加したデータを入力層に入れる方法。復元層から出てくるデータがノイズを付加する前のデータに近くなるように学習させる。ノイズ除去の能力が備わることを期待できる。
- 変分自己符号化器は、生成モデルの1つ。データが生成するメカニズムを仮定する。DAY8の生成モデルで詳しく説明する。

ボルツマンマシン型の計算グラフ

ボルツマンマシンの例
(全てが可視ユニット)



隠れ層ユニットを持つ
ボルツマンマシンの例

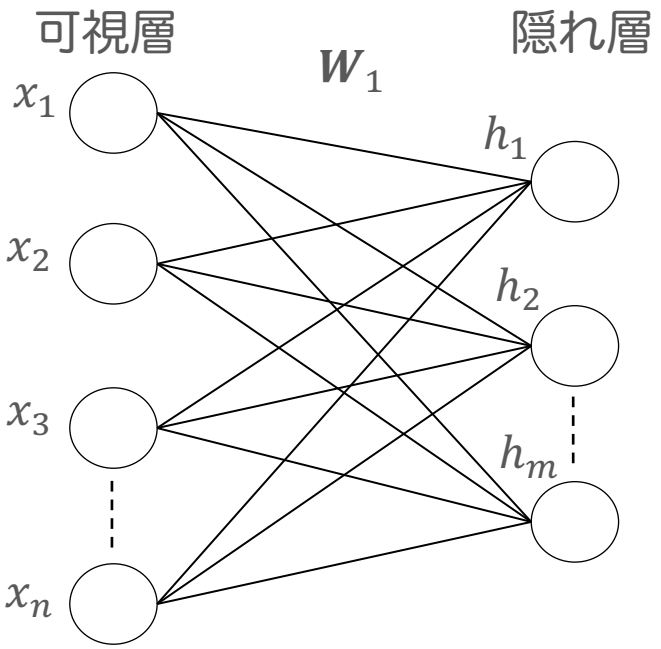


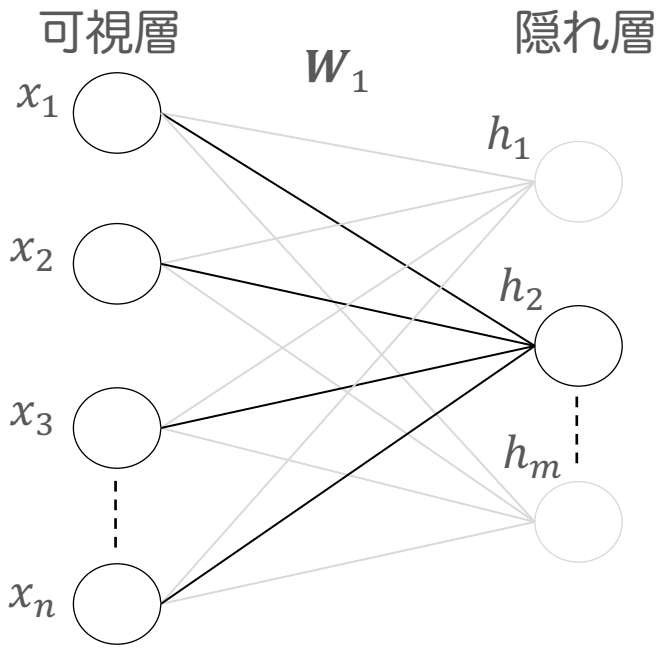
ボルツマンマシンとは

- ボルツマンマシン(Boltzmann machine)とは、ユニット間結合が双方向性を持つニューラルネットワークのこと。
- 1983年頃に提案された方法。
- 脳の連想記憶メカニズムに着想を得ている。
- ネットワークの挙動を確率的に記述することが特徴。
- ユニット間の結合重みは対称。例、$W_{1\_5} = W_{5\_1}$
- 各ユニットの出力は確率的に決まる。
- マルコフ確率場の1種。
- 生成モデルとして利用される。
- ボルツマンマシンを詳しく解説している書籍はこちら。
    - 『学習とニューラルネットワーク』(熊沢, 森北出版)

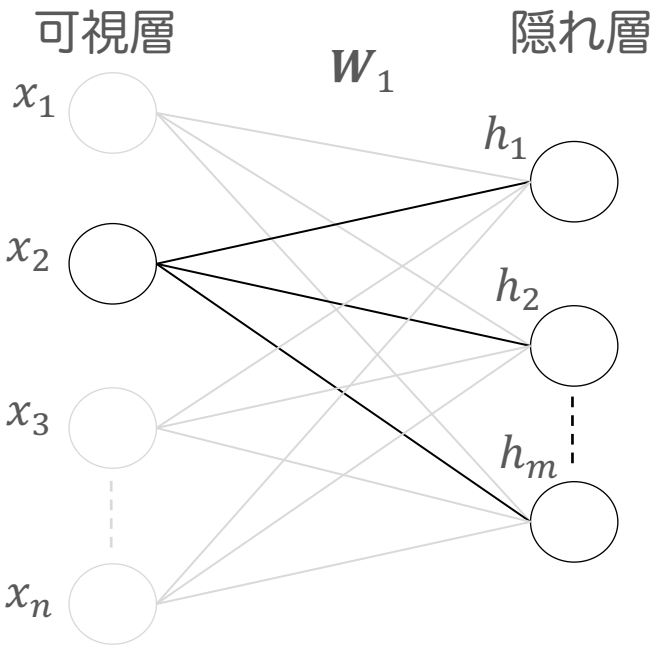
制約ボルツマンマシンの計算グラフ

制約ボルツマンマシンの例





可視層x_1~x_nが決まると、確率的に隠れ層h_2が決まる



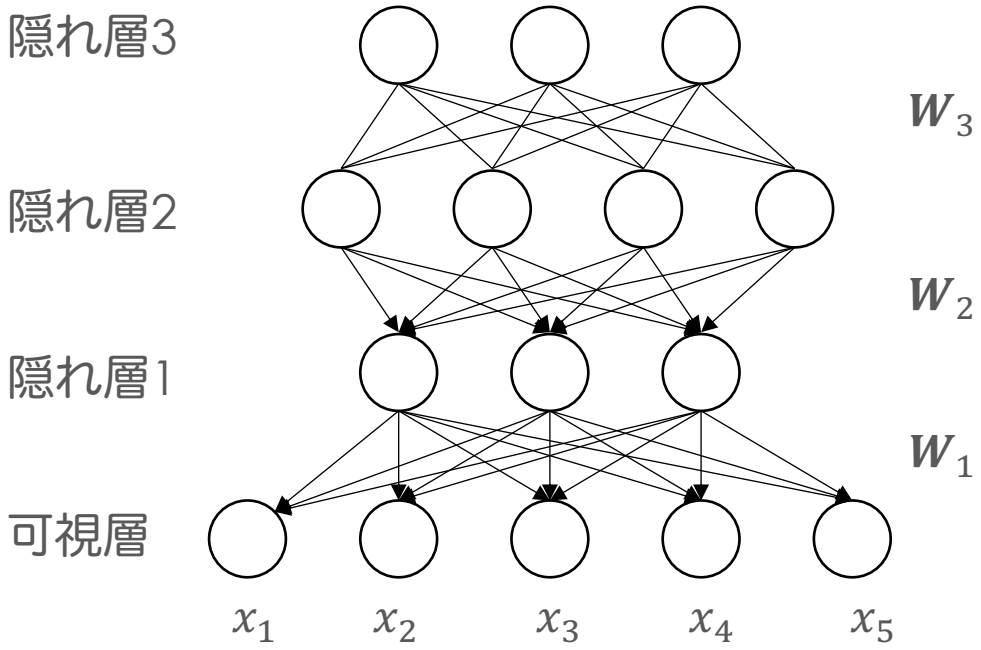
隠れ層h_1~h_mが決まると、確率的に可視層x_2が決まる

制約ボルツマンマシンとは

- 制約ボルツマンマシン(restricted Boltzmann machine)とは、隠れ変数を含むボルツマンマシンの一種。
- 1986年頃に提案された方法。
- 可視変数ユニットどうし、隠れ変数ユニットどうしの結合は持たない。
- 全ての結合は双方向。
- 自己符号化器と似た働きをもつ。
- 生成モデルとして利用されたり、自己符号化器のように隠れ層h_mを入力データの圧縮データとして扱うこともできる。
- 畳み込みニューラルネットワークの事前学習に用いられることもある。
- 深層信念ネットワークとは、制約ボルツマンマシンを多層化したもの。

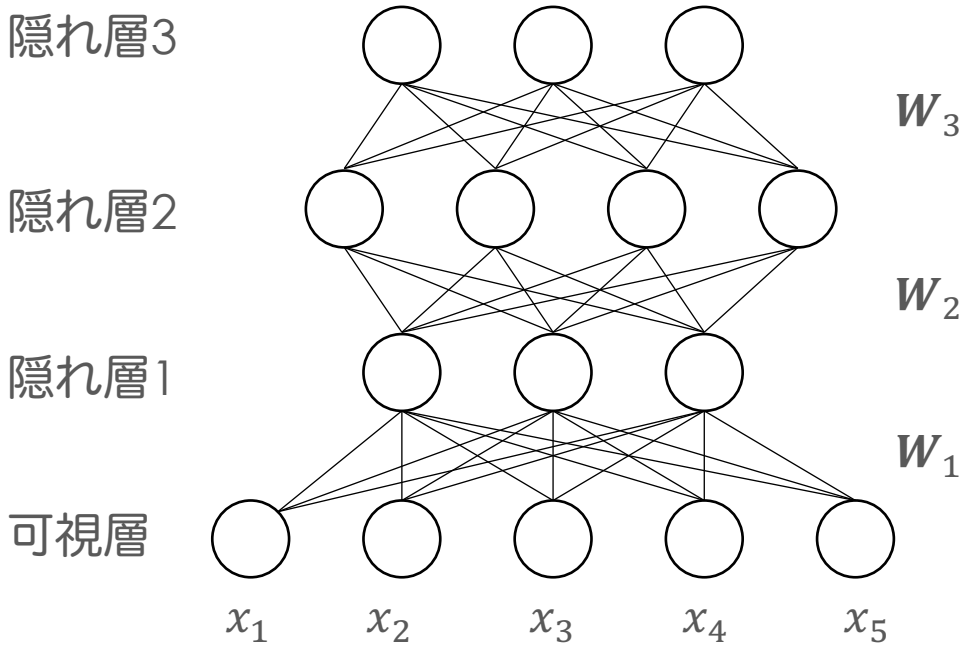
深層信念ネットワークと深層ボルツマンマシンの計算グラフ

深層信念ネットワークの計算グラフ例



最上位層のみ無向エッジ(双方向)で、
それ以外は有向エッジ

深層ボルツマンマシンの計算グラフ例



全ての結合が無向エッジ(双方向)

深層信念ネットワークとは

- 深層信念ネットワーク(deep belief network, DBN)とは、隠れ変数を含むボルツマンマシンの一種。
- 2006年頃に提案された方法。
- 近年の深層学習ブームの契機となったモデル。
- 最上位層のみ無向エッジ(双方向)で、それ以外は有向エッジ。

深層ボルツマンマシンとは

- 深層ボルツマンマシン(deep Boltzmann machine, DBM)とは、制約ボルツマンマシンを多層化したもの。
- 2009年頃に提案された方法。
- 全ての結合が無向エッジ(双方向) 。

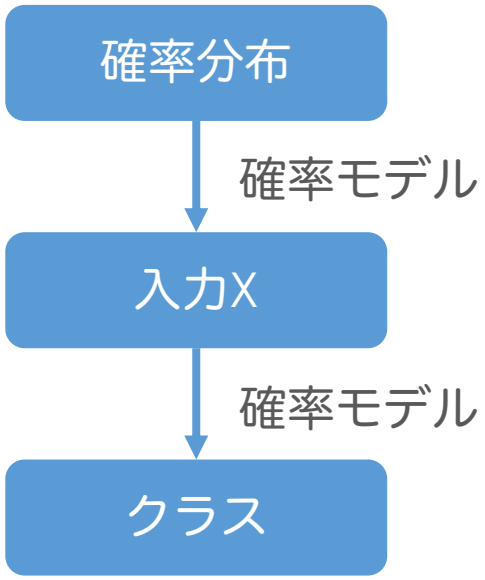
生成モデルとは

- 観測されたデータは、ある確率分布から生成されたサンプルであると仮定するモデリングの方法。
- 生成モデルでは、出力の分布だけでなく、入力の分布もモデル化する。
- これに対し、出力の分布だけを求める方法は、識別モデルと呼ばれる。
- 入力の分布がわかると、人工的なデータを生成することができるようになる。
  - 例、顔写真の生成, 室内写真の生成、文章のトピックの生成(トピックモデル)
- 代表的な生成モデルは、VAE(variational autoencoder)やGAN(generative adversarial network)。
- DAY8の生成モデルにて、VAEとGANを説明する。

クラス分類問題の3つのアプローチ

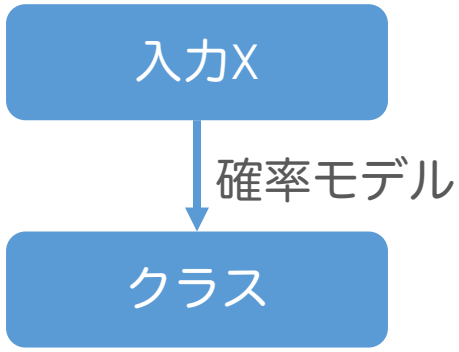
- クラス分類問題には3つのアプローチがある。参考：『パターン認識と機械学習, C.M.ビショップ, p.42』 『イラストで学ぶ機械学習, 杉山将, p.8』
- クラス分類を行いたいだけであれば、識別モデルまたは識別関数で十分。生成モデルでは、クラス分類を行えることに加え、人工的なデータを生成することができる。

生成モデル



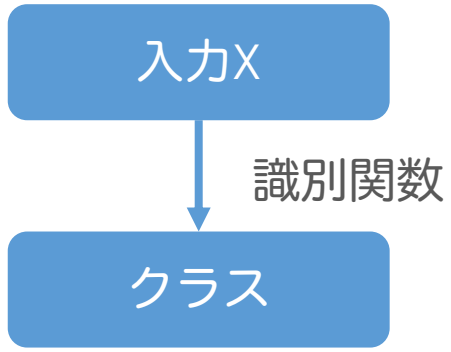
入力xはある確率分布から生成されたサンプルであると考え、その確率分布も同時に求める。

識別モデル



ある入力xとクラスの対応関係を確率的に考える。ロジスティック回帰,NNなど

識別関数



確率という考え方は出てこない。SVMなど

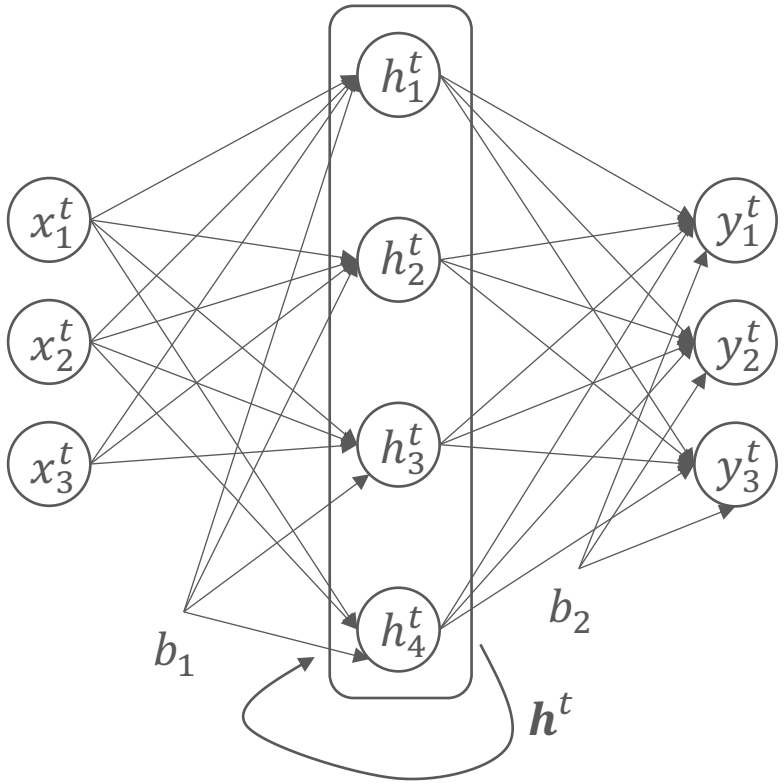
再帰型ニューラルネットワークとは

- 再帰型ニューラルネットワーク(recurrent neural network, RNN)とは、時間方向に状態を引き継ぎながら計算を進めることができるニューラルネットワーク。
- 自然言語などのように単語が順番に並んでいるデータを自然に扱うことができるため、自然言語処理や音声認識に利用される。
- 自然言語のように入力長が毎回異なるデータを扱う場合は、RNNが向いている。

再帰型ニューラルネットワークの種類

- シンプルなRNN
  - 単純な全結合層を用いて状態を更新していく。
  - 長い系列を学習しても、過去の情報がほとんど反映されないという欠点がある。
- LSTM (Long Short-Term Memory)
  - シンプルなRNNの課題を解決するために提案されたRNN。
  - 入力ゲート(input gate)、出力ゲート(output gate)、忘却ゲート(forget gate)、記憶セル(memory cell)と呼ばれる仕組みが導入され、必要な情報を長く記憶できるようになった。
  - “Long Short-Term Memory”とは、”短期記憶を長い時間継続できること”を意味する。
- GRU(Gated Recurrent Unit)
  - LSTMをシンプルにしたRNNであり、LSTMよりもパラメータの数が少ない。
  - リセットゲート(reset gate)と更新ゲート(update gate)のみで構成される。

シンプルなRNNの計算グラフ



中間層の出力が次の時刻
の中間層への入力になる

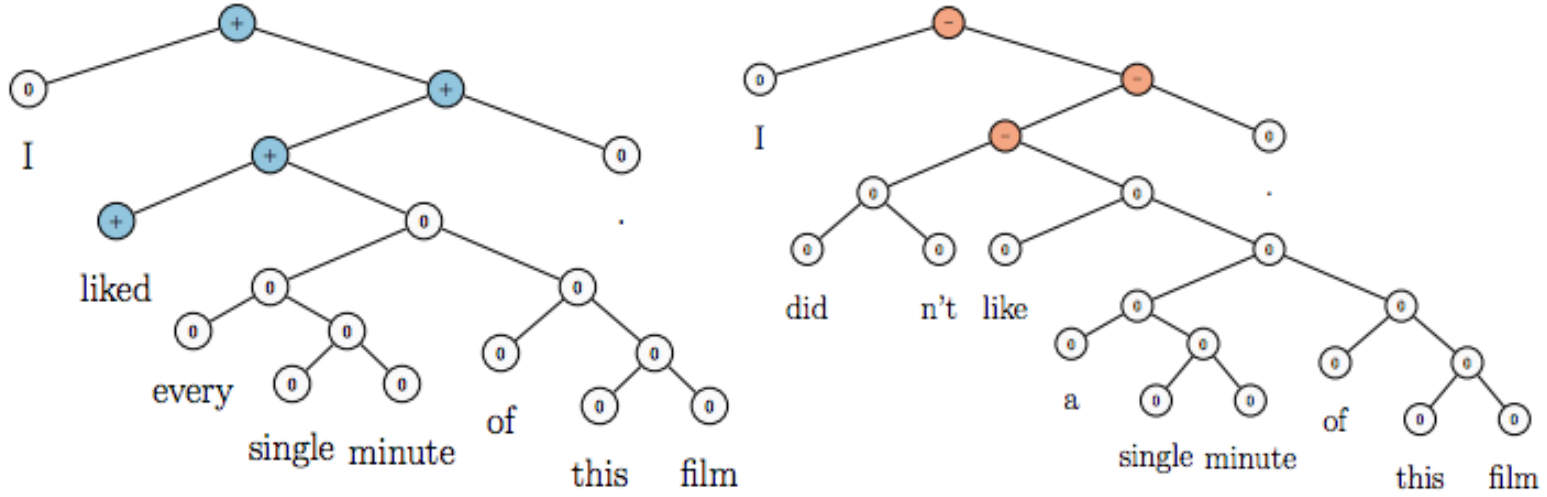
RNNに関する注意事項

- 本講座では、recurrent neural networkのことを再帰型ニューラルネットワークと呼んでいる。
- 『深層学習, Goodfellow』では、 recurrent neural networkは回帰結合型ニューラルネットワークと訳されている。また、この書籍では、再帰型ニューラルネットワークという言葉が、 recursive neural networkというモデルの訳として用いられている。
- E資格を受験される方は要注意。

参考：recursive neural network

- 木構造の計算グラフをもつネットワーク。
- 最適な木構造を決定する方法が確立されておらず、応用例は少ない。

recursive neural networkを用いた感情分析モデルの例



Richard Socher , Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank
https://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf

Any Questions?

講座の時間が余ったら

- 今回の復習をします。
- 次回の予習をします。