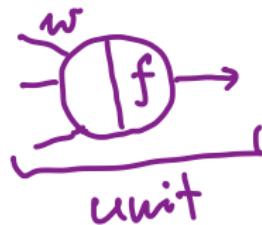


MLP/RNN/CNN

本谷 秀堅

名古屋工業大学

ニューラルネットワークのアーキテクチャ



ニューラルネットワークのアーキテクチャ：ユニットの結線の構造

データの性質に依存して適切なアーキテクチャが変わる。

ニューラルネットワークの基本的なアーキテクチャ

- 多層パーセプトロン

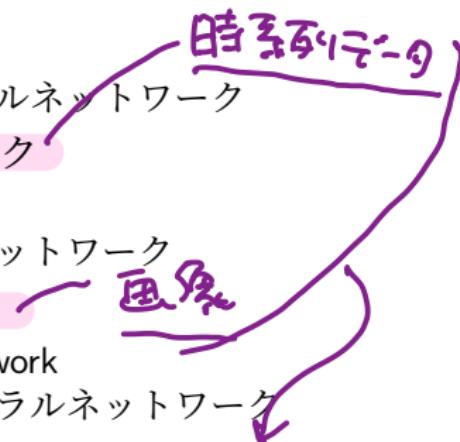
- MLP: Multi-Layer Perceptron
- 全結合層を積み重ねたニューラルネットワーク

- リカレントニューラルネットワーク

- RNN: Recurrent Neural Network
- 内部に閉路を持つニューラルネットワーク

- 畳み込みニューラルネットワーク

- CNN: Convolutional Neural Network
- 畳み込み層を積み重ねたニューラルネットワーク



- トランスフォーム

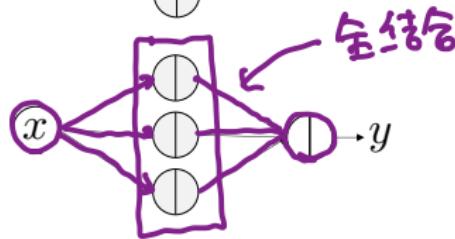
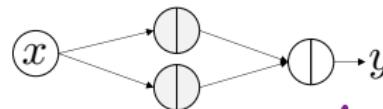
多層パーセプトロン

多層パーセプトロン (MLP: Multi-Layer Perceptron)

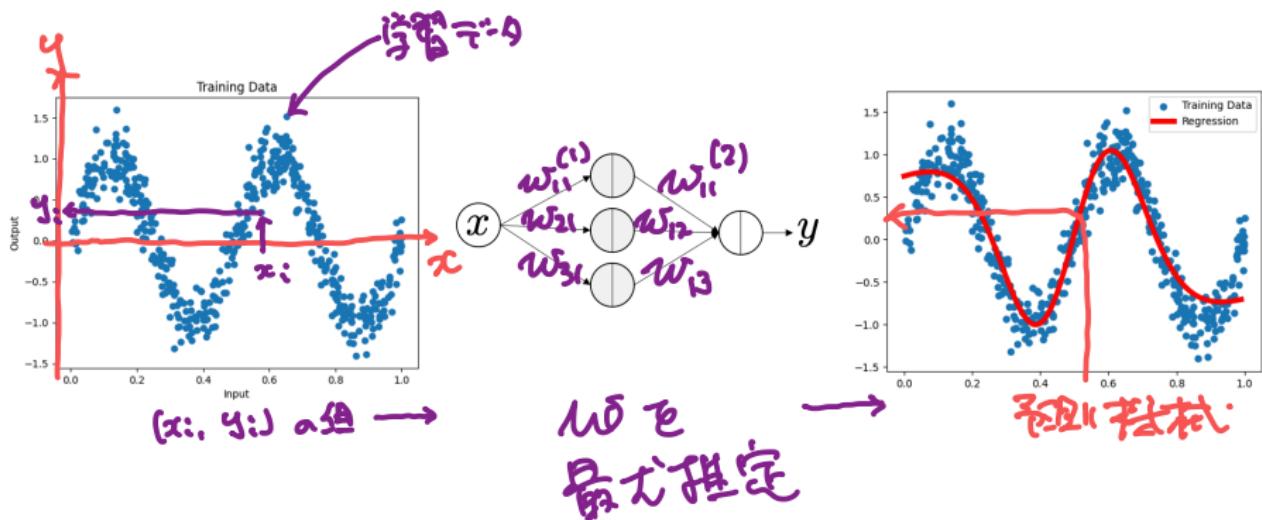
- 全結合層を積み重ねたニューラルネットワーク

MLP に関する：どのくらい複雑な関数を表現できるか

例：スカラを入力して、スカラを出力する 3 層パーセプトロン： $y = f(x)$



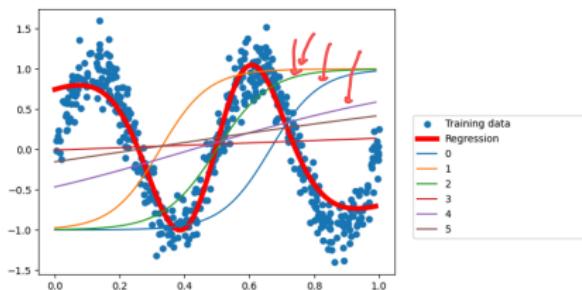
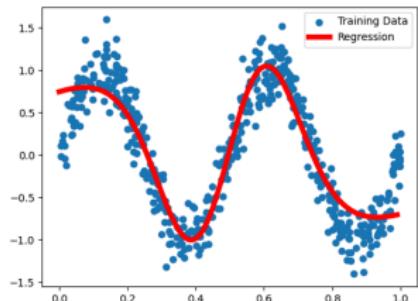
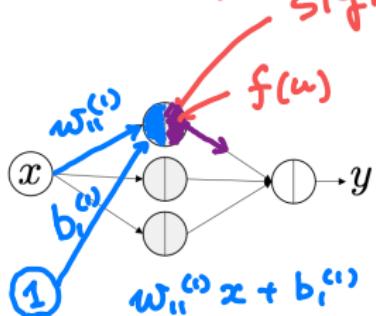
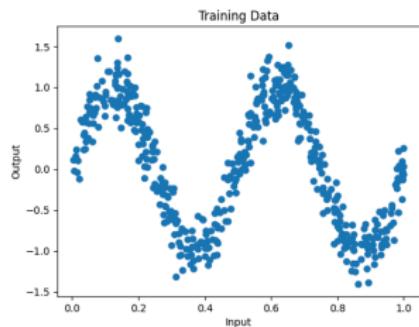
(以下の例では中間層の活性化関数はシグモイド関数、出力層は恒等関数とする)

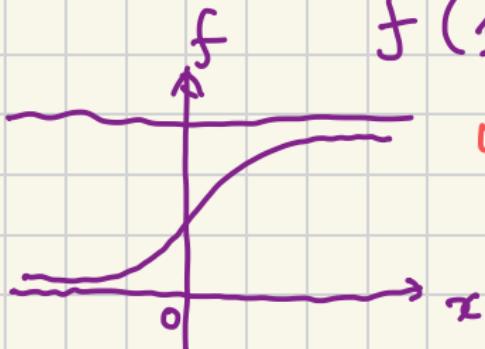


演習 (Google Colaboratory)

中間層の各ユニットが異なる基底関数になる

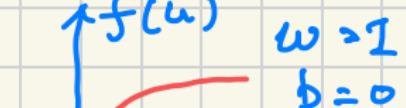
例 sigmoid 関数 + εε -





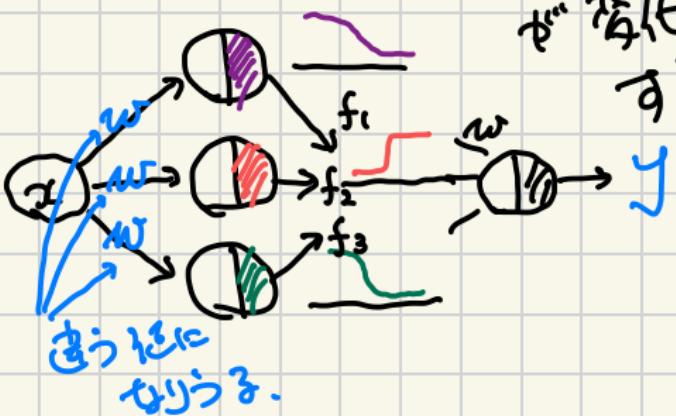
$f(\underline{w}^T x + b)$ がまた
 $f(u)$ へ

$$u = w^T x + b$$



$$\begin{aligned}w &= -1 \\b &= 0\end{aligned}$$

$$w=1$$



陰陽五行

文化

१७८

3

الآن
لهم

$$y = \sum_{i=1}^3 w_i^{(2)} f_i(x)$$

$y(x)$ を 基底関数表現

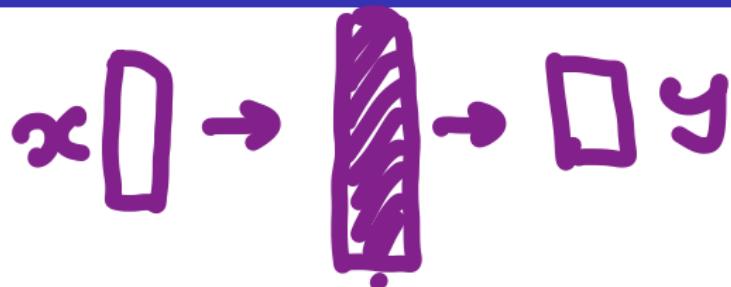
- ・ リーニング基底 Θ_1
- ・ 多項式基底 Θ_2
- ・ ステラル Θ_3
- ⋮

学習で
変化する。

$y = f(x)$ を
希望どおりに
表現するか?

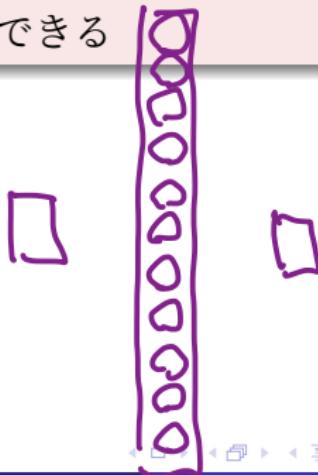
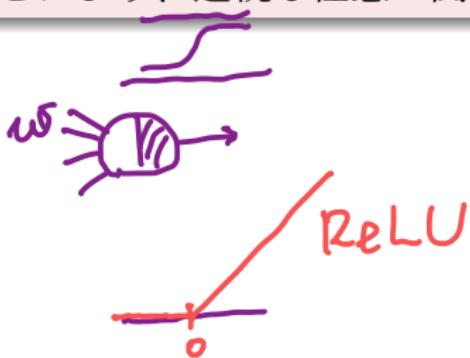
$\sin \omega x$
 $\cos \omega x$ } 任意の関数を
表現できる

万能関数近似定理 (Universal Approximation Theorem)

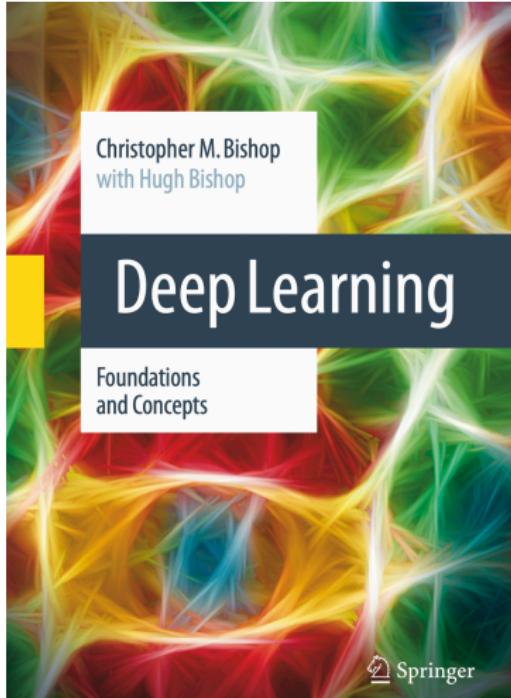


万能関数近似定理: Universal Approximation Theorem

(常識的な活性化関数を持つ) 中間層が1層あれば、ユニットの数を増やすことにより、連続な任意の関数を近似できる



以下、本資料の多くの図を下記の書籍に依っています。



Bishop, "Deep Learning: Foundations and Concepts", Springer, 2023

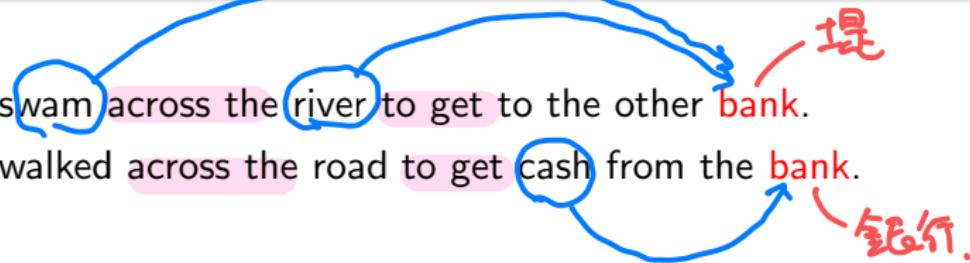
言語データと RNN

言語データと RNN

言語データ

- シンボルの系列
- 離れたシンボル間に強い関係
- シンボル間の非対称な関係（例：対義語・類義語・包含関係）

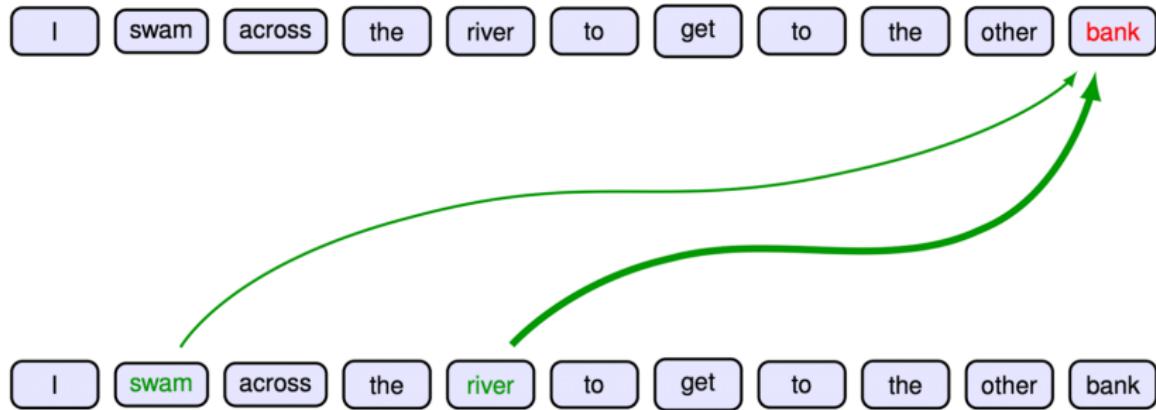
例

- I swam across the river to get to the other bank.
 - I walked across the road to get cash from the bank.
- 

言語データと RNN

言語データ

- シンボルの系列
- 離れたシンボル間に強い関係
- シンボル間の非対称な関係（例：対義語・類義語・包含関係）

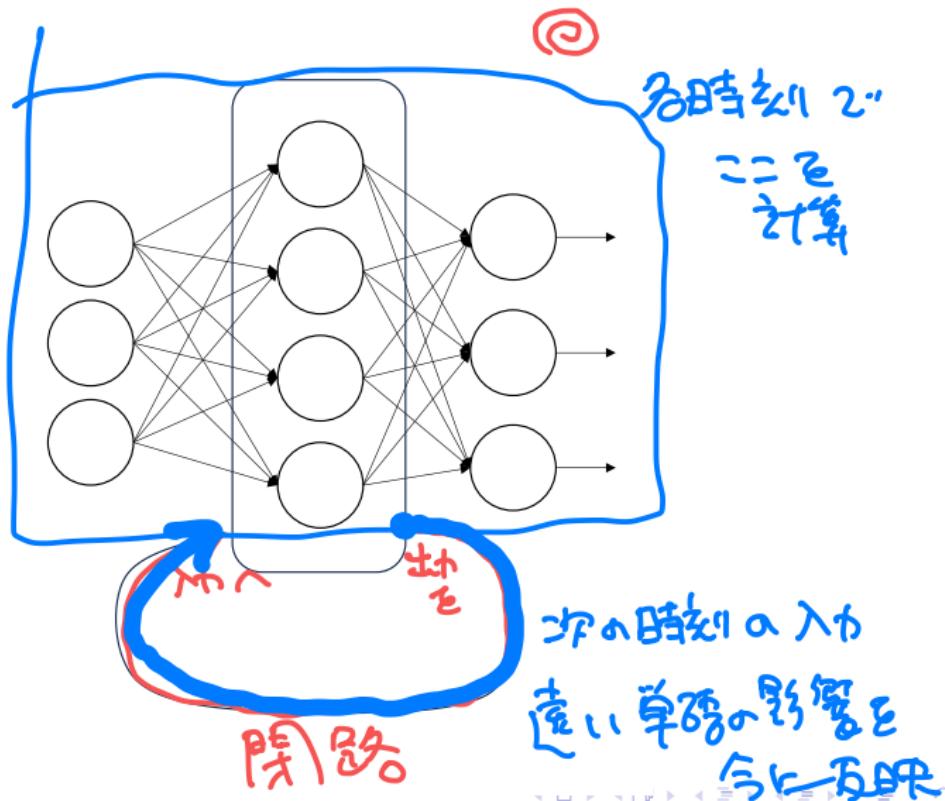




- リカレントニューラルネットワーク
 - RNN: Recurrent Neural Network
 - 内部に閉路を持つニューラルネットワーク

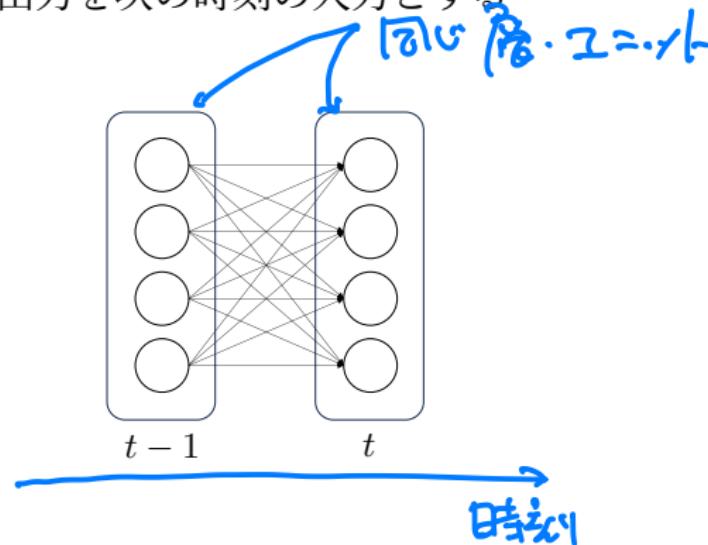
リカレントニューラルネットワーク (RNN)

内部に有向閉路を持つニューラルネットワーク

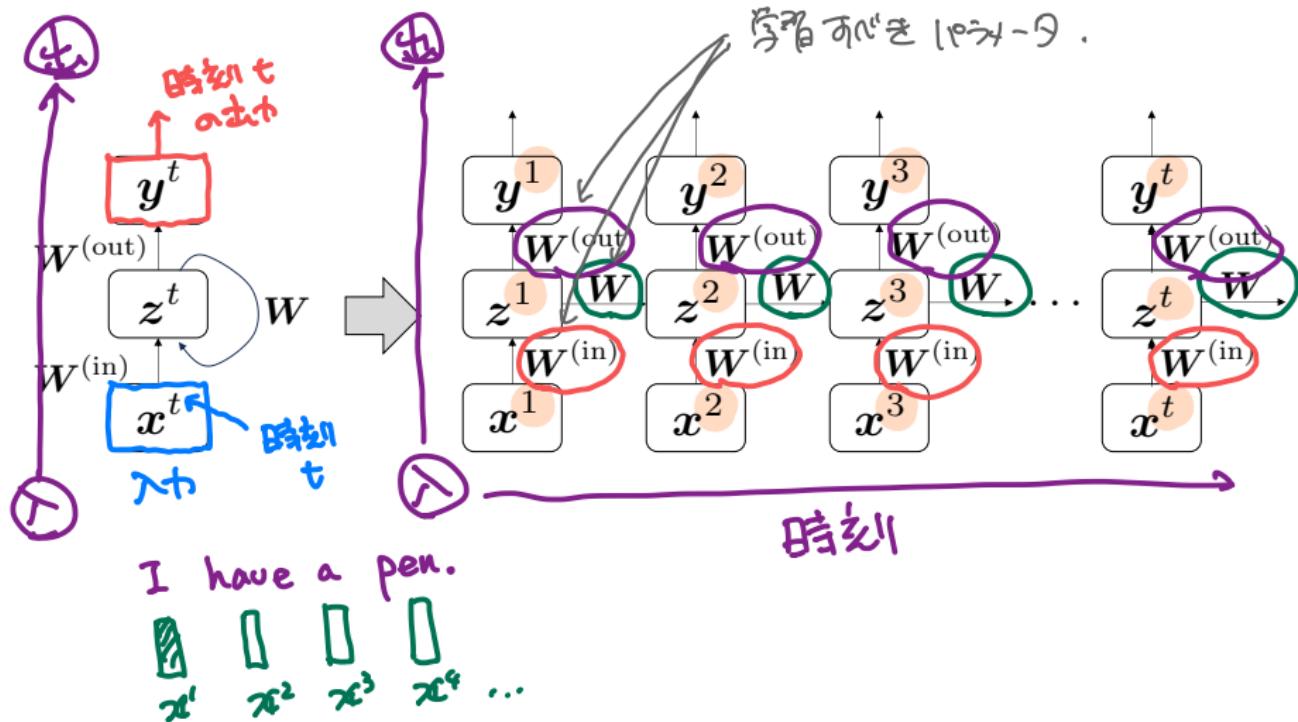


リカレントニューラルネットワーク (RNN)

帰還路：前の時刻の出力を次の時刻の入力とする



RNNのアーキテクチャ



RNNの入出力 (1/2)

- 入力 : x^1, x^2, \dots {時刻}.
- 出力 : y^1, y^2, \dots {時刻} 前から後ろ

中間層の計算は次のとおり ($z_j^0 = 0$ とする)

① 入力と重みとの内積

$$u_j^t = \sum_i w_{ji}^{(in)} x_i^t + \sum_{j'} w_{jj'} z_{j'}^{t-1}$$

前の層の出力との内積
前の時刻の
自分の出力
との内積

過去の反映

② 活性化関数の適用

$$z_j^t = f(u_j^t)$$

重みを行列表記すると次式になる

$$z^t = f(W^{(in)}x^t + Wz^{t-1})$$

RNN の入出力 (2/2)

出力層の計算は次のとおり：

- ① 出力層で重みと中間層からの出力の内積を計算：

$$v_k^t = \sum_j w_{kj}^{(\text{out})} z_j^t$$

- ② 次に活性化関数を適用して出力 y を得る：

$$y^t = f^{(\text{out})}(v^t) = f^{(\text{out})}(W^{(\text{out})}z^t)$$

RNNで解ける問題

- ① **T 入力、T 出力** : 訓練データ集合 \mathcal{D}

$$\mathcal{D} = \{[(\mathbf{x}_n^1, \mathbf{x}_n^2, \dots, \mathbf{x}_n^{T_n}), (\mathbf{d}_n^1, \mathbf{d}_n^2, \dots, \mathbf{d}_n^{T_n})] \mid n = 1, 2, \dots\}$$

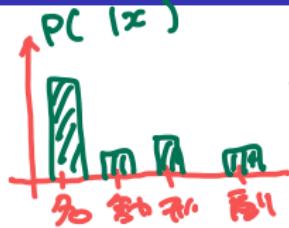
例: 単語列(文章)を入力 → 各単語の品詞を出力

- ② **T 入力、1 出力** : 訓練データ集合 \mathcal{D}

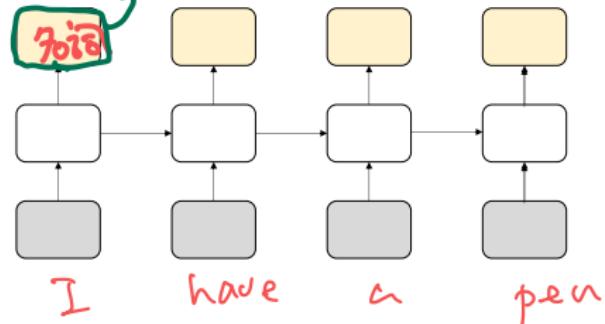
$$\mathcal{D} = \{[(\mathbf{x}_n^1, \mathbf{x}_n^2, \dots, \mathbf{x}_n^{T_n}), \mathbf{d}_n] \mid n = 1, 2, \dots\}$$

例: 単語列(文章)を入力 → 文章の感情を出力

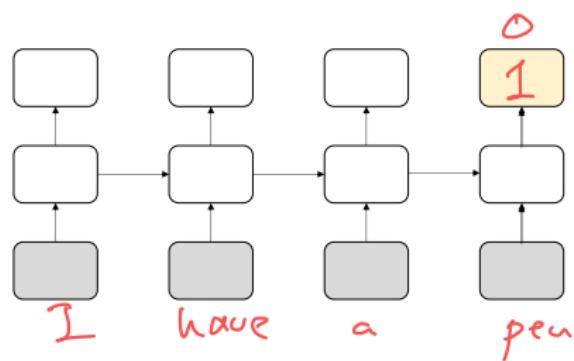
RNNの入出力



Soft max a
out



T 入力・ T 出力



T 入力・1 出力

T 入力・ T 出力損失関数

例. 各時刻へ單語 品詞

交差エントロピー (各時刻の出力を K クラス分類)

正解 y_k^t が 1
と 0

$$L(\mathbf{w}) = - \sum_{n=1}^N \sum_{t=1}^{T_n} \sum_{k=1}^K d_{nk}^t \log y_k^t(\mathbf{x}_n^t; \mathbf{w})$$

学習データ
各データを最初から最後まで

どの品詞が全切ため.

事後確率.

(例: $\mathbf{W}^{(\text{out})}$ は全結合・ $f^{(\text{out})}$ はソフトマックス関数)

T 入力・1 出力損失関数

交差エントロピー (最終時刻の出力を K クラス分類)

$$L(\mathbf{w}) = - \sum_{n=1}^N \sum_{k=1}^K d_{nk} \log y_k^{T_n}(\mathbf{x}_n^{T_n}; \mathbf{w})$$

↓
全時
間の
正解ラベルを
取得する

(例 : $\mathbf{W}^{(out)}$ は全結合・ $f^{(out)}$ はソフトマックス関数)

RNN の誤差逆伝播

復習：誤差逆伝播法

- $\delta_j^{(\ell)} \equiv \partial L / \partial u_j^{(\ell)}$

出力から
遠いと
計算易しくな...

- 層 $\ell + 1$ から ℓ への伝播

$$\delta_j^{(\ell)} = \sum_k \delta_k^{(\ell+1)} \left(w_{kj}^{(\ell+1)} f'(u_j^{(\ell)}) \right)$$

ひとつ上の層で計算して
伝播する

- 各層のパラメータによる微分

$$\frac{\partial L}{\partial w_{ji}^{(\ell)}} = \delta_j^{(\ell)} z_i^{(\ell-1)}$$

ひとつ下の層の
出力

乗算

RNN の誤差逆伝播

復習 : RNN

- 出力 : $y^t = f^{(\text{out})}(\mathbf{v}^t)$
- $\mathbf{v}_k^t = \sum_j w_{kj}^{(\text{out})} z_j^t$
- $z_j^t = f(u_j^t)$
- $u_j^t = \sum_i w_{ji}^{(\text{in})} x_i^t + \sum_{j'} w_{jj'} z_{j'}^{t-1}$

RNN の誤差逆伝播

出力層の更新用

Back-Propagation through time (BPTT) 法

- ① 出力層 : $\delta_k^{(out),t} \equiv \partial L / \partial v_k^t$
- ② 中間層 : $\delta_j^t \equiv \partial L / \partial u_j^t$

中間層用

- ③ 中間層の「デルタ」は次式で得られる ($\delta_j^{T+1} = 0$ とする)

$$\delta_j^t = \left(\sum_k w_{kj}^{(out)} \delta_k^{(out),t} + \sum_{j'} w_{j'j} \delta_{j'}^{t+1} \right) f'(u_j^t)$$



RNN の誤差逆伝播

玉手から伝去

Back-Propagation through time (BPTT) 法 (つづき)

- ① 各層のパラメータによる微分

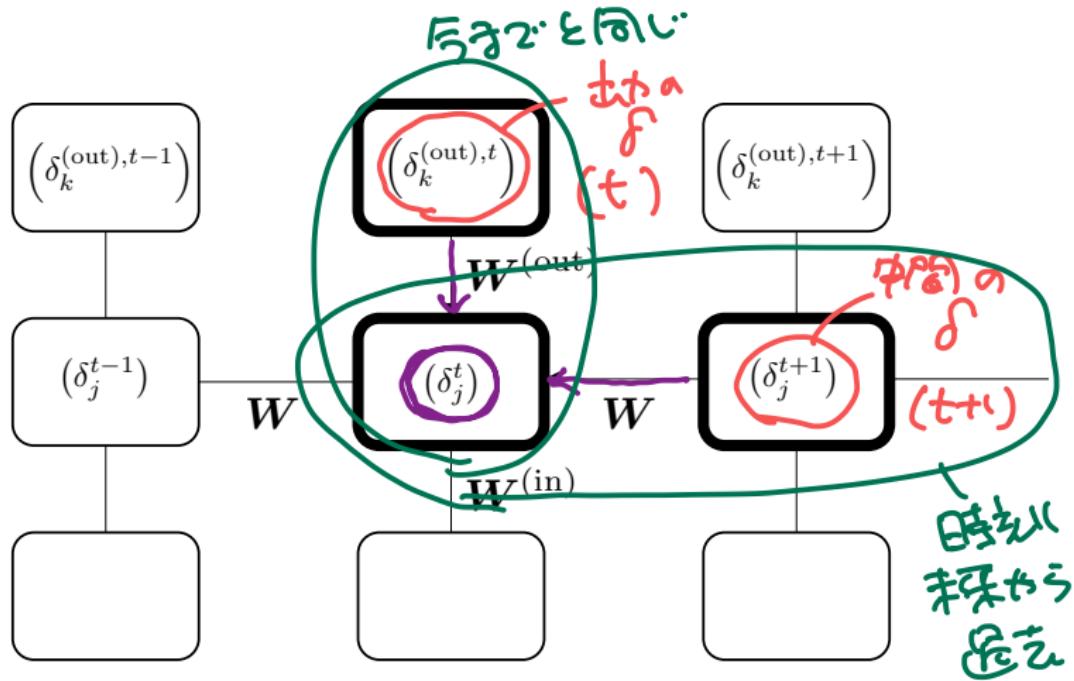
$$\frac{\partial L}{\partial w_{ji}^{(\text{in})}} = \sum_{t=1}^T \frac{\partial L}{\partial u_j^t} \frac{\partial u_j^t}{\partial w_{ji}^{(\text{in})}} = \sum_{t=1}^T \delta_j^t x_i^t$$

$$\frac{\partial L}{\partial w_{jj'}} = \sum_{t=1}^T \frac{\partial L}{\partial u_j^t} \frac{\partial u_j^t}{\partial w_{jj'}} = \sum_{t=1}^T \delta_{j'}^t z_{j'}^{t-1}$$

$$\frac{\partial L}{\partial w_{kj}^{(\text{out})}} = \sum_{t=1}^T \frac{\partial L}{\partial v_k^t} \frac{v_k^t}{\partial w_{kj}^{(\text{out})}} = \sum_{t=1}^T \delta_k^{(\text{out}),t} z_j^t$$

RNN の誤差逆伝播

Back-Propagation through time (BPTT) 法 (つづき)



言語データと RNN

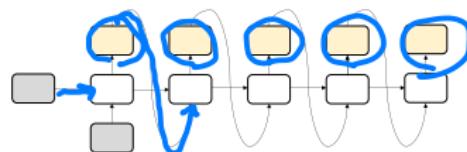
過去の影響を
すぐ消去する

RNN の性質

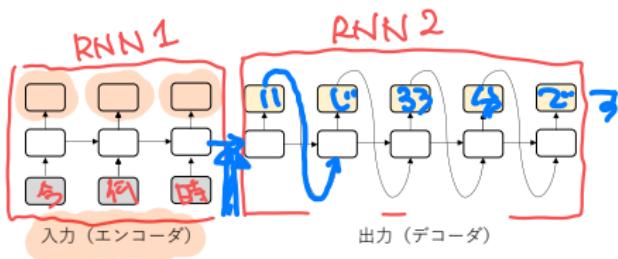
- 過去に受け取った入力すべてが現時刻の出力に影響を与える
- 万能関数近似の能力あり（中間層を十分に増やせば任意の系列から系列への写像を近似可）

RNNによる系列変換

- 入力 : $t - 1$ までの系列 (x^1, x^2, \dots, x^{t-1})
- 出力 : t 以降の系列 (x^t, x^{t+1}, \dots)



最初の時刻のみに入力



系列変換

系列変換でできること

次の応用はいずれも「系列変換」で実装可

- 機械翻訳
- 対話（チャットボット）
- 質問応答
- 文書要約
- ...

画像データと CNN

画像データとCNN

画像データ

- 画素値の配列
- 近傍の画素値間に強い関係
- 並進・回転に対する局所的不变性・同変性



180	180	50	14	54	10	33	48	106	159	181	
206	109	5	124	191	111	120	204	166	15	56	180
194	68	137	253	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	137	102	36	103	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	236	73	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	95	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	137	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	95	218

画像データと CNN

画像データ

- 画素値の配列
- 近傍の画素値間に強い関係
- **並進・回転に対する局所的不变性・同変性**



- 畳み込みニューラルネットワーク
 - CNN: Convolutional Neural Network
 - 畳み込み層を積み重ねたニューラルネットワーク

復習：畠み込み (Convolution)

信号 $u(x)$ と $h(x)$ の畠み込みは次式で表される

$$(u * h)(x) = \int u(\dot{s})h(x - \dot{s})d\dot{s}$$

離散の場合は次式になる

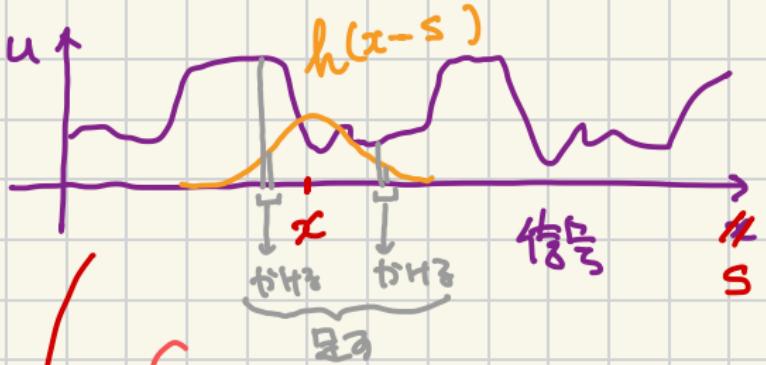
$$(u * h)(m) = \sum_i u(i)h(m - i)$$

$h'(m) = h(-m)$ で書き換えると、

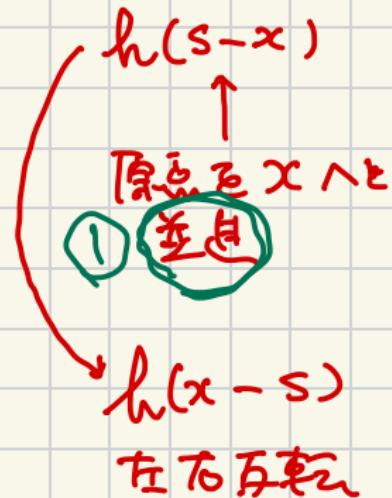
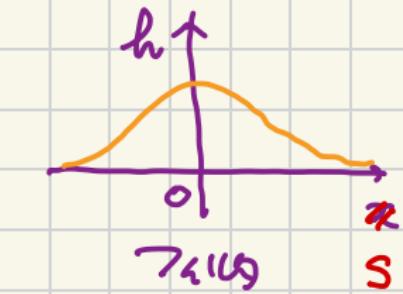
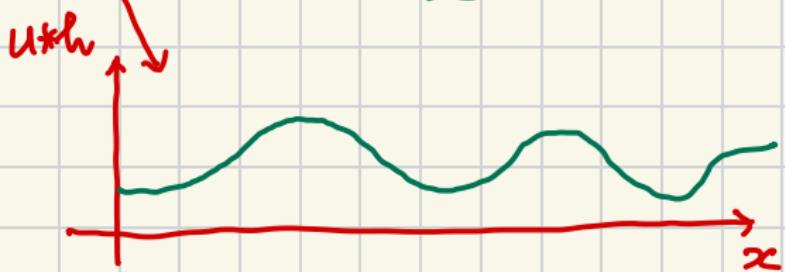
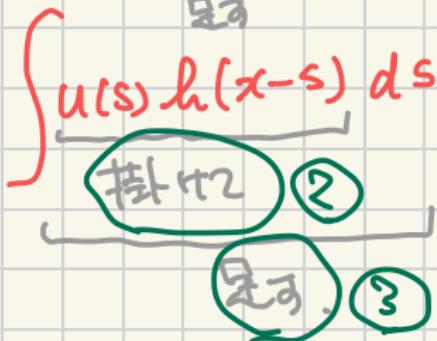
$$(u * h)(m) = \sum_i u(i)h'(i - m)$$

④学習をはじめます。

$h'(i - m)$ は $h'(i)$ を m だけ並進させた信号であり、 u と h' 両者の内積で表現されている。



$$\int u(s) h(x-s) ds$$



差分: $\frac{u(x+1) - u(x-1)}{2}$

$$+\frac{1}{2} \boxed{1} \boxed{1} \frac{1}{2}$$

畠み込みニューラルネットワーク

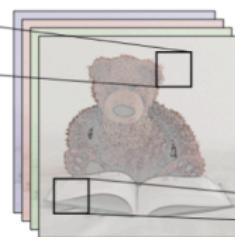
くり返す。

Convolutional Neural Network (CNN) の構造は概ね下記のとおり

入力 → [畠み込み層 → Pooling 層] → [全結合層] → 出力



Input image



Convolutions



Pooling

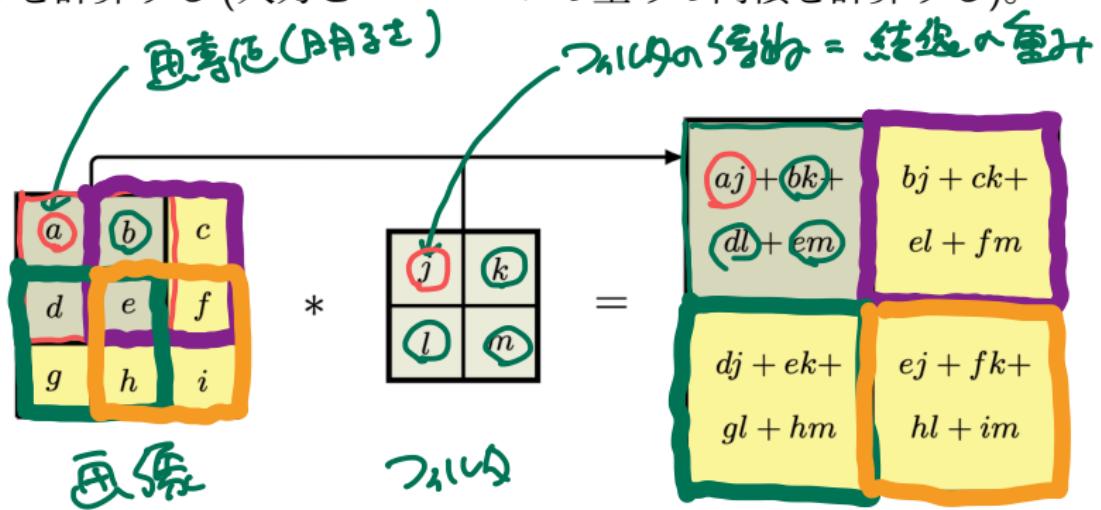


Fully Connected

各畠み込み層と全結合層は、内積計算のあと活性化関数を適用する

畳み込み層

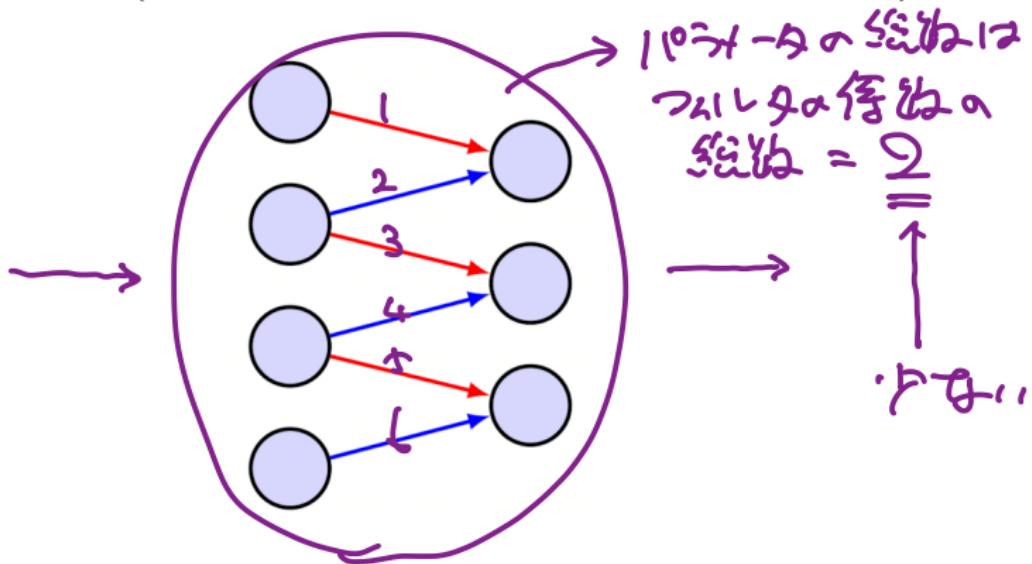
直前の層の出力を入力として受け取り、フィルタをスキャンさせながら畳み込みを計算する（入力とニューロンの重みの内積を計算する）。



フルタモズラル 内装

畳み込み層

直前の層の出力を入力として受け取り、フィルタをスキャンさせながら畳み込みを計算する(入力とニューロンの重みの内積を計算する)。

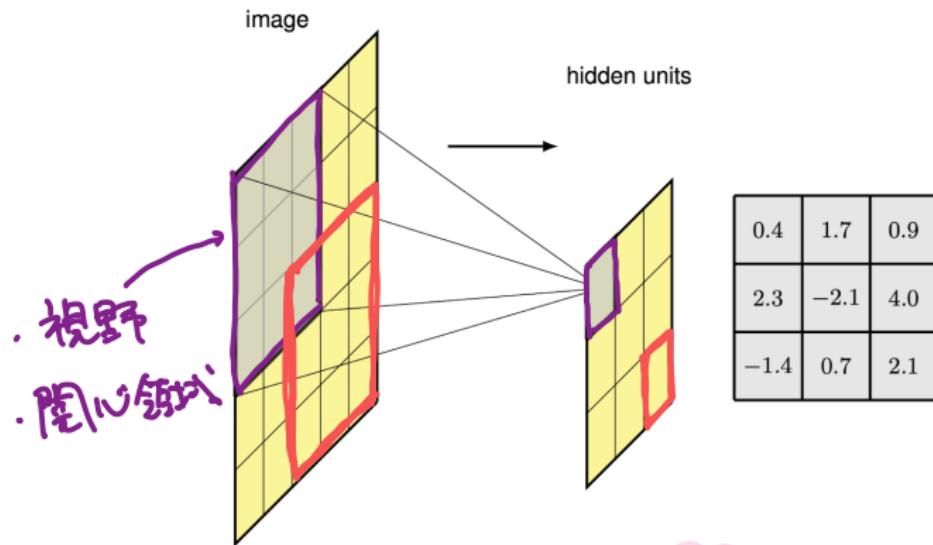


各ユニットで結合の重みは共通

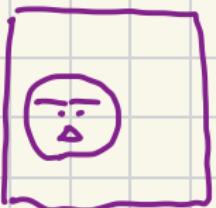
上記の例では6つの結線があるが学習するパラメータ数は2つ

畳み込み層

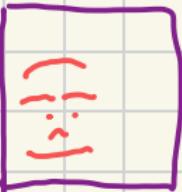
直前の層の出力を入力として受け取り、フィルタをスキャンさせながら畳み込みを計算する（入力とニューロンの重みの内積を計算する）。



並進同変性を有する



たて
よこ



特徴
(位置ごと)
(高さ)

入力

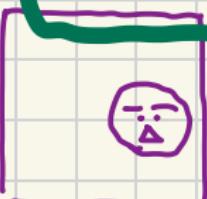
① 特徴
抽出

類

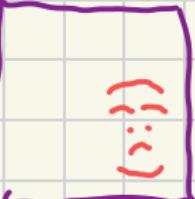
② 対応

①
対応
v2

並進



たて
よこ



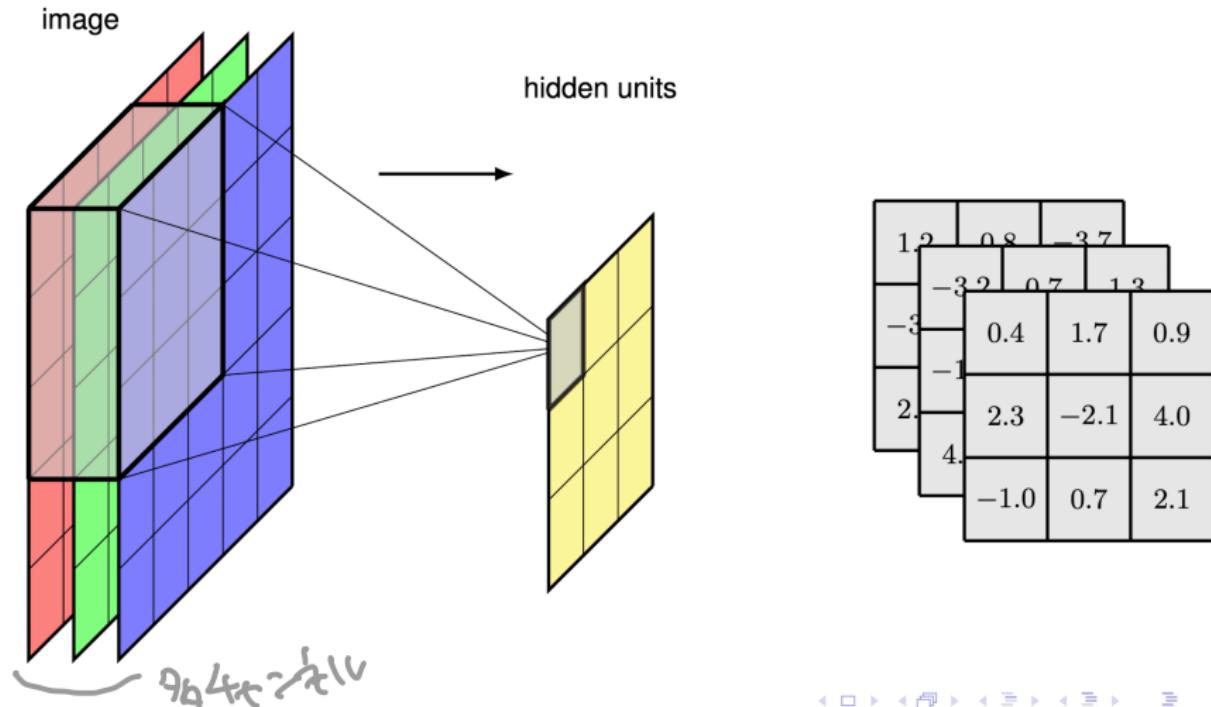
-
並進
対応

並進
同変

② 特徴抽出

畳み込み層

カラー画像のように「複数チャンネル」の信号が入力されたときには、フィルタは各位置の各チャンネルごとに係数を持つ。



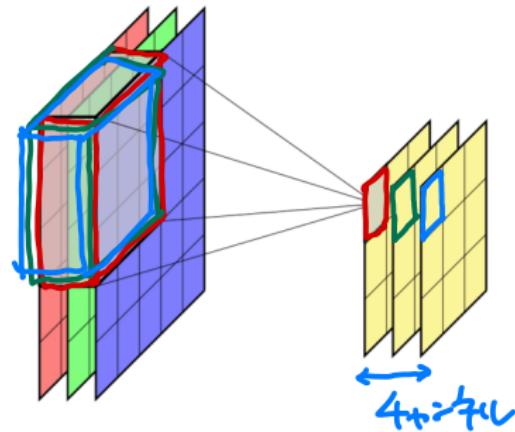
畳み込み層

フィルタを複数用意して、同一箇所に複数のフィルタを適用する。

フィルタの数だけ「潜在空間表現」が得られる。

異なるフィルタによる畳み込みにより、同一の入力から異なる出力を得る。

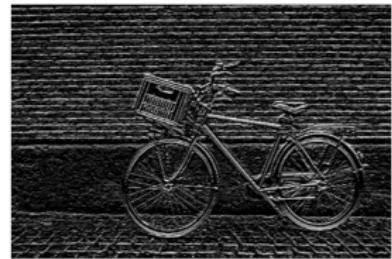
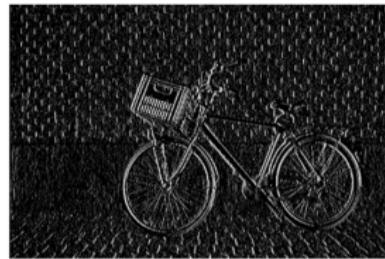
各出力を得る経路を「チャンネル」と呼ぶ。



同位置箇所から **フィルタの数だけの「特徴」**

畳み込み層

具体例



右ふたつは下記ふたつのフィルタそれぞれの出力結果

フィルタの数がチャンネルの数

-1	0	1
-1	0	1
-1	0	1

-1	-1	-1
0	0	0
1	1	1

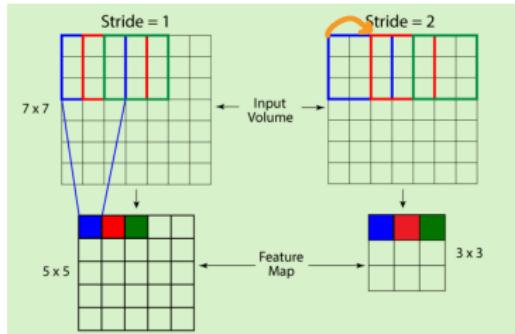
水平方向 垂直方向
ついで ついで

畳み込み層

- パディング：画像の縁からフィルタがはみ出すことへの対処。多くの場合、周囲を等幅 P 画素を「ゼロ」で埋める（ゼロパディング）。

0	0	0	0	0	0
0	X_{11}	X_{12}	X_{13}	X_{14}	0
0	X_{21}	X_{22}	X_{23}	X_{24}	0
0	X_{31}	X_{32}	X_{33}	X_{34}	0
0	X_{41}	X_{42}	X_{43}	X_{44}	0
0	0	0	0	0	0

- ストライド：フィルタをずらす量。縦横等しい間隔でずらす場合が多い。ずらす量を S で表す。フィルタの移動にともなう内積結果の変化が緩やかなときに S を大きくすると必要なメモリの量が小さくなる。



畠み込み層

畠み込み層の出力結果のサイズ

- 入力 : $J \times K$
- フィルタのサイズ : $M \times M$
- パディングの幅 : P
- ストライドの幅 : S

出力のサイズは下記のとおり

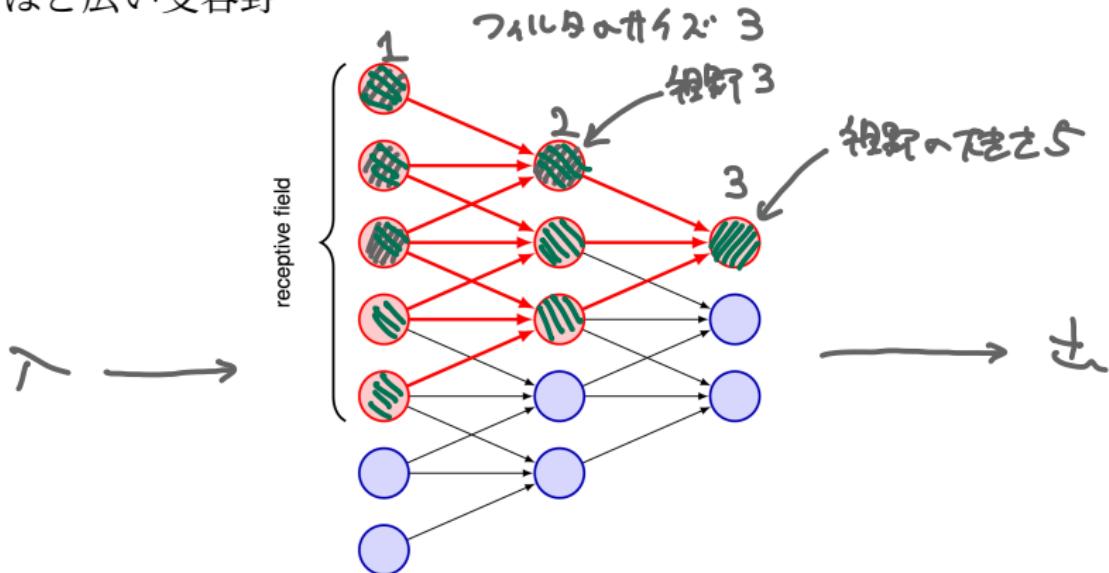
$$\left(\frac{J + 2P - M}{S} + 1 \right) \times \left(\frac{K + 2P - M}{S} + 1 \right)$$

概ね $1/S$ のサイズになる

右とで
ストライド → 大
メモリ → 小

畳み込み層

受容野：各ユニットに影響を及ぼす入力画像中の領域
深い層ほど広い受容野



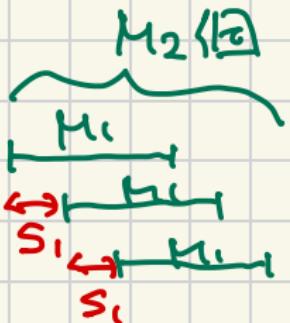
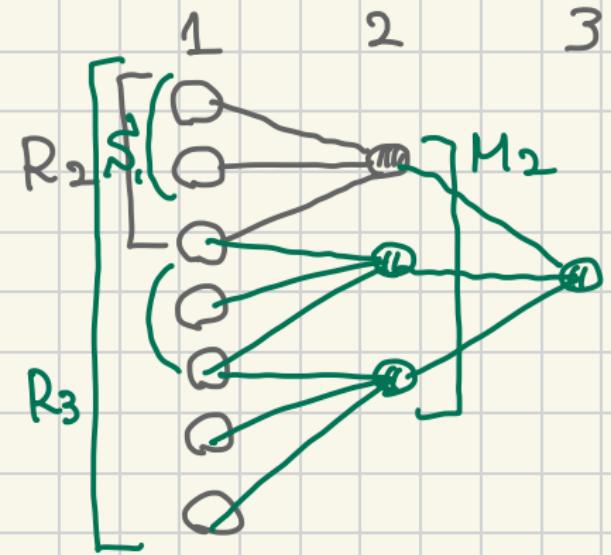
畳み込み層

第 $k = 2, 3, \dots$ 層目のひとつのユニットに対応する受容野の広さ
 $R_k \times R_k$

- 第 $j = 1, 2, \dots, k - 1$ 層目のフィルタのサイズ： $M_j \times M_j$
($S_0 = 1$ とする)
- 第 $j = 1, 2, \dots, k - 1$ 層目のストライドの大きさ： S_j

$$R_k = 1 + \sum_{j=1}^{k-1} \{(M_j - 1) \prod_{i=0}^{j-1} S_i\}$$

$$S_0 = 1$$



$$R_3 = (M_2 - 1) S_1 + M_1$$

$$\begin{aligned}
 R_2 &= 1 + \sum_{j=1}^l (M_1 - 1) \prod_{i=0}^{j-1} S_i \\
 &= 1 + M_1 - 1 = M_1
 \end{aligned}$$

2層目への影響率の
合計は M_1
(つづいて下記)

畳み込み層

並進同変性

：画像を並進させてからフィルタとの畳み込みを計算した結果と、画像とフィルタの畳み込みを計算したあとで並進した結果が一致する性質

- 画像の特徴表現にとって並進同変性は有用

画像データと CNN

CNN の畳み込み層は並進同変性を持つ

プーリング層

特徴空間をダウンサンプルする

入力の局所的な変位・変形に対して特徴を不变にする

畳み込み演算の結果を格子状の小領域に分割して、各領域の最大値を選択したり、平均値を計算したりする。最大値を選択することを max pooling と呼び、平均値を計算することを average pooling と呼ぶ。多くの CNN が max pooling を採用する

色の
集約

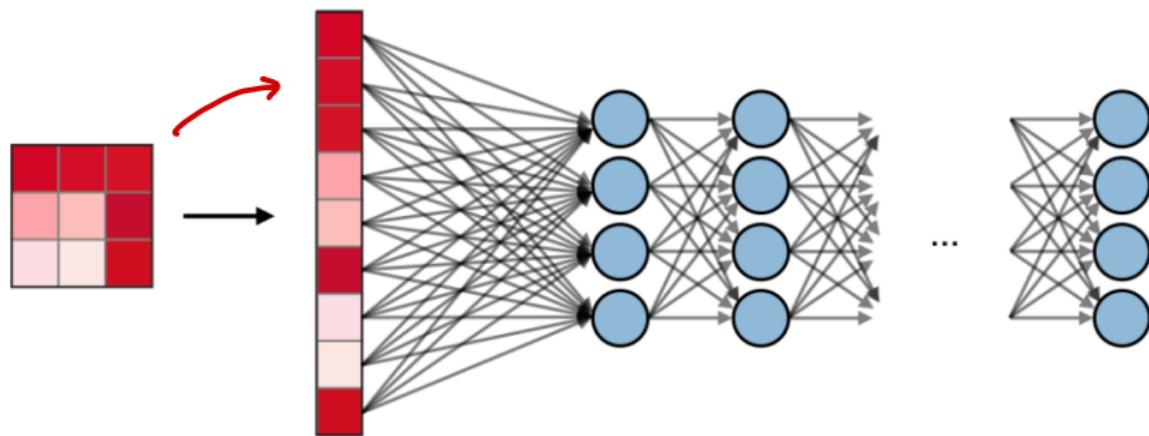
3	5	4	6
1	1	9	4
7	10	9	5
12	2	9	4



5	9
12	9

全結合層

畳み込みと異なり、直前の層の各ニューロンを全てのニューロンに結合する。実装上は直前の層を **Flatten** してから全結合する



モデルの複雑さ / パラメータの数

入力
(画像の一辺のサイズ)
4分割 (RGB 3)

モデルのパラメータ数は各層で次のように

	畳み込み	Pooling	全結合
入力	$I \times I \times C$	$I \times I \times C$	N_{in}
出力	$O \times O \times K$	$O \times O \times K$	N_{out}
パラメータ数	$(M \times M \times C) \cdot K$	0	$(N_{in} + 1) \times N_{out}$
メモ	$K = 2C$ が多い	チャンネルごと	+1はbias項

大きい

つまらない

大きい
パラメータ
($\sim 10^{10}$)

$M = 100 \times 2$

$C \times K \times 10$
 $CNN \leftarrow 10^4$

$N_{in} = 1000 \times 1000$
 10^6

$N_{out} = 10^6$
 $MLP \rightarrow 10^{12}$

画像分類用の CNN

画像分類用の代表的なアーキテクチャ

- AlexNet (2012)
- VGG (2014)
- GoogleNet (2014)
- ResNet (2015)

画像データベース

ImageNet Large Scale Visual Recognition Challenge(ILSVRC)

2012-2017: 1000 カテゴリ、120 万枚の訓練画像、5 万枚の評価用画像、10 万枚のテスト画像



14,197,122 images, 21841 synsets indexed

[Home](#) [Download](#) [Challenges](#) [About](#)

Not logged in. [Login](#) | [Signup](#)

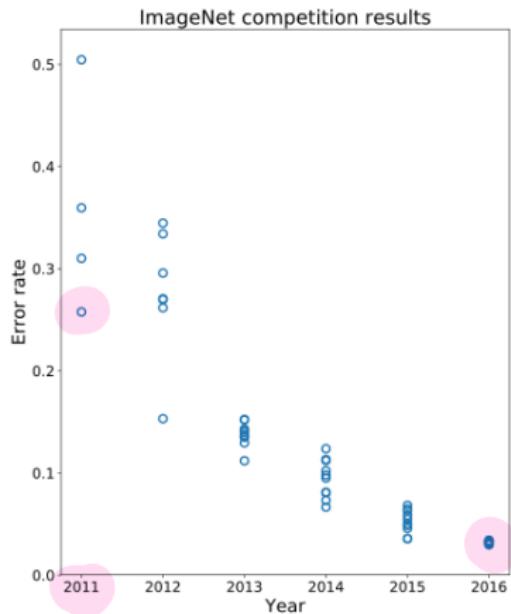
Download

Download ImageNet Data

The most highly-used subset of ImageNet is the [ImageNet Large Scale Visual Recognition Challenge \(ILSVRC\)](#) 2012-2017 image classification and localization dataset. This dataset spans 1000 object classes and contains 1,281,167 training images, 50,000 validation images and 100,000 test images. This subset is available on [Kaggle](#).

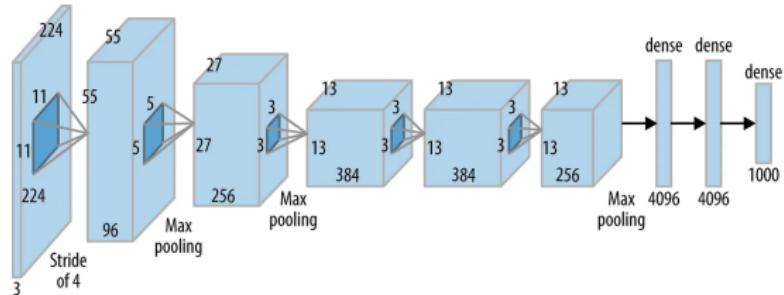
For access to the full ImageNet dataset and other commonly used subsets, please login or request access. In doing so, you will need to agree to our terms of access.

性能改善の歴史 (2011-2017)

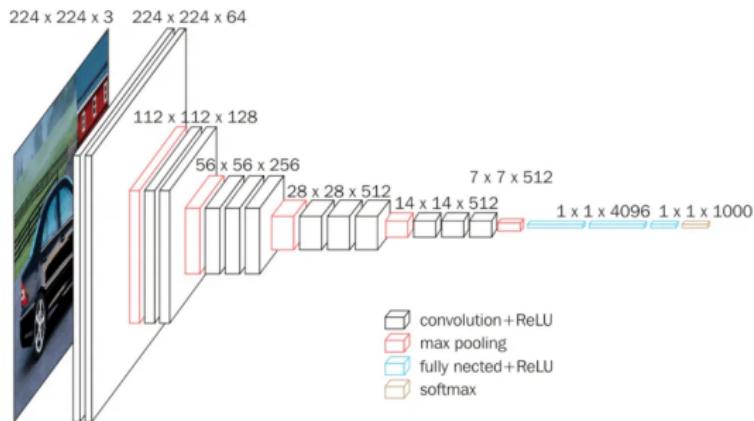


2017年に参加チームのほとんどが識別率95%を達成。
ILSVRCは役目を終えたと考えられて終了。

AlexNet and VGG16

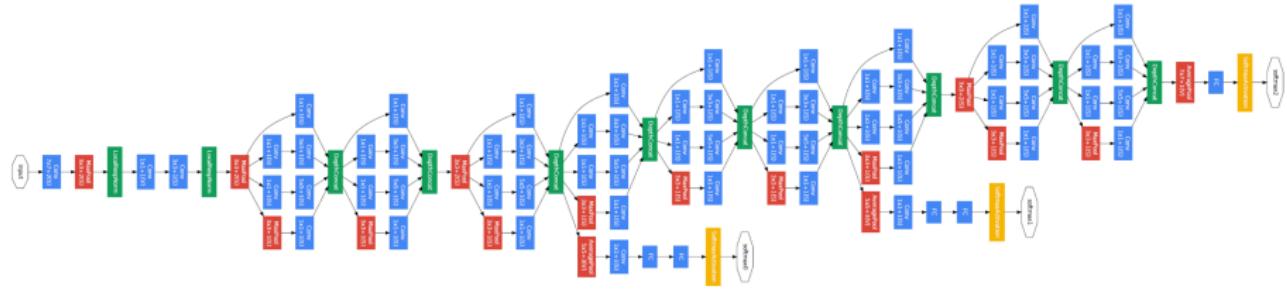


AlexNet

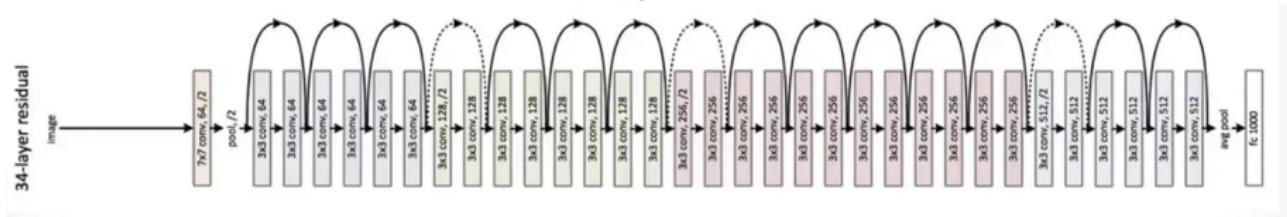


VGG16

GoogleNet and ResNet



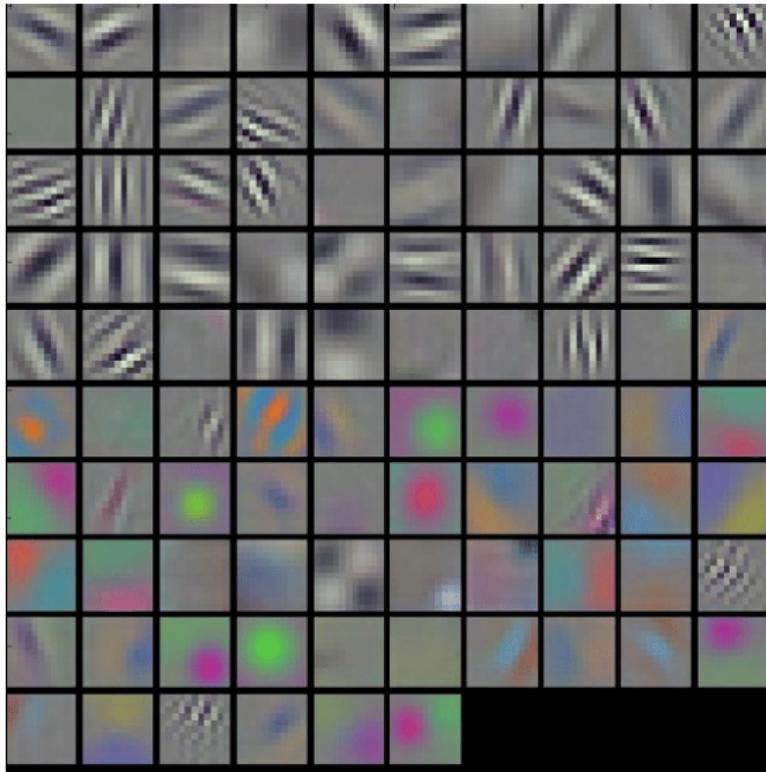
GoogleNet



ResNet34

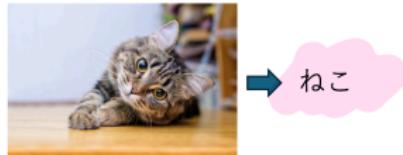
AlexNet のフィルタ

ImageNet で学習して得られた AlexNet の最初の層のフィルタ

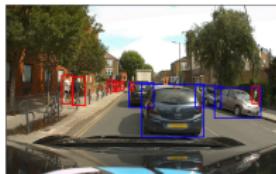


画像処理事例

- 画像認識：画像中の対象のクラスを推定する



- 物体検出：画像中の対象を検出する（矩形で囲む）



(矩形内部のクラス識別/矩形位置の回帰)

- 画像セグメンテーション：画像を領域へと分割する



(画素ごとに多クラス識別)