



CNNの基本用語

| CNNの基本用語

- ニューラルネットワーク

ニューラルネットワークは画像や時系列データなど

それぞれに特化したものが数多く存在する

→画像処理分野では

CNN（畳み込みニューラルネットワーク）がよく使われる

→言語処理、音声処理などは他のニューラルネットワークが使用

CNNの基本用語

- CNN（畳み込みニューラルネットワーク）
画像認識の分野に特化したニューラルネットワーク
→OCR、パターン認識など幅広い分野で使用されている
- OCR（光学的文書認識）
手書きや印刷された文字を読み取りデジタル化する技術

| CNNの基本用語

- CNN（畳み込みニューラルネットワーク）

- パターン認識（画像の場合）

画像データから一定のパターン・特徴を見つけ識別し、

取り出す技術のこと（文字認識、顔認識、画像認識）

→OCRは文字認識の1つである

→音声認識もパターン認識の1つである

| CNNの基本用語

- CNN（畳み込みニューラルネットワーク）

パターン認識

音声認識

画像認識

CNNの基本用語

・画像処理

画像データは**2次元データ**（縦と横の情報）なので

2次元データのまま処理をする方が望ましい

→画像データは上下左右、位置が重要な意味を持つことが多い

グッドマークは上下変わるだけで意味が変わる

→一般的なNNは1列で処理するため画像処理に向かない

CNNの基本用語

・画像処理

画像データ（2次元データ）の処理に適した
ニューラルネットワークが**CNN**である

- 画像データは厳密には3次元データである
赤色、緑色、青色の3色を使用して様々な色を表現している
- モノクロ写真の場合、2次元データである

| CNNの基本用語

- 画像処理



| CNNの基本用語

- CNNの歴史

福島邦彦によってCNNの原型となる

ネオコグニトロンと呼ばれるモデルが発表される

→人間の視覚野に関する神経細胞をもとに作られたモデル

→神経細胞の単純型細胞、複雑型細胞のはたらきを

もとにして作成されたモデルである

CNNの基本用語

- 神経細胞

- 単純型細胞 (S細胞)

- 画像の特徴を抽出する細胞のこと

- 複雑型細胞 (C細胞)

- 位置ずれを許容する細胞のこと

- 物体の位置が変わっても同一の特徴とみなす

| CNNの基本用語

- ネオコグニトロン

S細胞層とC細胞層を組み込んだモデル（初期のCNNモデル）

→S細胞層とC細胞層を交互に接続し、人間の脳の視覚野を再現

入力層→S細胞層→C細胞層→・・・→出力層

- S細胞層：単純型細胞（S細胞）の役割を持つ層
- C細胞層：複雑型細胞（C細胞）の役割を持つ層

CNNの基本用語

- LeNet

1998年にヤン・ルカンによって発表されたCNNのモデル

→学習に誤差逆伝播法を使用しているモデルである

ネオコグニトロンはadd-if silentを学習で使用していた

→入力層・畳み込み層・プーリング層（サブサンプリング層）・

全結合層・出力層で様々な処理が行われている

| CNNの基本用語

- LeNet

LeNetとネオコグニトロンと似た構造をしている

- 畳み込み層はS細胞層に対応している

→画像の特徴を抽出する層である

- プーリング層はC細胞層に対応している

→位置ずれを許容する層である

畳み込み層



畳み込み層

・畳み込み層

フィルタ（カーネル）を使って画像の特徴を抽出する層

→基本的にカーネルは画像サイズよりも小さいサイズのものを使う
カーネルのサイズをカーネル幅と呼ぶ

→畳み込み処理によって得られた2次元データを特徴マップという

畳み込み層

- 畳み込み層

画像

1	1	0	0
1	0	1	1
0	0	1	0
0	1	1	1

カーネル

1	1	0
0	1	1
1	0	1

(3×3)

畳み込み層

・畳み込み層

画像

1	1	0	0
1	0	1	1
0	0	1	0
0	1	1	1

カーネル

1	1	0
0	1	1
1	0	1

特徴マップ

畳み込み層

・畳み込み層

画像			
1	1	0	0
1	0	1	1
0	0	1	0
0	1	1	1

カーネル		
1	1	0
0	1	1
1	0	1

$$\begin{aligned} & 1 + 1 + 0 \\ & + 0 + 0 + 1 \\ & + 0 + 0 + 1 = 4 \end{aligned}$$

特徴マップ	
4	

畳み込み層

・畳み込み層

画像			
1	1	0	0
1	0	1	1
0	0	1	0
0	1	1	1

カーネル		
1	1	0
0	1	1
1	0	1

$$\begin{aligned} & 1 + 0 + 0 \\ & + 0 + 1 + 1 \\ & + 0 + 0 + 0 = 3 \end{aligned}$$

特徴マップ	
4	3

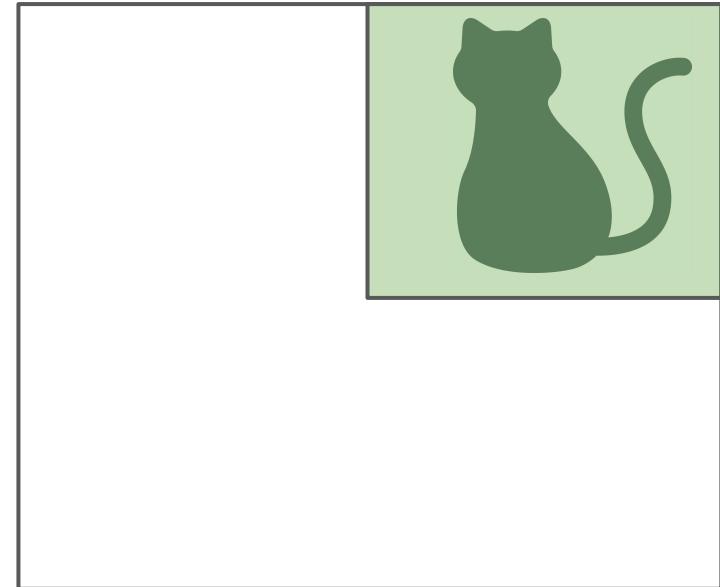
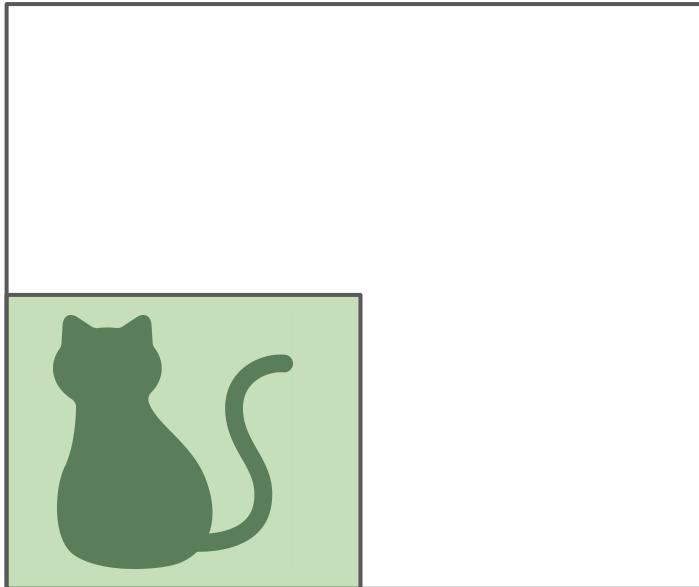
畳み込み層

・畳み込み層

フィルタの値（重み）は学習することによって変化していく
→猫を判別するCNNモデルを作る場合は、
猫の画像データを大量に渡し、フィルタの値を更新していく
→猫を判別することに特化したフィルタを作成することで、
猫の位置に関係なく猫を判別することができる

| 置み込み層

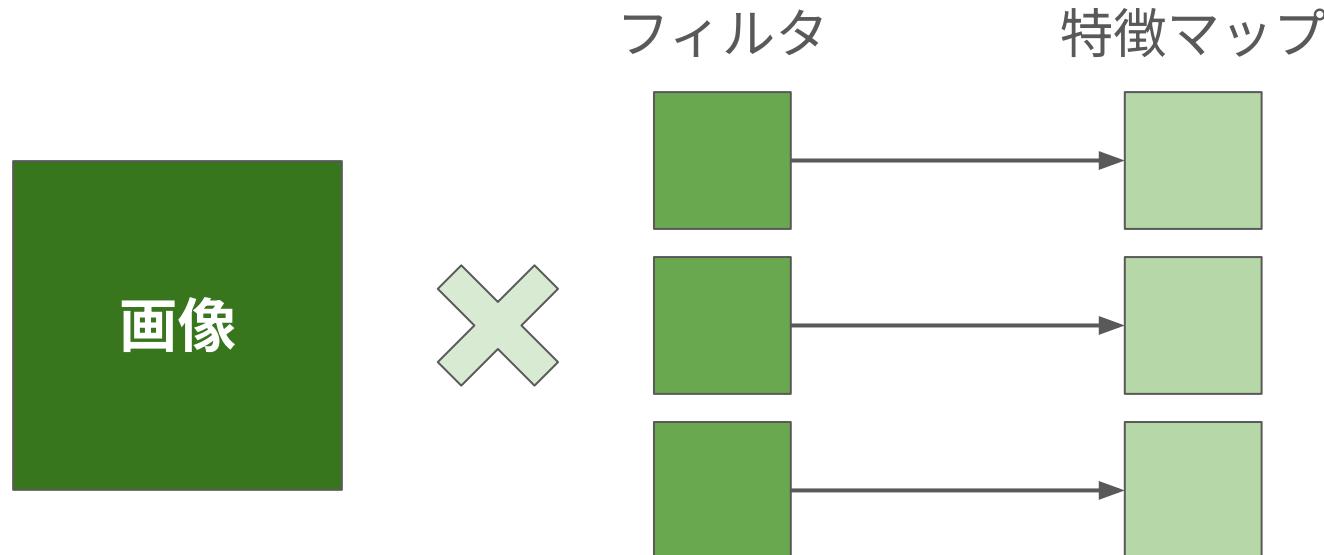
- 置み込み層



畳み込み層

・畳み込み層

フィルタは1枚だけではなく、複数のフィルタを使用する

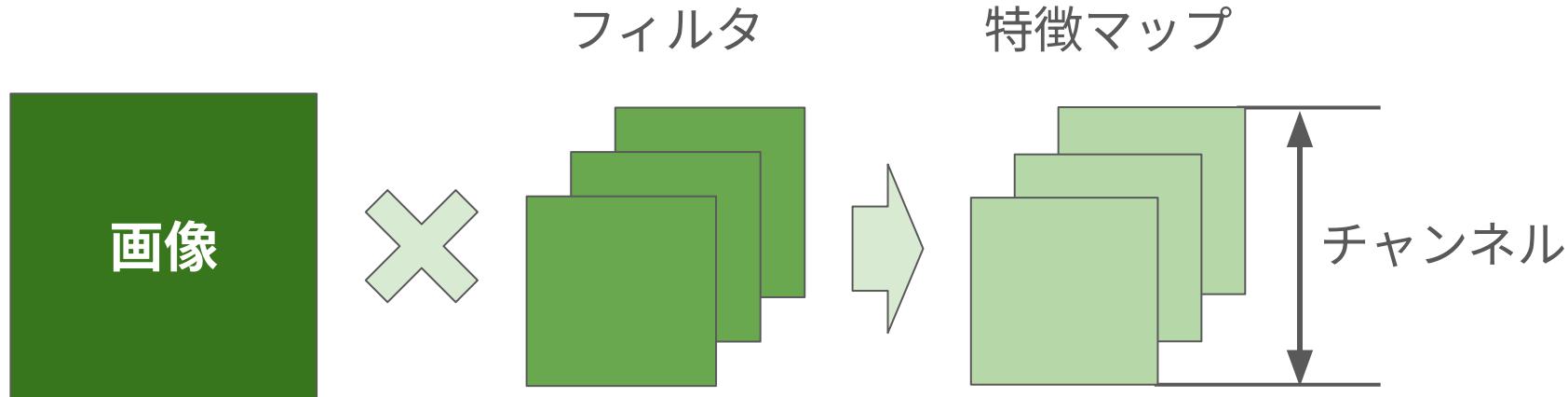


畳み込み層

- ・チャンネル数

畳み込み層で作成された特徴マップの数のこと

→特徴マップはカーネルの枚数分作成される



| 置み込み層

- ストライド

カーネルをスライド（移動）させる幅のこと

→ストライドが2なら、2ずつスライドさせていく

- パディング

作成する特徴マップの大きさを調整するために

画像データの周りに値（0）を入れること

畳み込み層

- ストライドが2の場合

画像

1	0	1	0	0
1	1	0	1	1
0	1	1	1	0
0	1	1	1	1
0	1	1	1	0

画像

1	0	1	0	0
1	1	0	1	1
0	1	1	1	0
0	1	1	1	1
0	1	1	1	0

画像

1	0	1	0	0
1	1	0	1	1
0	1	1	1	0
0	1	1	1	1
0	1	1	1	0

畳み込み層

・パディング

画像

0	0	0	0	0
0	1	0	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

特徴マップのサイズ計算



特徴マップのサイズ計算

・特徴マップのサイズ

畳み込み処理ではフィルタを使用して画像を抽出する

→畳み込み処理によって得られる**特徴マップのサイズ**は、
入力された画像サイズ、フィルタのサイズ、
パディング、ストライドが分かれば求めることができる

特徴マップのサイズ計算

- 特徴マップ幅

(入力画像幅 + パディング × 2 - フィルタ幅) ÷ ストライド + 1

- 特徴マップの高さ

(入力画像高さ + パディング × 2 - フィルタ高さ) ÷ ストライド + 1

特徴マップのサイズ計算

問題1

入力画像のサイズが 8×18 、 フィルタのサイズ 4×4 、
ストライドが 2、 パディングが 1 のとき、 特徴マップのサイズ

特徴マップの幅 : $(8 + 1 \times 2 - 4) \div 2 + 1$ から 4

特徴マップの高さ : $(18 + 1 \times 2 - 4) \div 2 + 1$ から 9

特徴マップのサイズは「 4×9 」になる

特徴マップのサイズ計算

問題2

入力画像のサイズが 13×25 、フィルタのサイズ 3×3 、
パディングが1、特徴マップのサイズは 4×7 のとき、ストライド

$$\text{特徴マップ幅} = (\text{画像幅} + \text{パディング} \times 2 - \text{フィルタ幅}) \div \text{ストライド} + 1$$

$$4 = (13 + 1 \times 2 - 3) \div \text{ストライド} + 1$$

$$4 = 12 \div \text{ストライド} + 1 \Rightarrow \text{ストライド} = 4$$

プリング層・全結合層



| プーリング層・全結合層

・ プーリング層

畳み込み層で出力した特徴マップをルールに従って
小さくしていく層で、重要な情報を残しながら圧縮していく
→ダウンサンプリング、サブサンプリング と言われる

→圧縮することで計算コストを下げることができる
重みなどはなく、決められた計算を機械的に行っている

| プーリング層・全結合層

- プーリングの操作

- マックスプーリング（最大値プーリング）

特徴マップの各区間の最大値を抽出する手法

- アベレージプーリング（平均値プーリング）

特徴マップの各区間の平均値を使用する手法

| プーリング層・全結合層

- マックスプーリング（最大値プーリング）

特徴マップ

3	7	8	10
3	4	3	4
4	9	2	2
1	1	3	7

最大値を抽出



7	10
9	7

| プーリング層・全結合層

- ・アベレージプーリング（平均値プーリング）

特徴マップ

3	7	8	10
3	4	3	4
4	9	2	2
1	1	3	7

平均値を計算



4.25	6.25
3.75	3.5

| プーリング層・全結合層

・ プーリング層

物体の位置が異なっていたとしてもデータを圧縮することで似た値になるため**位置のズレ**に強くなる

→**畳み込み層**と異なり**重みはない**ため学習しても変化はない

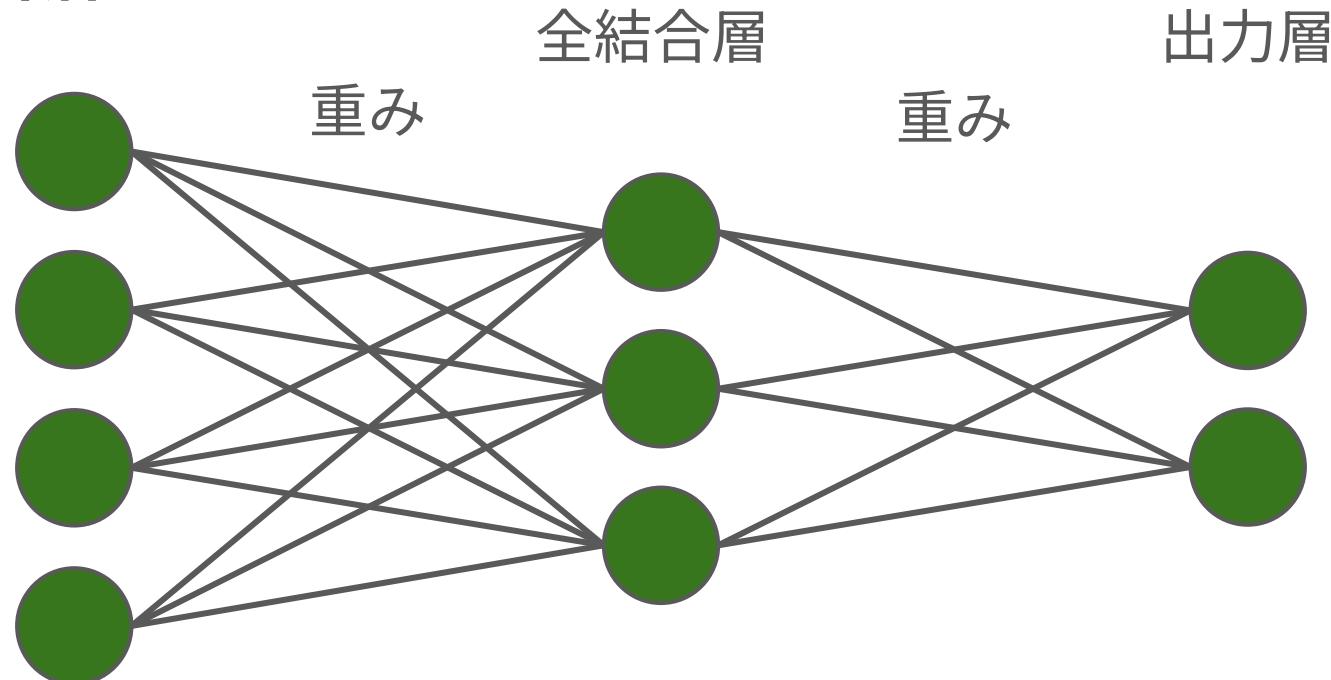
| プーリング層・全結合層

• 全結合層

- 一般的なニューラルネットワークと同様に
前後の層の全てのノードと結合している層のこと
→畳み込み層は一部のノードだけと結合している層である
- 画像データの特徴を抽出したデータを結合し、
活性化関数を用いて利用しやすい値に変換していく層である

| プーリング層・全結合層

- 全結合層



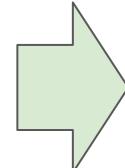
| プーリング層・全結合層

• 全結合層

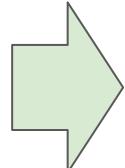
全結合層に値を入力する前に2次元情報である
特徴マップを**1列に並べる**処理をする必要がある

特徴マップ

3	1
5	2



3
5
1
2



全結合層

| プーリング層・全結合層

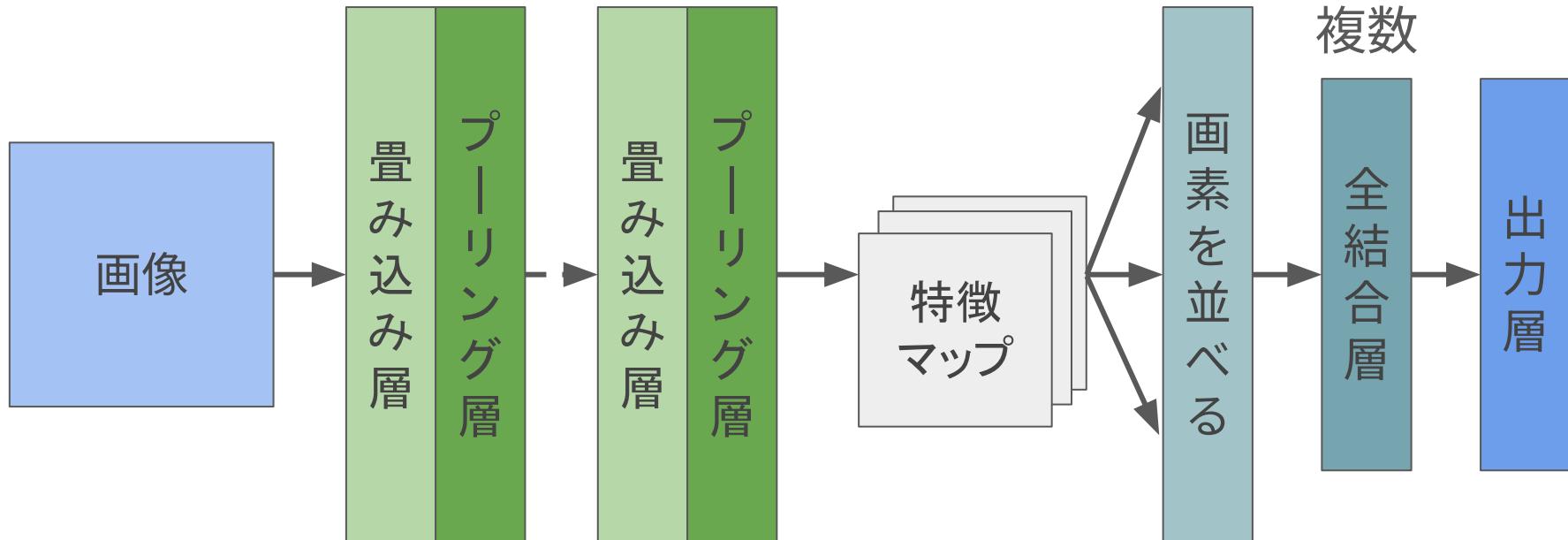
• 全結合層

畳み込み層で抽出された特徴マップをもとに
目的に応じた値を出力するための層と考えることができる
→シグモイド関数などを使って目的に応じた値を出力

→特徴を抽出した特徴マップの値に基づいて
分類を行っていくため「**分類器**」としての役割がある

| プーリング層・全結合層

・全結合層



| プーリング層・全結合層

- ・グローバルアベレージプーリング (GAP)

最近では全結合層を使わず、

特徴マップの平均値を使う処理がよく使われている

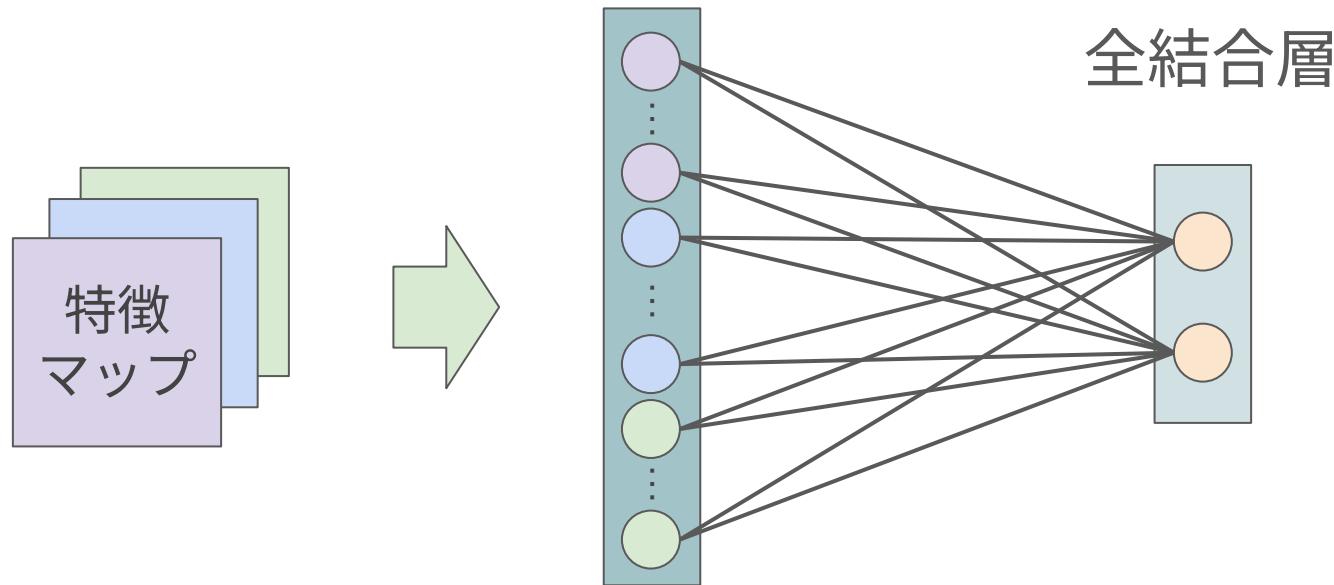
→全結合層を使用するよりもパラメータの数が少なくて済む

GAP層では全結合層と同様に活性化関数を使用して値を変換

→過学習が起きにくいとされている

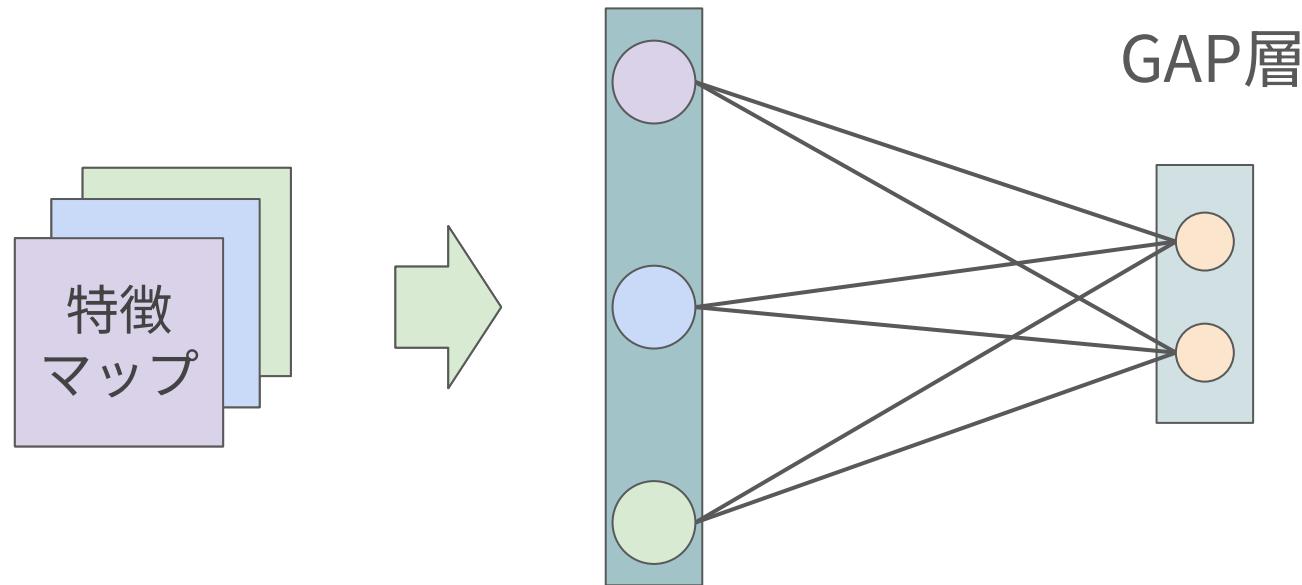
| プーリング層・全結合層

- ・グローバルアベレージプーリング (GAP)



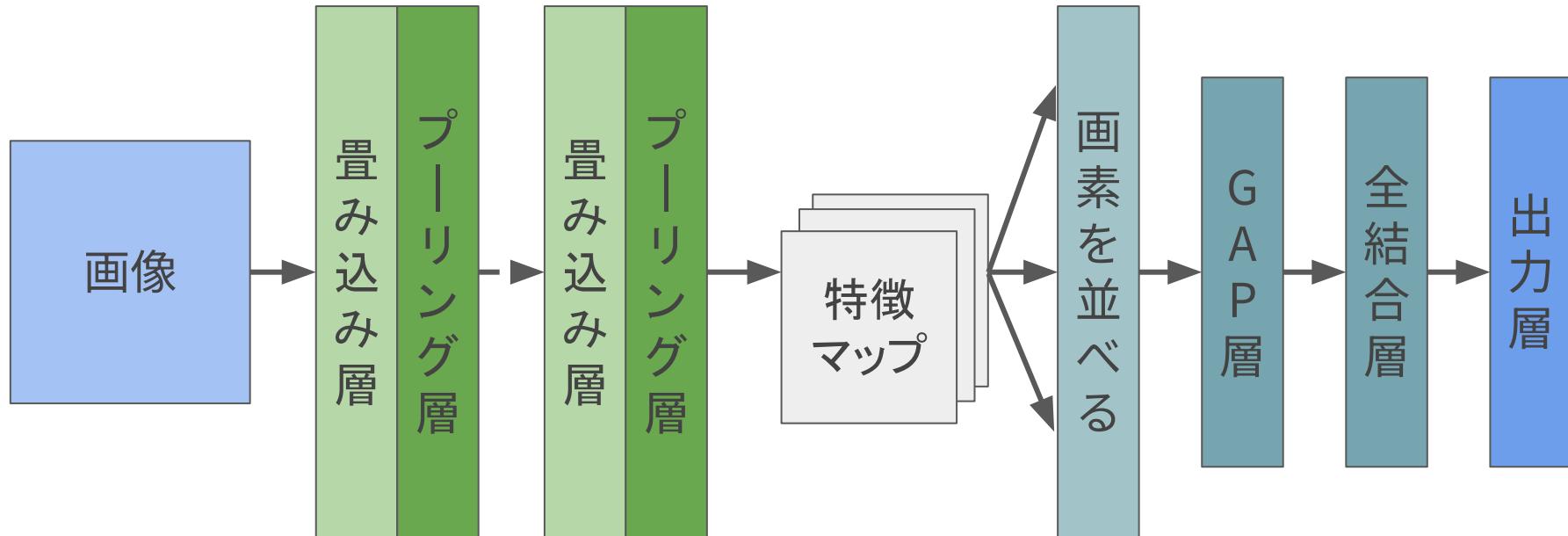
| プーリング層・全結合層

- ・グローバルアベレージプーリング (GAP)



| プーリング層・全結合層

・グローバルアベレージプーリング (GAP)



正規化層



正規化層

・正規化層

- 隠れ層などに入力するデータを正規化（標準化）するための層
→データが大きすぎたり、小さすぎたりすると使いにくいため
→過学習が起きにくくなり、学習が早くなる
- 使用される正規化の種類には、バッチ正規化、レイヤー正規化、
インスタンス正規化、グループ正規化などがある

正規化層

・バッチ正規化

各チャネルごとにミニバッチ全体のデータを正規化する手法

データA



データB



同じチャネルを

取り出す

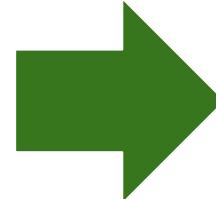


正規化層

・レイヤー正規化

各データごとにチャンネル全体で正規化する方法

データA



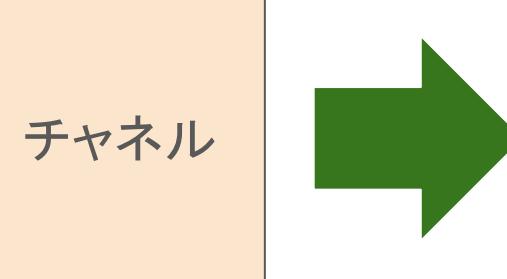
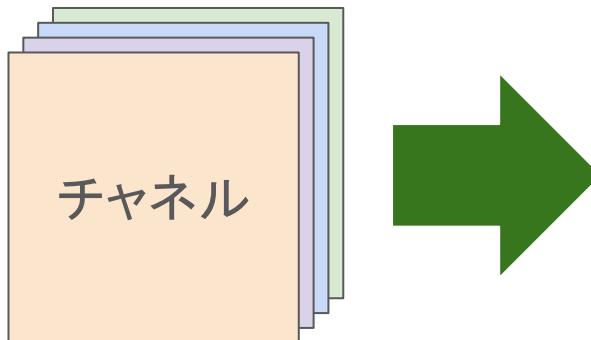
全チャンネルをまとめて
データを正規化

正規化層

- ・インスタンス正規化

各データごとに各チャネルを正規化する方法

データA



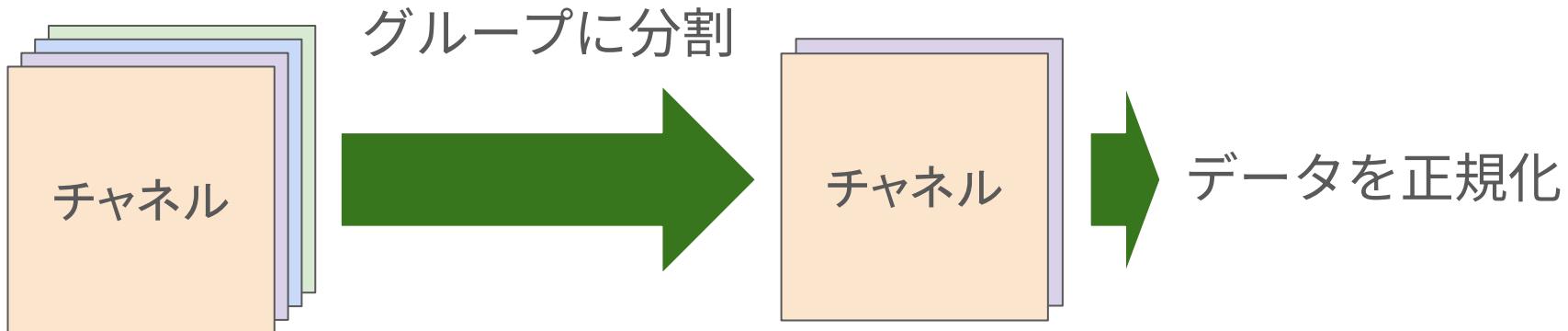
データを正規化

正規化層

・グループ正規化

各データごとにチャネルを分割、グループごとに正規化する手法

データA



オートエンコーダ



| オートエンコーダ

・オートエンコーダ

多層化したニューラルネットワークのパラメータを効果的に更新することができないことから人気がなかった

- ジェフリー・ヒントンが問題を解決する手法としてオートエンコーダ（自己符号化器）を提唱した
- 現在は異なる手法を使用して問題を解決している

| オートエンコーダ

・ オートエンコーダ

可視層と隠れ層の2層から構成されるネットワーク

→可視層とは入力層と出力層がセットになったもの

→データが可視層（入力層）から隠れ層、

隠れ層から可視層（出力層）へ伝わっていく

→入力と出力が同じデータになるように学習を行っていく

| オートエンコーダ

- オートエンコーダ

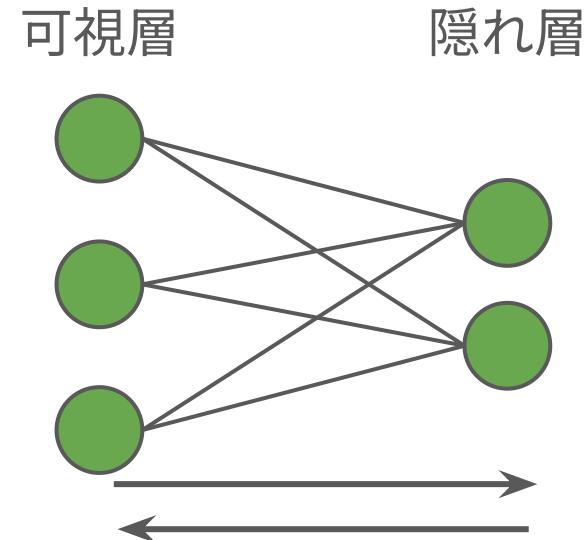
隠れ層のノード数は可視層のノードの数よりも少ない

→入力データが隠れ層で圧縮されている

重要な情報を抽出している

→出力された情報が入力の**情報**と同じに

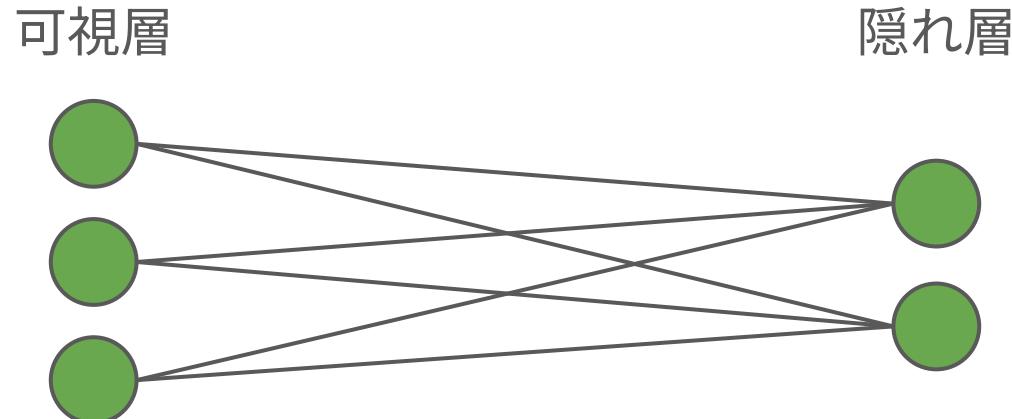
なるように学習が行われていく



| オートエンコーダ

- ・オートエンコーダ

可視層から隠れ層への処理をエンコードといい、
隠れ層から可視層への処理をデコードという



| オートエンコーダ

・ 積層オートエンコーダ

ジェフリー・ヒントンが提唱したモデル

オートエンコーダを何層も重ね合わせたものである

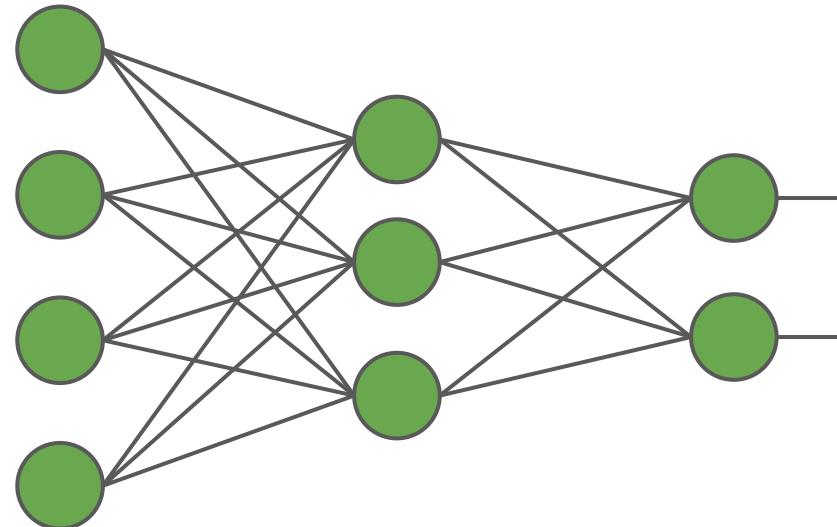
→可視層（入力層）から順番に学習していく方法が採用

→順番に学習していくことを事前学習という

入力層から順番に学習を行うため勾配消失問題は発生しない

| オートエンコーダ

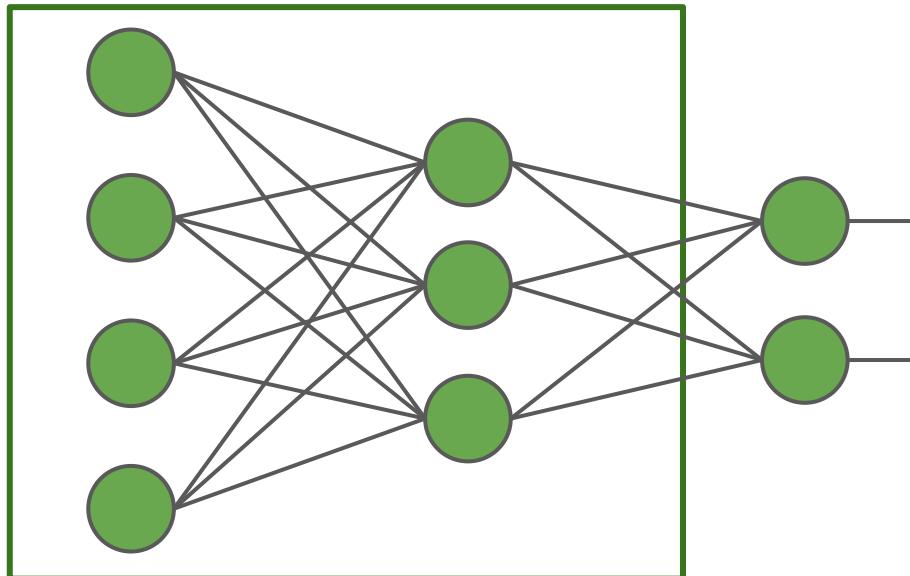
- ・積層オートエンコーダ



| オートエンコーダ

- ・積層オートエンコーダ

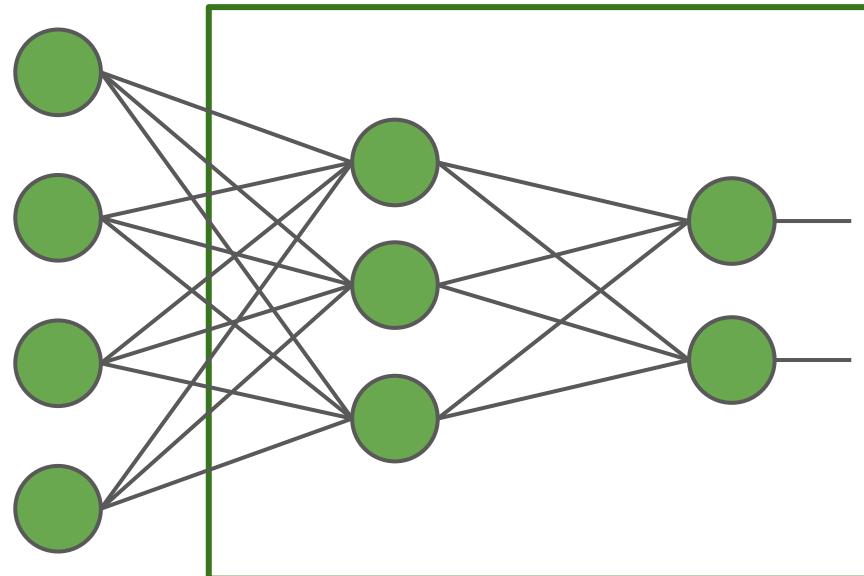
学習



| オートエンコーダ

- ・積層オートエンコーダ

学習



| オートエンコーダ

・ 積層オートエンコーダ

- ニューラルネットワークの初期値に事前学習させた値を使用することで層が深くなても効果的な学習が可能
→ 勾配消失問題を抑制することができた
- 事前学習は計算量が膨大になるため
現在はネットワーク全体を一気に学習させるようになった

| オートエンコーダ

・ 積層オートエンコーダ

- 出力と入力が同じ値になるように学習しているため
教師あり学習（回帰・分類）にはならない
→教師なし学習（次元削減）の状態になっている
- オートエンコーダを積み重ねた最後の層に予測を行う層を
追加することで予測や分類ができるようになる

| オートエンコーダ

・ 積層オートエンコーダ

- 分類問題ならば、ロジスティック層を最後に追加する
→2値分類ではシグモイド関数（2値分類）、
多クラス分類ではソフトマックス関数が使用される
- 回帰問題ならば、線形回帰層を最後に追加する
層を追加したため、モデル全体の重みを調整する必要がある

| オートエンコーダ

・ 積層オートエンコーダ

モデル全体で重みを再学習（調整）させることを

ファインチューニングという

→出力層あたりの重みを中心に調整するため誤差逆伝播法が有効

→重みを調整するだけなので学習率は低く設定しておく

| オートエンコーダ

- 積層オートエンコーダ

教師なし学習（次元削減）の状態になっている

→エンコーダでは重要な情報を残して圧縮している

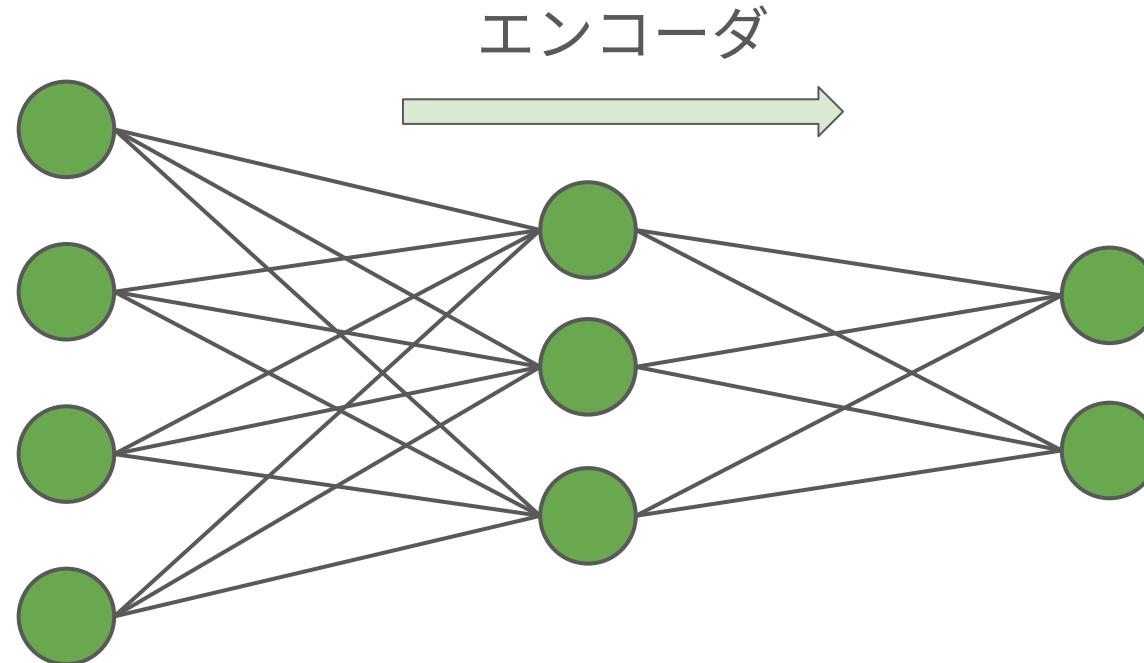
重要度が低い情報は削ぎ落とされるという特徴がある

→画像を入力すると、エンコードでノイズが取り除かれる

ノイズ処理された画像が出力される

| オートエンコーダ

- ・積層オートエンコーダ



| オートエンコーダ

- 深層信念ネットワーク

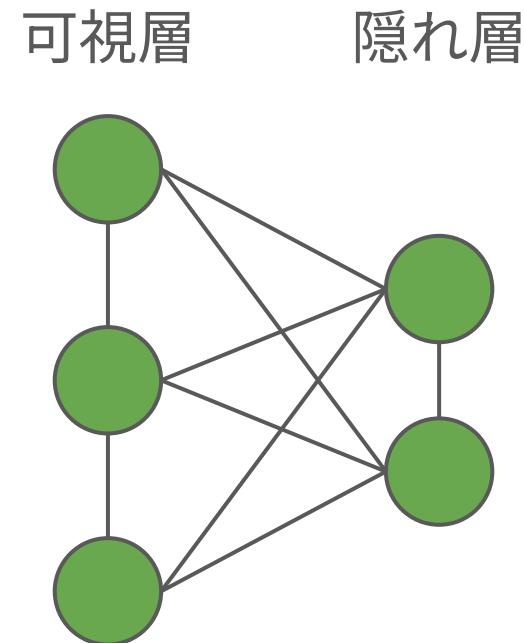
ジェフリー・ヒントンが提唱したモデル

→オートエンコーダの代わりに

制限付きボルツマンマシンを積み重ねたもの

- ボルツマンマシン

各ユニットが全て結合されている



| オートエンコーダ

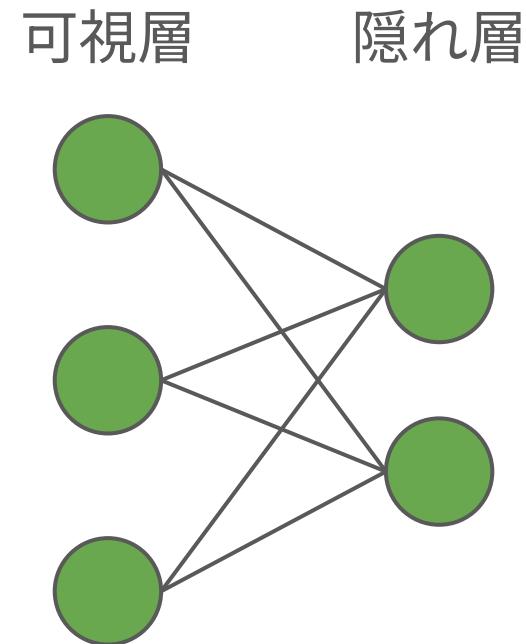
- ・深層信念ネットワーク
- ・制限付きボルツマンマシン

同じ層のユニットは結合しておらず

可視層と隠れ層のユニットを結合している

→オートエンコーダとの違いは

使われているアルゴリズムが異なること



時系列データ



| 時系列データ

・時系列データ

気温、株価の動き、人口動態など時間的に変化した情報を保有しているデータのこと

- ある時点の情報は、その時点よりも**前の情報**に影響を受ける
- 時系列データを使用して予測するとき、過去の情報を入力する
- どれだけ過去の情報を入力するかが難しい

| 時系列データ

・時系列データ

- 過去の情報が多すぎると計算コストが高くなったり、
ネットワークが複雑化したりして現実的ではない
→過去の情報が少なすぎると精度が落ちてしまう可能性がある
- 過去の情報をいかに反映させるかが難しい
データは時系列に沿って入力していくことが求められる

| 時系列データ

・ 時系列データ

- 音声データやテキストデータも時系列データの1つ
→過去の情報が、その後の情報（未来の情報）に影響を与える
- 音声データやテキストデータは
気温、株価の動き、人口動態などと異なり
未来の情報が、過去の情報に影響を与えることもある

RNN



RNN

- ・時系列データ

時系列データは過去の情報が現在や未来に影響を与える

→過去の情報などを保持し、現在や未来に適度に影響を与える必要

→一般的なニューラルネットワークは過去の情報を保持し

現在や未来に適度に影響を与えることが得意ではない

→各時刻に対して**独立の処理**を行うという特徴があるため

RNN

- ・リカレントニューラルネットワーク（RNN）

過去の情報などを保持し、現在や未来の情報に

適度に影響を与えることができるニューラルネットワーク

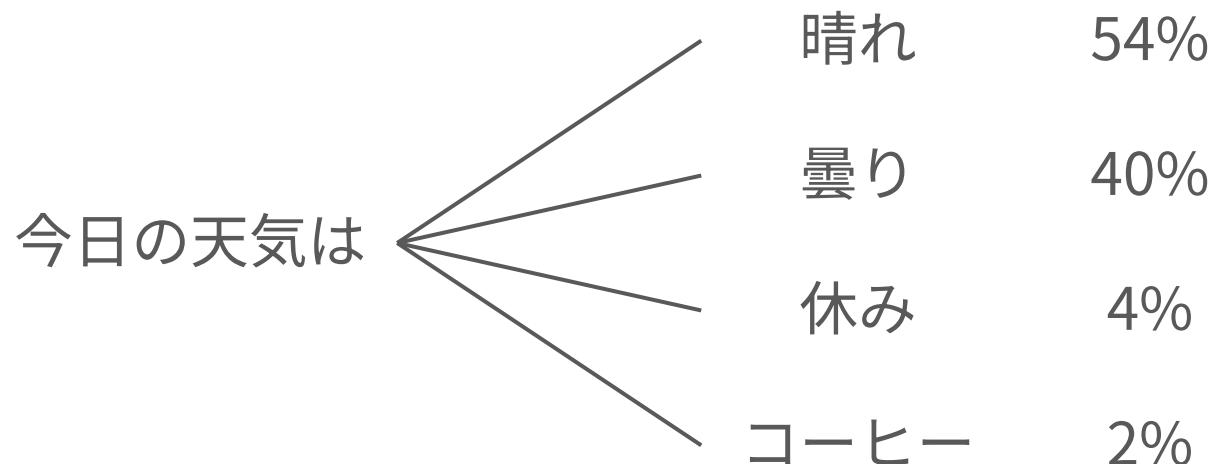
→時系列データを扱うときに使用されるニューラルネットワーク

→機械翻訳、音声処理、文章生成、言語モデルなどで使用される

RNN

- 言語モデル

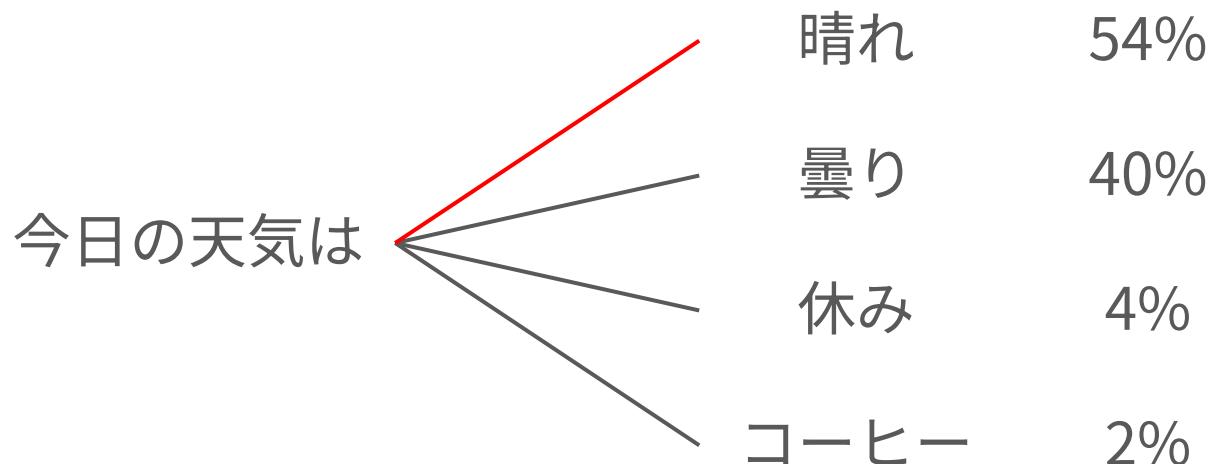
人間の言語を単語の出現確率などを用いてモデル化したもの



RNN

- 言語モデル

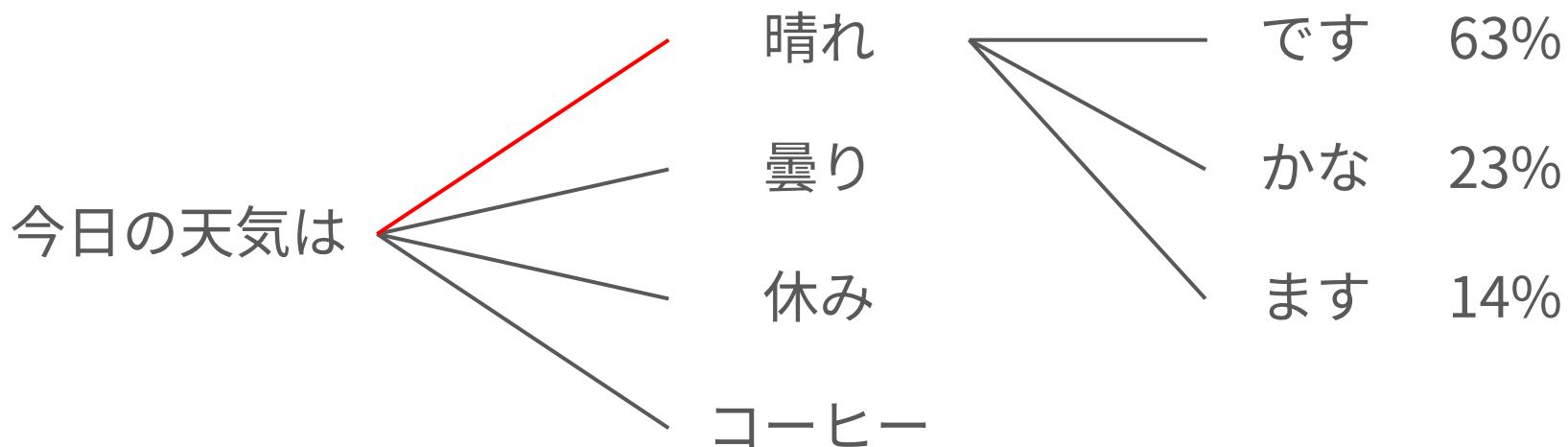
人間の言語を単語の出現確率などを用いてモデル化したもの



RNN

・言語モデル

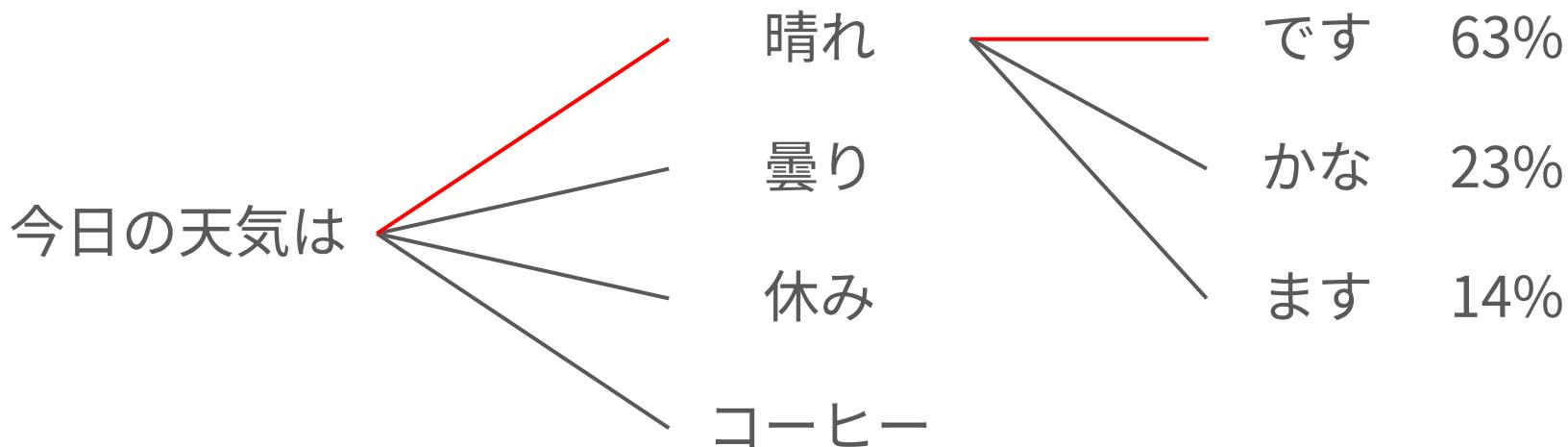
人間の言語を単語の出現確率などを用いてモデル化したもの



RNN

- 言語モデル

人間の言語を単語の出現確率などを用いてモデル化したもの



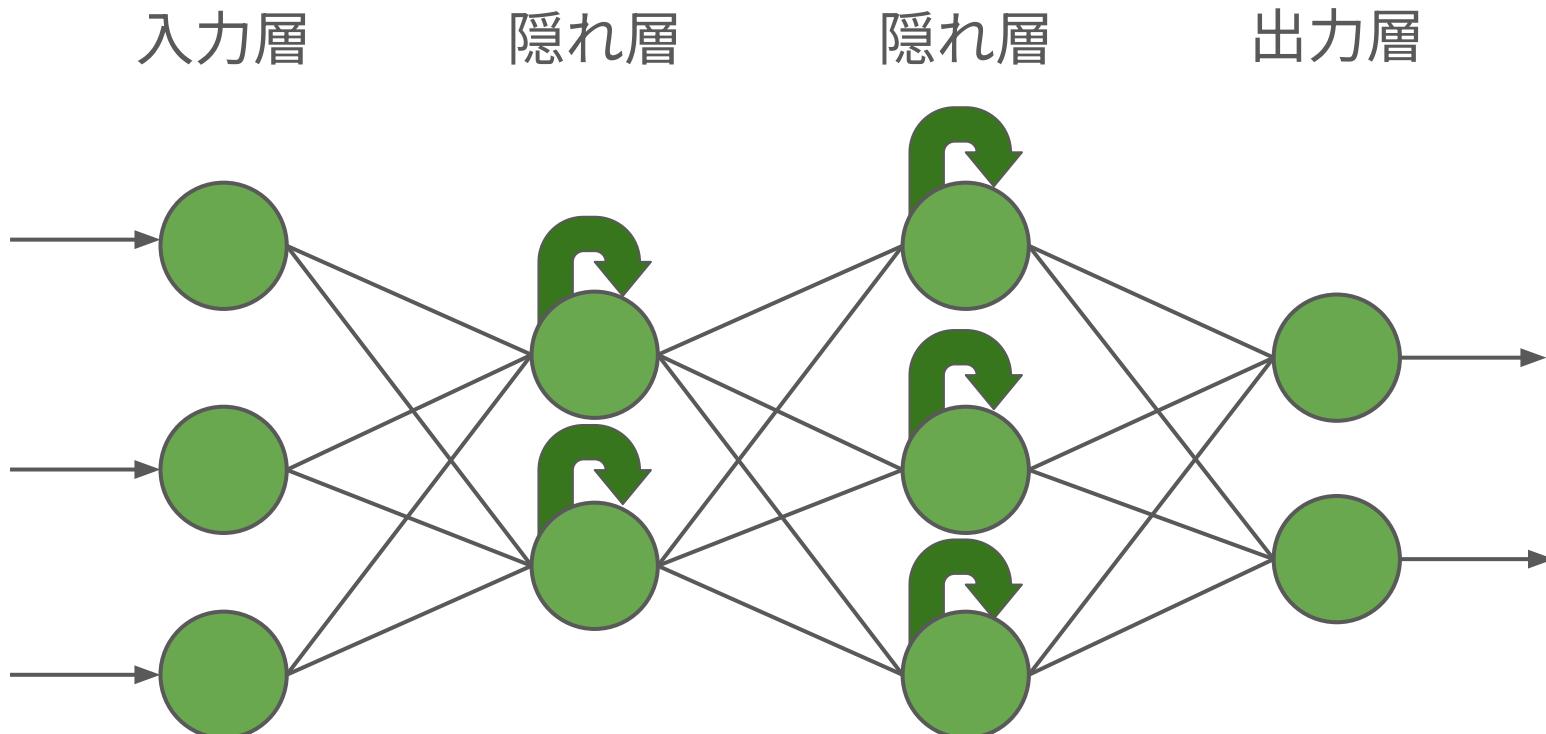
RNN

- ・リカレントニューラルネットワーク（RNN）

隠れ層に過去の状態を反映させる仕組みを導入することで
時系列データなどを上手いこと処理することができる

- 隠れ層と過去の隠れ層に繋がり（重み）を作ることで
過去の情報を上手いこと反映させることができる
- 「再帰構造」をもったニューラルネットワークである

RNN



RNN

- ・リカレントニューラルネットワーク (RNN)

パラメータの更新はニューラルネットワークと同じように
誤差逆伝播法が使われている

- RNNでは過去のパラメータも更新していく必要があり
BPTT (BackPropagation Through-Time) が使用されている
- 古いデータから時間軸にそって誤差を伝播させていく手法である

RNN

- ・リカレントニューラルネットワーク（RNN）

音声認識でもRNNは使用されている

→音声によっては、入力された音声データの数（入力数）と
出力すべき音声データの数（出力数）が一致しないことがある

→NNでは入力数と出力数を一致させる必要がある

→CTCという手法を導入することで問題を解決している

RNN

- ・入力数と出力数が一致しないケース

「こんにちはああ」と元気な声で挨拶した音声を入力
出力するべきテキストは「こんにちは」である場合
→入力数は「こんにちはああ」で7であるが、
求める出力数は「こんにちは」で5である
→このような矛盾を解決する手法がCTCである（空文字、縮約）



エルマンネットワーク ジョーダンネットワーク

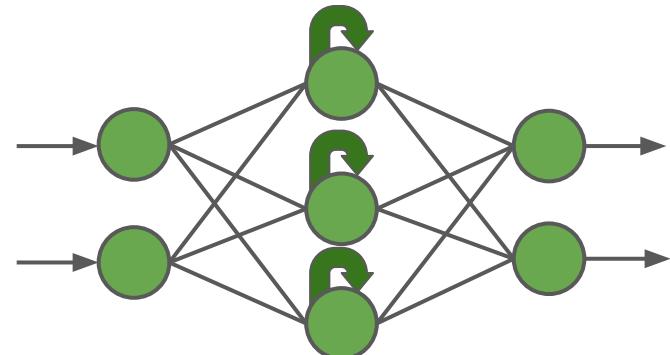
| エルマンネットワークとジョーダンネットワーク

- RNN

隠れ層と過去の隠れ層に繋がりを作ることで
過去の情報を上手いこと反映させている

→過去の出力を次の入力として利用する
再帰的な構造を持つ層のことを
特に回帰結合層という

回帰結合層
(隠れ層)



| エルマンネットワークとジョーダンネットワーク

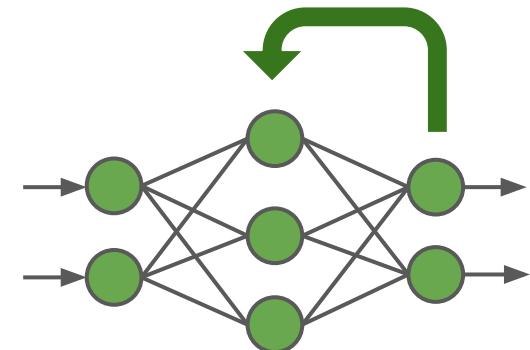
・エルマンネットワークとジョーダンネットワーク

隠れ層と過去の隠れ層に繋がりを作るような

RNNをエルマンネットワークという

隠れ層と過去の出力層に繋がりを作るような

RNNをジョーダンネットワークという



回帰結合層
(出力層)

RNN (LSTM・GRU)

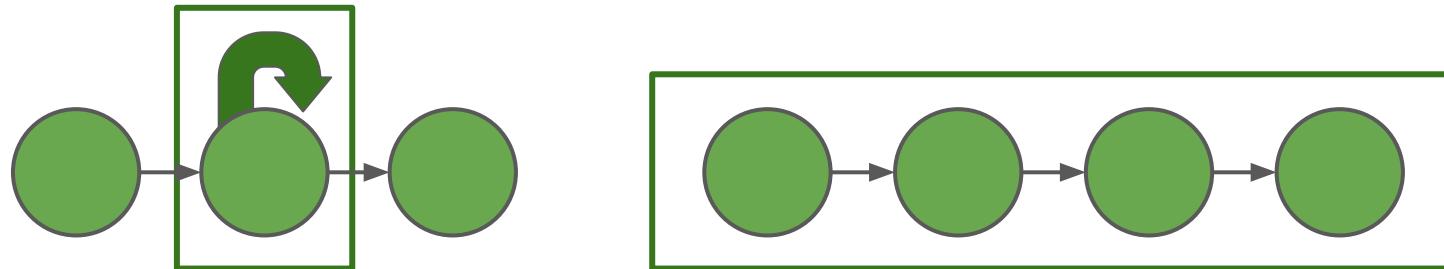


RNN (LSTM・GRU)

- リカレントニューラルネットワーク (RNN)

RNNでは過去のパラメータも更新していく必要があるため
層が深くなりやすい特徴がある

→勾配消失問題が発生しやすく、学習が進みにくい特徴



RNN (LSTM・GRU)

・リカレントニューラルネットワーク (RNN)

一般的に関係のある入力は重みが大きく、

関係のない入力は重みを小さくすることが原則である

→現在にとって関係のない入力でも、未来にとって関係のある

入力の場合重みを大きくすることがあり、重みの原則に反する

→このような問題のことを入力重み衝突という

RNN (LSTM・GRU)

- ・リカレントニューラルネットワーク (RNN)

一般的に関係のある出力は重みが大きく、

関係のない出力は重みを小さくすることが原則である

→現在にとって関係のない出力でも、未来にとって関係のある

出力の場合重みを大きくすることがあり、重みの原則に反する

→このような問題のことを出力重み衝突という

RNN (LSTM・GRU)

- LSTM (Long Short-Term Memory)

勾配消失問題、入力重み衝突や出力重み衝突のような問題を
回避するために考えられた手法になる

→LSTMブロックと呼ばれる機構を導入することで

時系列情報を保持することできる

→LSTMブロックはニューラルネットワークのユニットに対応



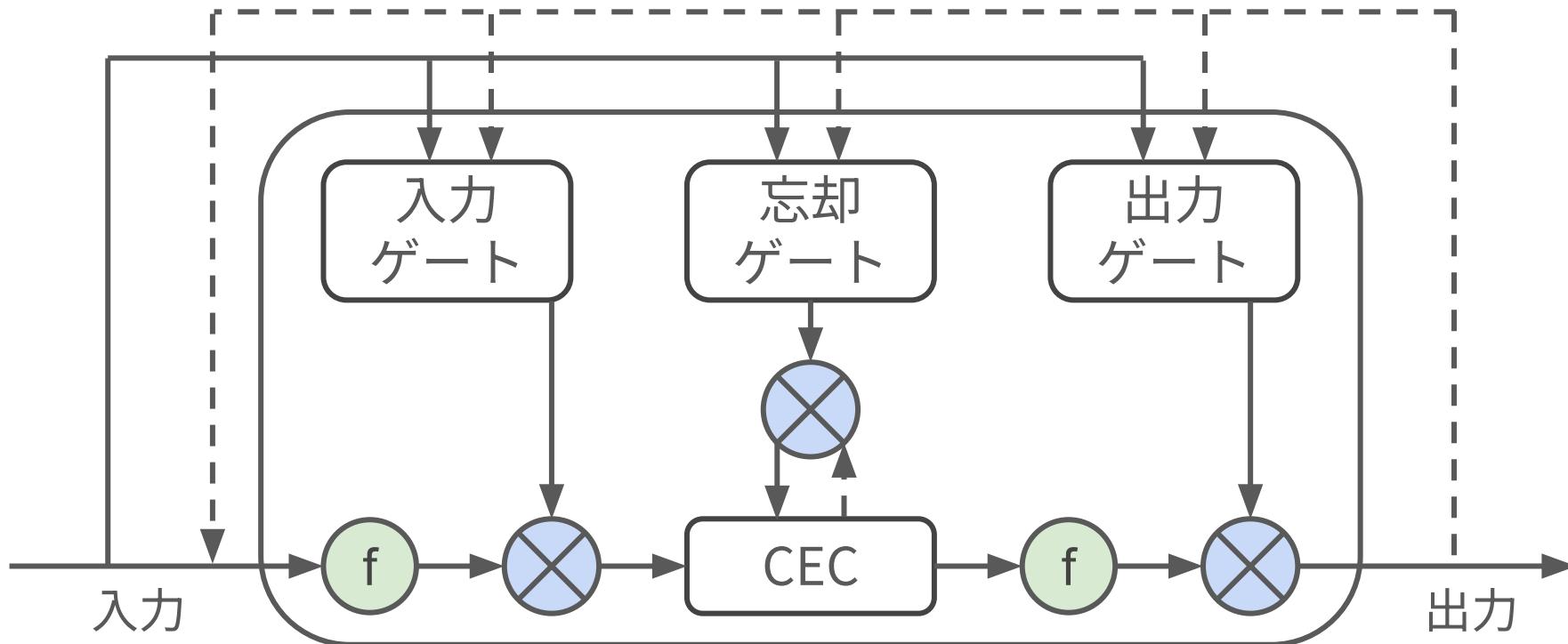
活性化関数

———— 現在の時刻のデータ



要素積

----- 1つ前の時刻のデータ



RNN (LSTM・GRU)

- LSTMブロック

- CEC

誤差を内部にとどめ勾配消失を防ぐ、セルとも呼ばれる

- 入力ゲート：入力重み衝突に対応するゲート
- 出力ゲート：出力重み衝突に対応するゲート
- 忘却ゲート：過剰な誤差をリセットするゲート

RNN (LSTM・GRU)

• LSTMとGRU

LSTMは複雑なため計算量が多くなってしまう特徴

→ゲートの数を減らし計算量を削減したモデルが**GRU**である

GRUは**Gated Recurrent Unit**の略である

→リセットゲート、更新ゲートが

入力ゲート、出力ゲート、忘却ゲートの役割を果たす



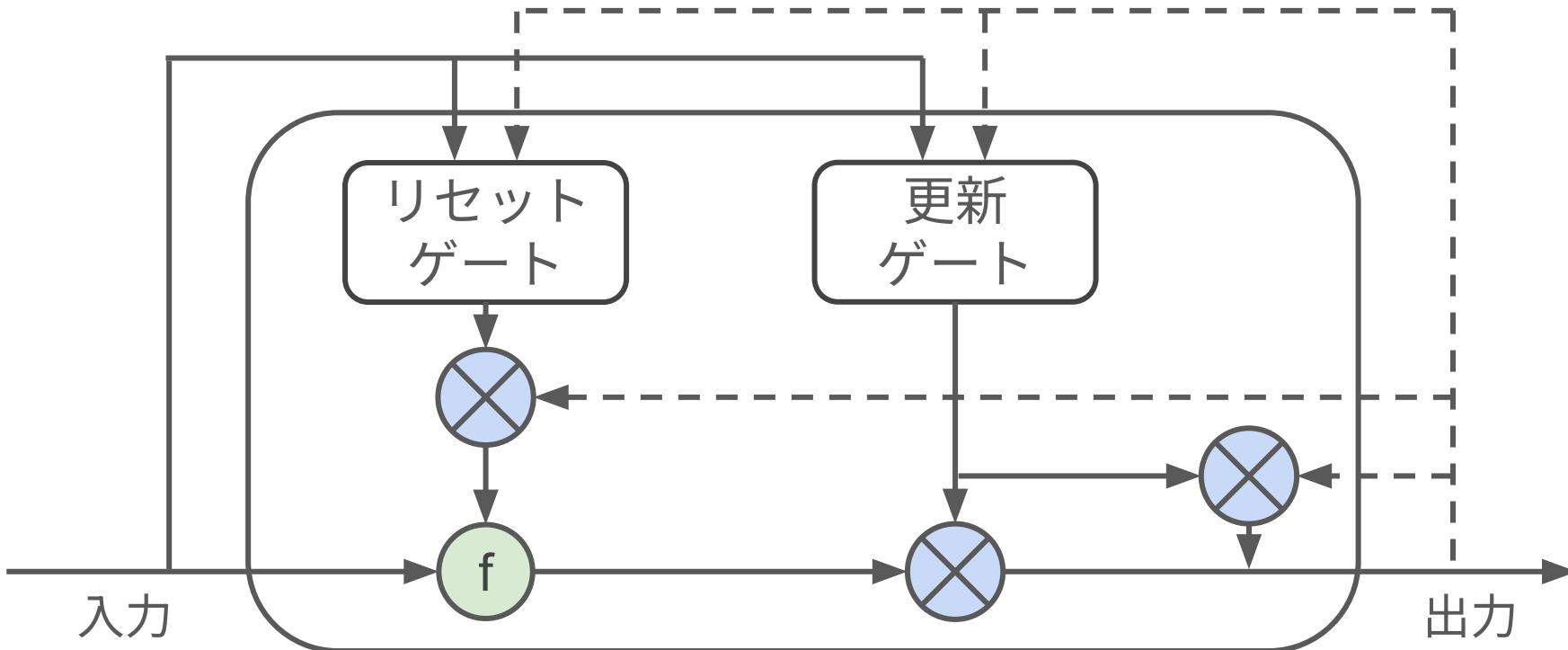
活性化関数

———— 現在の時刻のデータ



要素積

----- 1つ前の時刻のデータ



入力

出力



双向RNN エンコーダ-デコーダ

| 双方向RNNとエンコーダ-デコーダ

- 双方向RNN（Bidirectional RNN、BiRNN）

過去のデータだけではなく、未来のデータからも

学習・予測できるようにしたモデルのこと

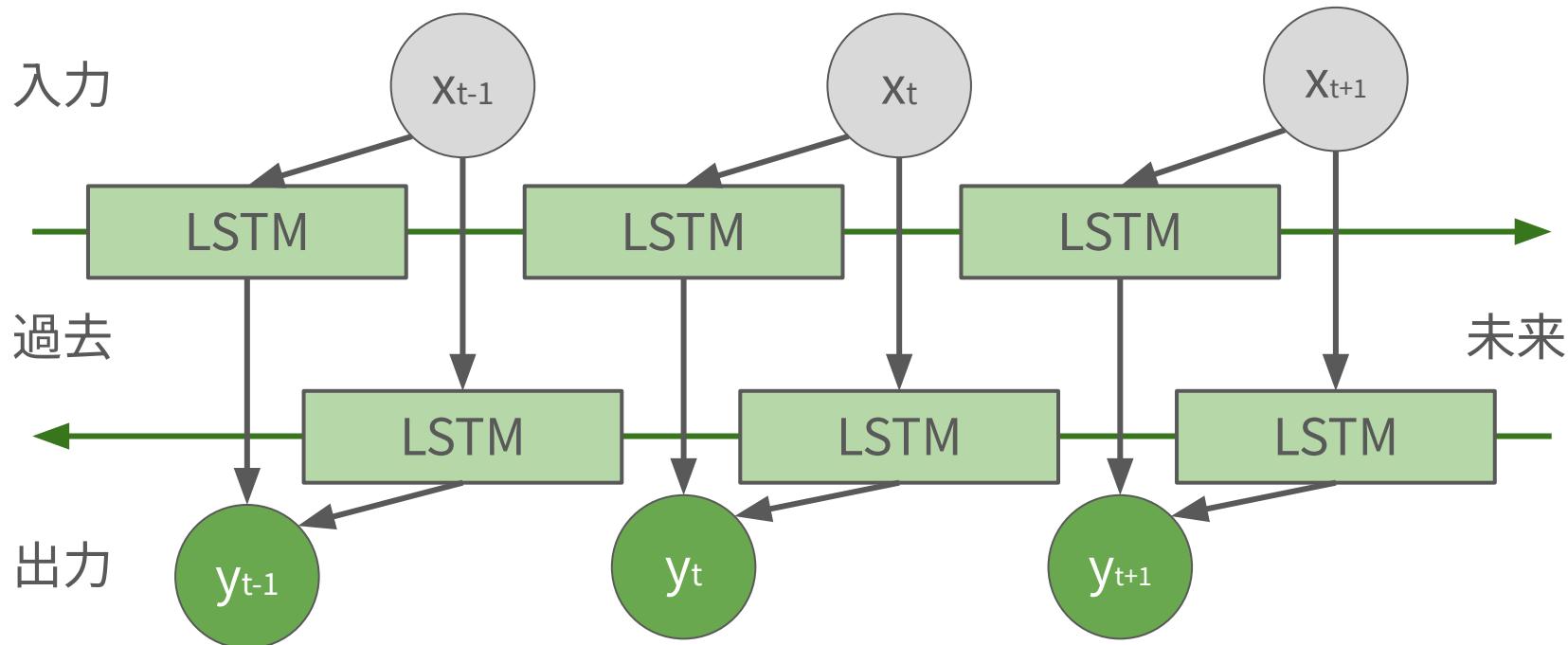
→一般的なRNNは未来のデータを予測に使用していなかった

→自然言語処理の場合なら、過去の単語だけではなく

未来の単語からも意味を推測した方が精度は高くなる

双向RNNとエンコーダ-デコーダ

- 双向RNN (Bidirectional RNN、 BiRNN)



| 双方向RNNとエンコーダ-デコーダ

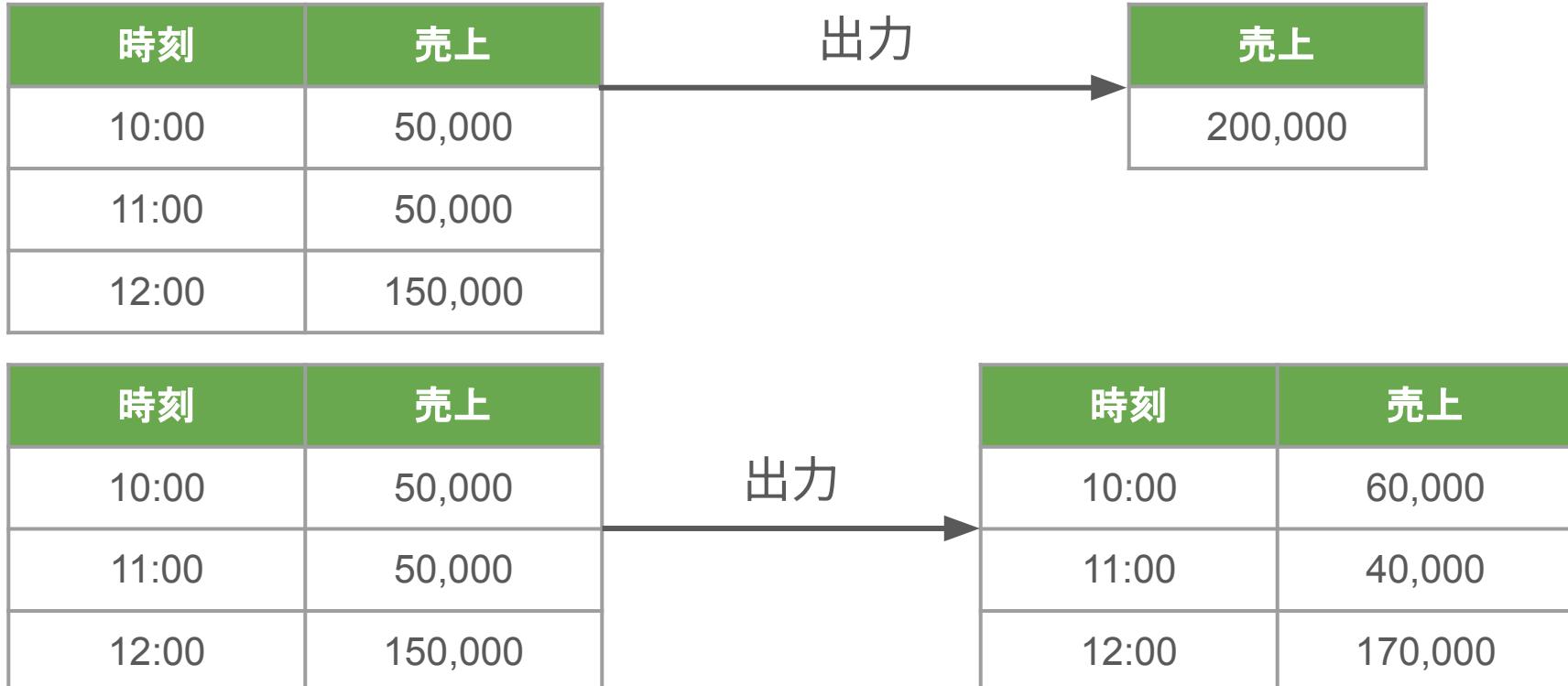
- Seq2Seq (Sequence To Sequence)

今までのRNNは時系列データを入力すると出力が1つだった
→出力も時系列データとして予測したいことがある

→時系列データから別の時系列データに変換する

モデルをSeq2Seq (Sequence To Sequence) という

双向RNNとエンコーダ-デコーダ



| 双方向RNNとエンコーダ-デコーダ

- RNN エンコーダ-デコーダ

Seq2Seqを実現するモデルの1つである

→エンコーダとデコーダの2つのRNNを組み合わせたもの

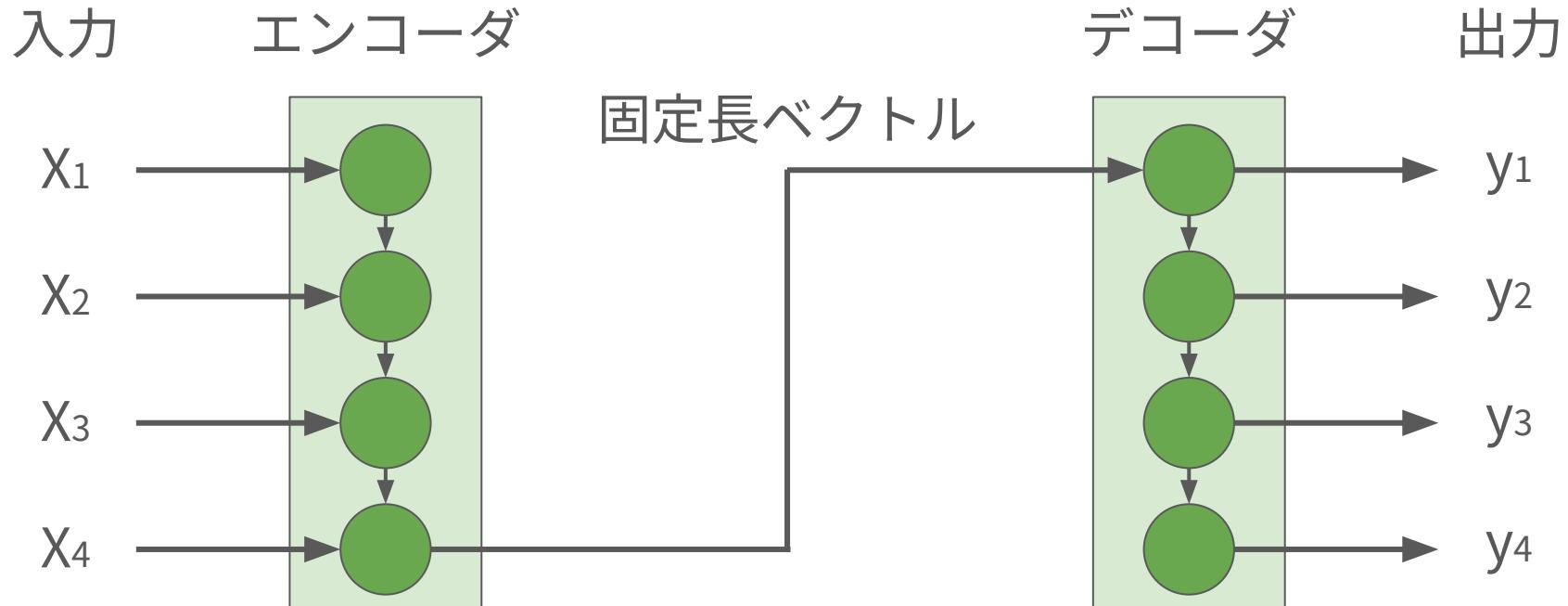
→機械翻訳などでよく使われる手法である

→エンコーダで入力された時系列データを固定長のベクトルに変換

デコーダで固定長のベクトルから時系列データを出力

双向RNNとエンコーダ-デコーダ

- RNN エンコーダ-デコーダ



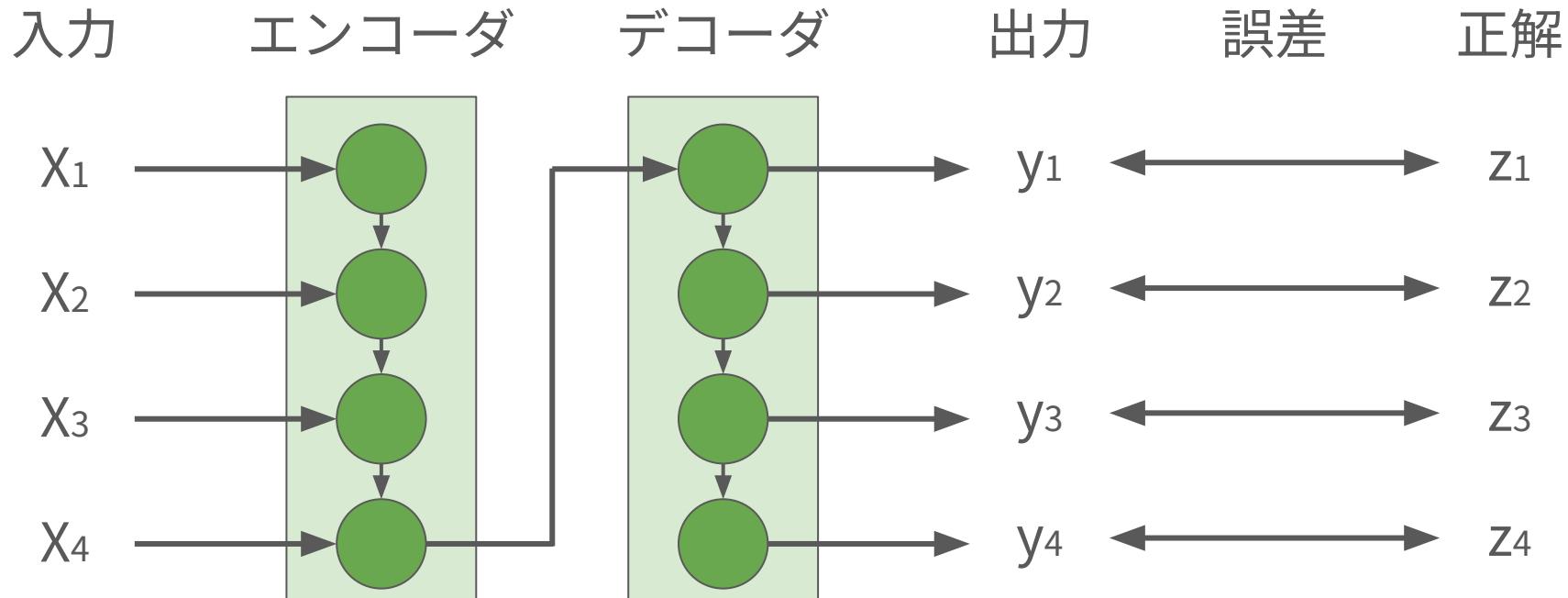
| 双方向RNNとエンコーダ-デコーダ

- 問題点

文章を生成しているとき、前に出力した単語に合う単語を出力
→不適切な単語が出力されると、その単語に合う単語が
出力されてしまい、求めていた文章と異なる文章が生成される
→誤差が大きくなってしまうため、学習が不安定になる

双向RNNとエンコーダ-デコーダ

- 問題点



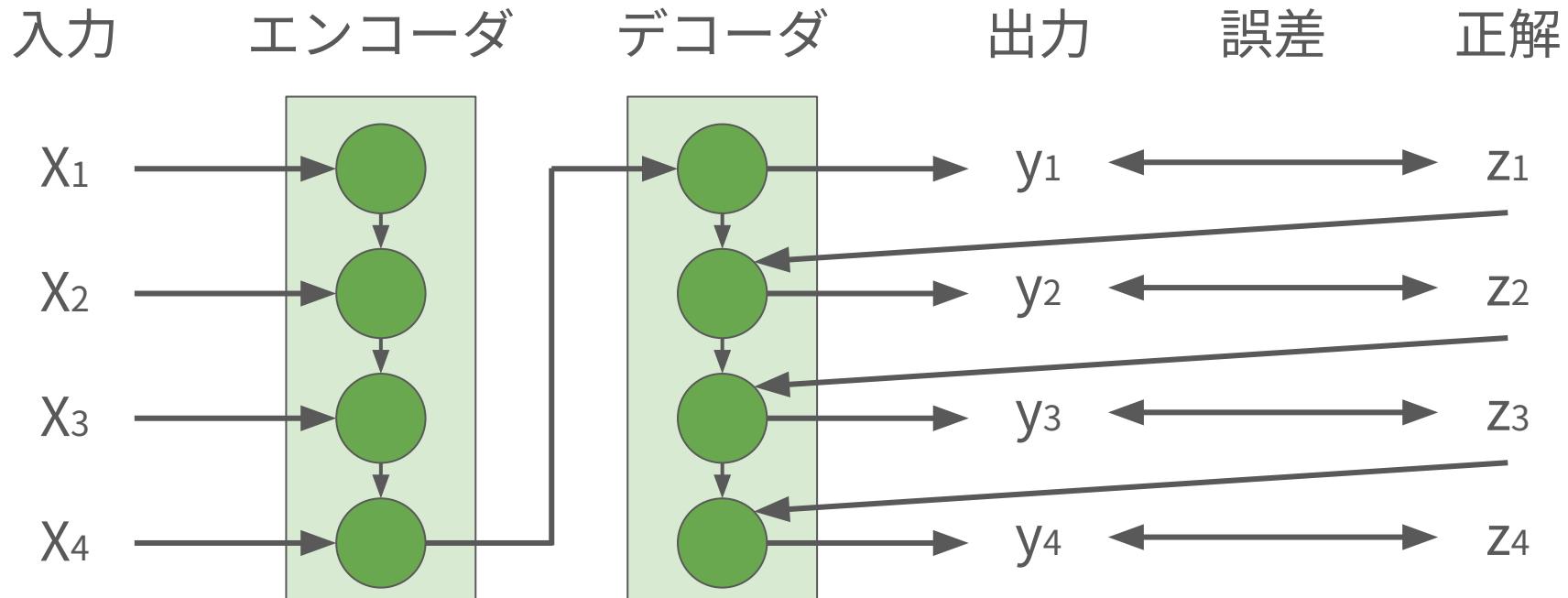
| 双方向RNNとエンコーダ-デコーダ

- 問題点

- 学習時、前の時刻の正解データを現時点の入力データとして使用することで学習を安定させることができる
→前の時刻の出力が間違っていても、前の時刻の正解データを使用するため誤差が大きくなりすぎないため
→以上のような手法を教師強制という

双向RNNとエンコーダ-デコーダ

- 問題点



Attention



Attention

- RNN エンコーダ-デコーダ

エンコーダで入力された時系列データを固定長のベクトルに変換

デコーダで固定長のベクトルから時系列データを出力

→長い時系列データになると固定長のベクトルだと

適切に情報が入りきらないという問題が起きてしまう

→長文の文章を扱う場合、精度が悪化してしまう

Attention

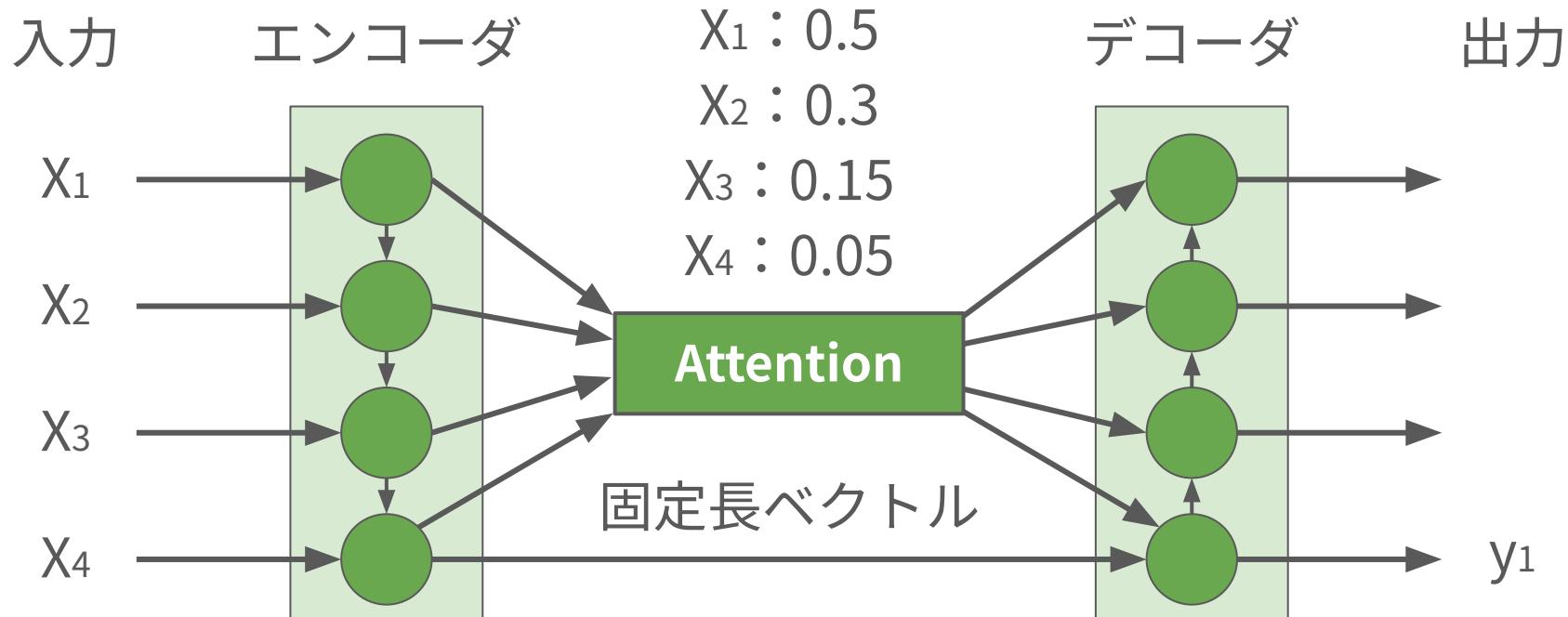
- RNN エンコーダ-デコーダ

ある時刻の状態が他の時刻の状態にどの程度影響を与えるか分からぬという問題も存在する

- 出力するときどの入力データが予測で重要なのかを重み付けして問題を改善しようとした（長文翻訳の精度が向上）
- 重み付けを行う機構を**Attention**機構という

Attention

- RNN エンコーダ-デコーダ



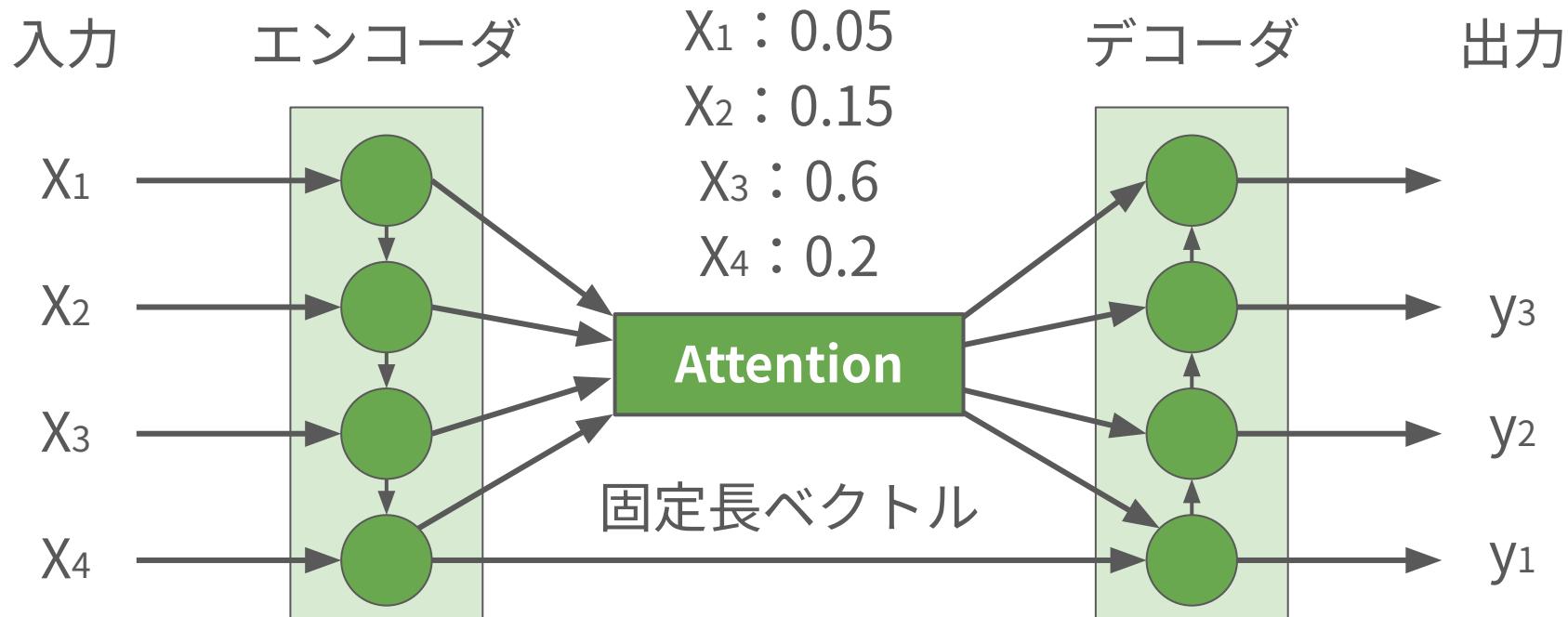
Attention

- RNN エンコーダ-デコーダ



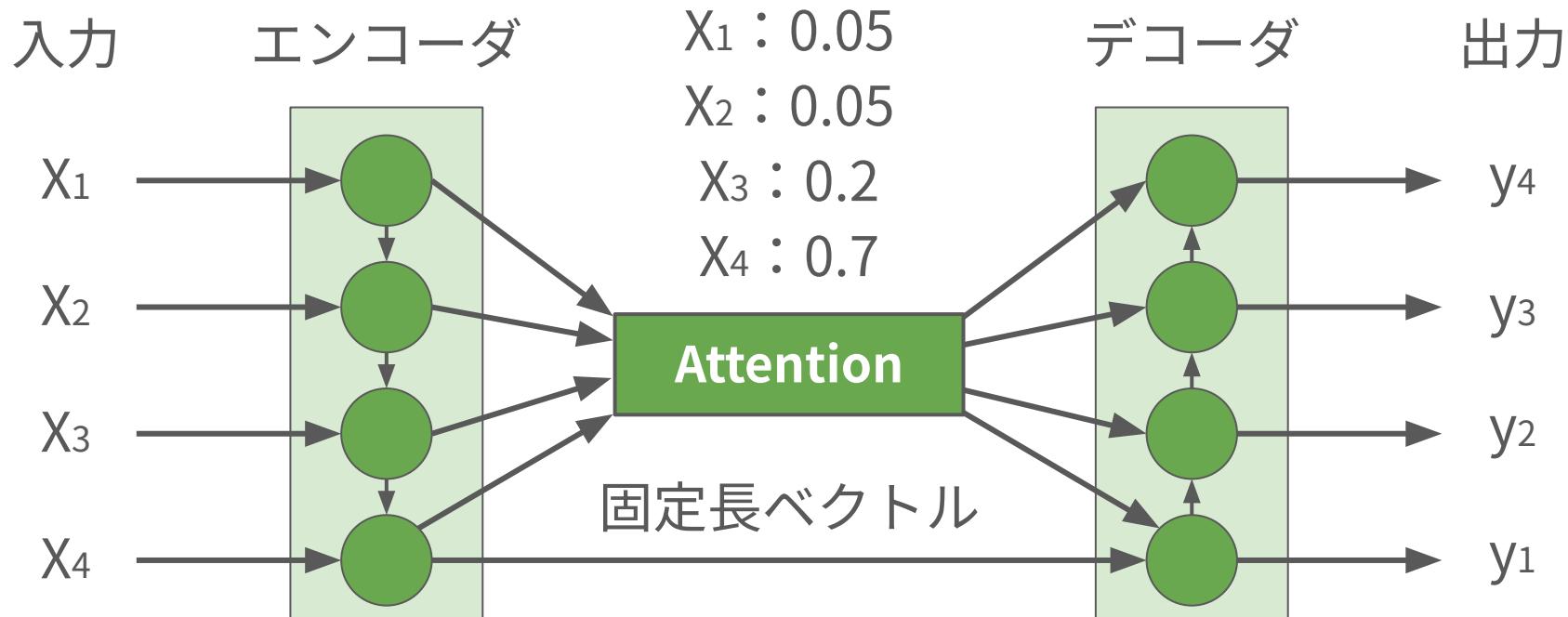
Attention

- RNN エンコーダ-デコーダ



Attention

- RNN エンコーダ-デコーダ



Attention

- Attention

機械翻訳、Neural Image Captioning (NIC) などで使用

- Image Captioning

入力した画像に対して、画像の説明文を生成するタスクのこと
→人が座っている画像を入力したら、

「人が座っています」というように説明文を生成する

Attention

- Neural Image Captioning (NIC)

ニューラルネットワークを使用したImage Captioning

→Neural Image Captioningはマルチタスク言語モデルの代表例

- マルチタスク言語モデル

複数のタスクを同時にこなすことができる言語モデルのこと

→NICは画像処理モデルと言語モデルを組み合わせたモデル

Attention

• Attention

RNNによって時系列データを高い精度で扱えるようになった
→並列的に計算することができないため処理速度が遅い

→入力データが長くなると単語間の関係性を
正しく反映させることが難しかった
→以上の問題を解決する方法としてトランスフォーマーが提案

トランスフォーマー



| トランスフォーマー

- トランスフォーマー

RNNを使用せず、Attention機構で構成されている

エンコーダ・デコーダモデルのこと

→RNNを使用しないため並列処理が可能になり

データの処理速度や学習速度を高速化することに成功した

→離れた単語同士の関係もうまく捉えることができるようになった

| トランスフォーマー

- トランスフォーマー

Source-Target Attentionと**Self-Attention**が使用

- **Source-Target Attention**

入力文と出力文の単語の関連度を計算するAttentionのこと

→**Encoder-Decoder Attention**とも呼ばれている

入力文をSource、出力文をTargetという

| トランスフォーマー

- **Source-Target Attention**

デコーダは生成した単語と入力文の単語との関連度を計算することができ、より適切な単語を生成することが可能

→デコーダは入力文全体の情報を利用して出力をすることができる

→入力文と出力文を繋ぐ役割を果たしていると言える

| トランスフォーマー

- **Self-Attention (自己注意機構)**

入力文内、出力文内の単語間の関連度を計算するAttention構造
→入力文内の単語間の関連度は、並列計算が可能で、計算が速い
単語同士の計算であるため、他の影響を受けないため

→This is a red pen. の場合、各単語は独立しているので、
「this」と「is」、「red」と「pen」の関連度を同時に計算可能

| トランスフォーマー

- **Self-Attention (自己注意機構)**

直前の単語などに基づいて単語を1つずつ出力していく
→出力時点では、出力文全体を手に入れることはできない特徴

→出力文内の単語の関連度は途中までの出力文と
入力文全体の情報を利用して単語の出力をしながら再計算

| トランスフォーマー

- Self-Attention（自己注意機構）

再計算を行う理由は、生成される単語によって、
単語間の関連度は変化していくため

→直前の単語などに基づいて単語を1つずつ出力していくため
並列計算を行うことはできないという特徴がある

| トランスフォーマー

- Self-Attention（自己注意機構）

単語間の関連度は計算できるが、単語の位置の考慮が不可能

→ 単語に位置関係の情報を与える処理である

位置エンコーディングにより位置情報を保有することが可能

| トランスフォーマー

- **Multi-Head Attention**

トランスフォーマーには複数のAttentionで重み付けを行う

Multi-Head Attentionという仕組みがある

→複数の視点で単語などに重み付けを行うことができるため

多様な情報に基づいた重み設定が可能になる

→文脈、品詞、意味的な関係などに基づいて重み付けが行われる

| トランスフォーマー

- **Multi-Head Attention**

各Attentionは独立して計算可能であるため、

並列計算が可能であり、計算速度が速い

→ 1つのAttentionだけでは多様な視点から重み付けができるないため、複数のAttentionで重み付けが行われる

| トランスフォーマー

- トランスフォーマー

Query (クエリ) 、 **Key** (キー) 、 **Value** (バリュー) を
使用して重要度の計算を行っている

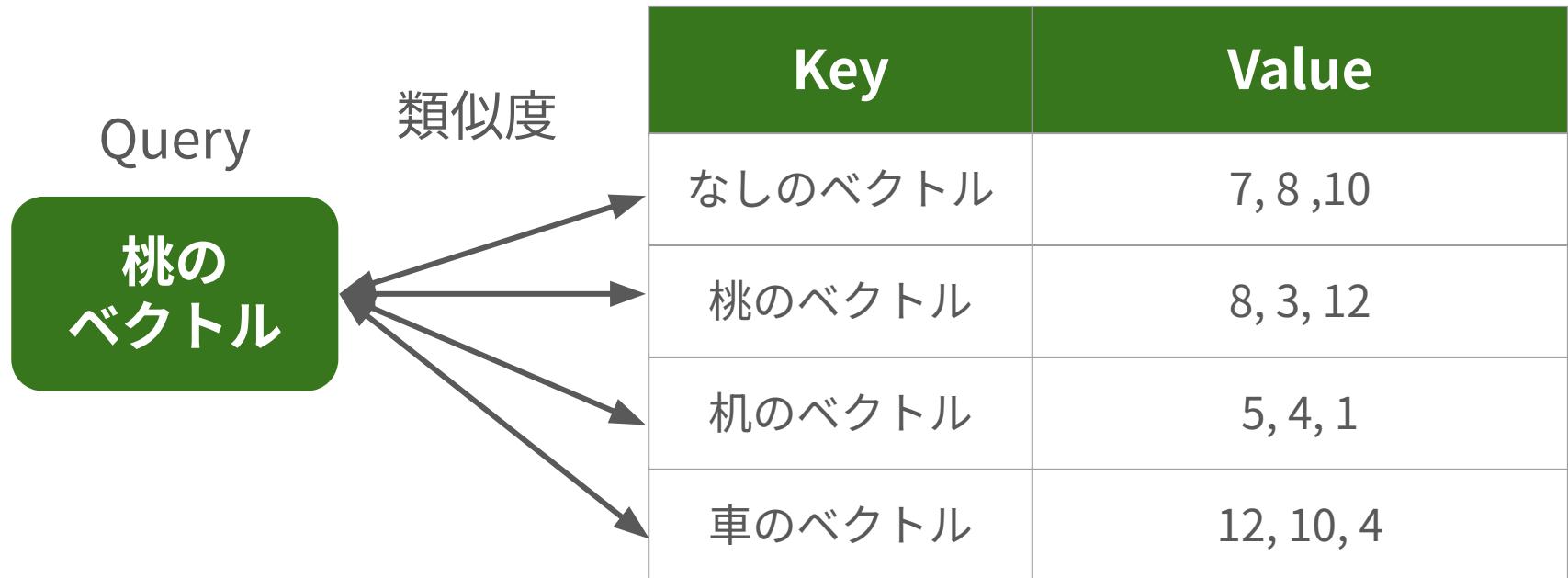
Query (クエリ) : 問い合わせる単語ベクトルなど

Key (キー) : 問い合わされる単語ベクトルなど

Value (バリュー) : Attentionの重み付けなどに利用される値

| トランスフォーマー

- トランスフォーマー



| トランスフォーマー

- トランスフォーマー

Query (クエリ) と Key (キー) の類似度を計算

→類似度が高いと重要度が高いと判断する

各Key (キー) に対する重要度が設定される

→重要度と Value (バリュー) からどの程度出力に影響を与えるか

計算して求める (最終的な出力に大きな影響を与える)

| トランسفォーマー

- トランسفォーマー

Query (クエリ) 、 **Key** (キー) 、 **Value** (バリュー) は
学習を通して算出される値である

→トランسفォーマーは数式を使用しないと

理解しにくい部分が多いので、用語とキーワードを押さえる

→「Multi-Head Attention」なら「複数視点」「並列計算」



転移学習と ファインチューニング

| 転移学習とファインチューニング

・転移学習

ある領域で学習させたモデルを他の領域で活用する手法

→学習済みモデルを活用することで、

モデル構築コストを削減することができる

→転移学習の場合、データが少なくとも

ある程度精度の高いモデルを構築することができる

転移学習とファインチューニング

・転移学習

学習済みのモデルの後ろに新しいモデルを追加したり、
出力層に近いパラメータのみを更新したりする

→モデルの特徴として、入力層付近は**全体的な特徴**を捉え、
出力層付近は**個別の特徴**を捉えているため、
出力層付近に層を追加したりすることでモデルの転用が可能

| 転移学習とファインチューニング

・ ファインチューニング

ある領域で学習させたモデルを他の領域で活用するために
全てのパラメータを学習させる手法のこと

→転移学習は一部のパラメータを学習させたりして
他の領域でモデルを活用することを示すことが多い

→画像認識などの分野では学習済みモデルが多く公開されている

転移学習とファインチューニング

・ 繙続学習

モデルの運用後などに、分類するクラスを増やしたい、新しい情報に対応できるように追加データを学習させたいという出来事が起きた場合に、データを学習させること

→新しく学習した結果、今まで出来ていたタスクの精度が悪化してしまう**破壊的忘却**が起きないように学習する必要がある

| 転移学習とファインチューニング

- 学習方法

Few Shot Learning : 少量のデータから学習すること

One Shot Learning : 1つのデータから学習すること

Zero Shot Learning : データを示さずに指示だけを与える方法

→モデルや状況によって、パラメータを変更する場合もあれば、
パラメータを変更しない場合もある

| 転移学習とファインチューニング

- データを与える学習

動物を分ける学習済みモデルに新しい動物の写真を複数枚与え、学習させることを **Few Shot Learning**
→ 1枚の写真を与える場合は **One Shot Learning**、
→ 画像を与えず、新しい動物のテキストなどの補足情報を与え、
学習させるのが **Zero Shot Learning**（論文ごとに定義が異なる）

| 転移学習とファインチューニング

・例題を与える学習

AIを使用して日本語から英語に翻訳するときに、

「りんご」は「apple」、本は「book」というように
複数の例を加え、学習させることを**Few Shot Learning**

→例が1つの場合は**One Shot Learning**、

例を示さずに翻訳させる場合は**Zero Shot Learning**である

| 転移学習とファインチューニング

- メタ学習

学習の仕方を学習する手法のこと

→以前に行った学習から、最適なパラメータの初期値などの
知識を獲得することで、その知識を使って
効率的に学習を行うことができる

→メタ学習の代表的な手法に**MAML**がある

| 転移学習とファインチューニング

- 自己教師あり学習 (SSL)

正解ラベルが付与されていないデータを使用して、
擬似的な問題（プレテキストタスク）を通して学習していく手法
→画像の一部を隠して復元、文の次の単語予測 など

→学習したモデルを下流タスクに合わせて
ファインチューニングを行うことで特定タスクを解くことが可能