

Super VIP チートシート: ディープラーニング 1 置み込みニューラルネットワーク

アシンニアミディ・シェルビンアミディ著

チャントゥアンアン・中井喜之訳

6日10月2019年

目次

1 置み込みニューラルネットワーク

1.1	概要	1
1.2	層の種類	1
1.3	フィルタハイパーパラメータ	2
1.4	ハイパーパラメータの調整	2
1.5	よく使われる活性化関数	3
1.6	物体検出	3
1.6.1	顔認証及び認識	4
1.6.2	ニューラルスタイル変換	4
1.6.3	計算トリックを使うアーキテクチャ	4

2 リカレントニューラルネットワーク

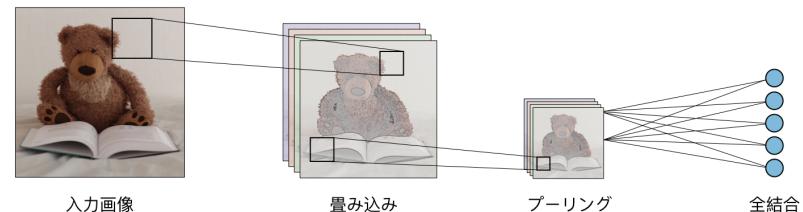
2.1	概要	5
2.2	長期依存関係の処理	5
2.3	単語表現の学習	6
2.3.1	動機と表記	7
2.3.2	単語の埋め込み	7
2.4	単語の比較	8
2.5	言語モデル	8
2.6	機械翻訳	8
2.7	アテンション	9

3 ディープラーニングのアドバイスやコツ

3.1	データ処理	9
3.2	ニューラルネットワークの学習	9
3.2.1	定義	9
3.2.2	最適な重みの探索	10
3.3	パラメータチューニング	10
3.3.1	重みの初期化	10
3.3.2	収束の最適化	10
3.4	正則化	10
3.5	おすすめの技法	11

1.1 概要

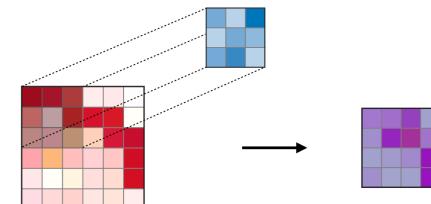
□ 伝統的な置み込みニューラルネットワークのアーキテクチャ – CNNとしても知られる置み込みニューラルネットワークは一般的に次の層で構成される特定種類のニューラルネットワークです。



置み込み層とプーリング層は次のセクションで説明されるハイパーパラメータに関してファインチューニングできます。

1.2 層の種類

□ 置み込み層(CONV) – 置み込み層(CONV)は入力 I を各次元に関して走査する時に、置み込み演算を行うフィルタを使用します。置み込み層のハイパーパラメータにはフィルタサイズ F とストライド S が含まれます。結果出力 O は特徴マップまたは活性化マップと呼ばれます。

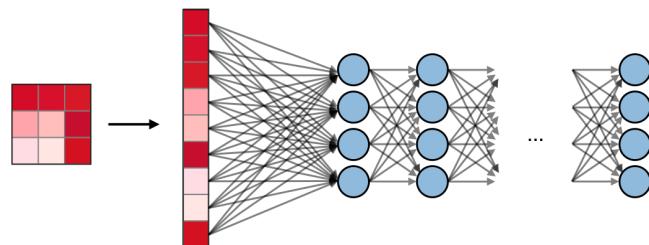


注: 置み込みステップは1次元や3次元の場合にも一般化できます。

□ プーリング(POOL) – プーリング層(POOL)は位置不变性をもつ縮小操作で、通常は置み込み層の後に適用されます。特に、最大及び平均プーリングはそれぞれ最大と平均値が取られる特別な種類のプーリングです。

種類	最大プーリング	平均プーリング
	各プーリング操作は現在のビューの中から最大値を選ぶ	各プーリング操作は現在のビューに含まれる値を平均する
図		
コメント	- 検出された特徴を保持する - 最も一般的に利用される	- 特徴マップをダウンサンプリングする - LeNetで利用される

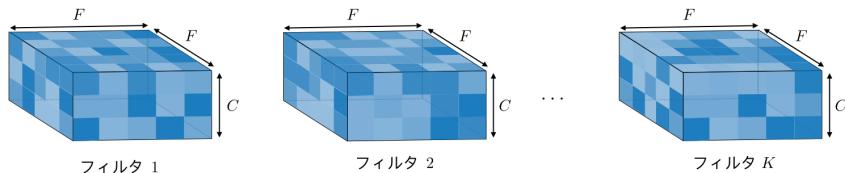
□ **全結合(FC)** – 全結合(FC) 層は平坦化された入力に対して演算を行います。各入力は全てのニューロンに接続されています。FC層が存在する場合、通常CNNアーキテクチャの末尾に向かって見られ、クラススコアなどの目的を最適化するため利用できます。



1.3 フィルタハイパラメータ

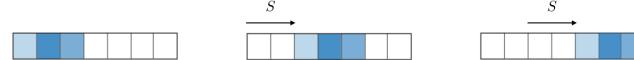
畳み込み層にはハイパラメータの背後にある意味を知ることが重要なフィルタが含まれています。

□ **フィルタの次元** – C 個のチャネルを含む入力に適用される $F \times F$ サイズのフィルタの体積は $F \times F \times C$ で、それは $I \times I \times C$ サイズの入力に対して畳み込みを実行して $O \times O \times 1$ サイズの特徴マップ(活性化マップとも呼ばれる)出力を生成します。



注: $F \times F$ サイズの K 個のフィルタを適用すると、 $O \times O \times K$ サイズの特徴マップの出力を得られます。

□ **ストライド** – 置み込みまたはプーリング操作において、ストライド S は各操作の後にウィンドウを移動させるピクセル数を表します。



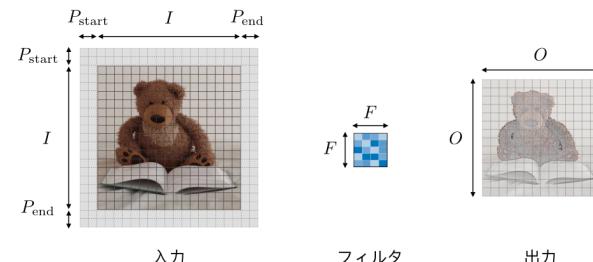
□ **ゼロパディング** – ゼロパディングとは入力の各境界に対して P 個のゼロを追加するプロセスを意味します。この値は手動で指定することも、以下に詳述する3つのモードのいずれかを使用して自動的に設定することもできます。

	Valid	Same	Full
値	$P = 0$	$P_{\text{start}} = \left\lceil \frac{S \lceil \frac{I}{S} \rceil - I + F - S}{2} \right\rceil$ $P_{\text{end}} = \left\lceil \frac{S \lceil \frac{I}{S} \rceil - I + F - S}{2} \right\rceil$	$P_{\text{start}} \in [0, F - 1]$ $P_{\text{end}} = F - 1$
図			
目的	- パディングなし - もし次元が合わなかつたら最後の置み込みをやめる	- 特徴マップのサイズが $\lceil \frac{I}{S} \rceil$ になるようなパディング - 出力サイズは数学的に扱いやすい - 「ハーフ」パディングとも呼ばれる	- 入力の一番端まで置み込みが適用されるような最大パディング - フィルタは入力を端から端まで「見る」

1.4 ハイパラメータの調整

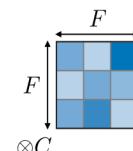
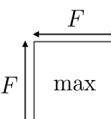
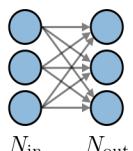
□ **畳み込み層内のパラメータ互換性** – I を入力ボリュームサイズの長さ、 F をフィルタの長さ、 P をゼロパディングの量、 S をストライドとすると、その次元に沿った特徴マップの出力サイズ O は次式で与えられます:

$$O = \frac{I - F + P_{\text{start}} + P_{\text{end}}}{S} + 1$$



注: 多くの場合 $P_{start} = P_{end} \triangleq P$ であり、上記の式の $P_{start} + P_{end}$ を $2P$ に置き換える事ができます。

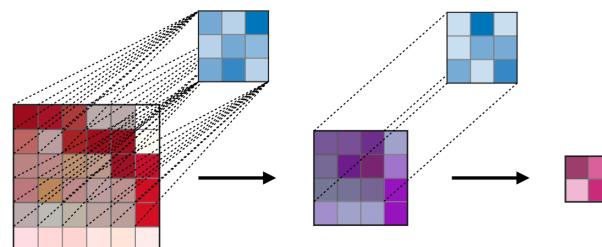
□ **モデルの複雑さを理解する** – モデルの複雑さを評価するために、モデルのアーキテクチャが持つパラメータの数を測定することがしばしば有用です。畳み込みニューラルネットワークの各層では、以下のように行なわれます。

	CONV	POOL	FC
図			
入力サイズ	$I \times I \times C$	$I \times I \times C$	N_{in}
出力サイズ	$O \times O \times K$	$O \times O \times C$	N_{out}
パラメータの数	$(F \times F \times C + 1) \cdot K$	0	$(N_{in} + 1) \times N_{out}$
備考	<ul style="list-style-type: none"> - フィルタごとに1つのバイアスパラメータ - ほとんどの場合、$S < F$ - Kの一般的な選択は$2C$ 	<ul style="list-style-type: none"> - プール操作はチャネルごとに行われる - ほとんどの場合、$S = F$ 	<ul style="list-style-type: none"> - 入力は平坦化される - ニューロンごとにひとつずつバイアスパラメータ - FCのニューロンの数には構造的制約がない

□ **受容野** – 層 k における受容野は、 k 番目の活性化マップの各ピクセルが「見る」ことができる入力の $R_k \times R_k$ の領域です。層 j のフィルタサイズを F_j 、層 i のストライド値を S_i とし、慣例に従って $S_0 = 1$ とすると、層 k での受容野は次の式で計算されます：

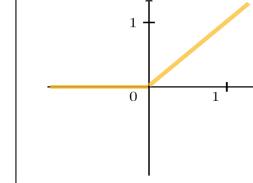
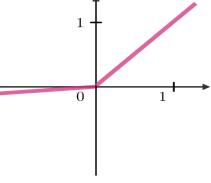
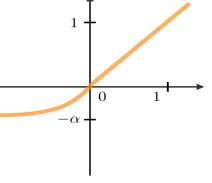
$$R_k = 1 + \sum_{j=1}^k (F_j - 1) \prod_{i=0}^{j-1} S_i$$

下記の例のように $F_1 = F_2 = 3$ 、 $S_1 = S_2 = 1$ とすると、 $R_2 = 1 + 2 \cdot 1 + 2 \cdot 1 = 5$ となります。.



1.5 よく使われる活性化関数

□ **正規化線形ユニット** – 正規化線形ユニット層(ReLU)はボリュームの全ての要素に利用される活性化関数 g です。ReLUの目的は非線型性をネットワークに導入することです。変種は以下の表でまとめられています：

ReLU	Leaky ReLU	ELU
$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ ただし $\epsilon \ll 1$	$g(z) = \max(\alpha(e^z - 1), z)$ ただし $\alpha \ll 1$
		
生物学的に解釈可能な非線形複雑性	負の値に対して ReLUが死んでいる問題に対処する	どこでも微分可能

□ **ソフトマックス** – ソフトマックスのステップは入力としてスコア $x \in \mathbb{R}^n$ のベクトルを取り、アーキテクチャの最後にあるソフトマックス関数を通じて確率 $p \in \mathbb{R}^n$ のベクトルを出力する一般化されたロジスティック関数として見ることができます。次のように定義されます。

$$p = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} \quad \text{ここで} \quad p_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

1.6 物体検出

□ **モデルの種類** – 物体認識アルゴリズムは主に3つの種類があり、予測されるものの性質は異なります。次の表で説明されています。

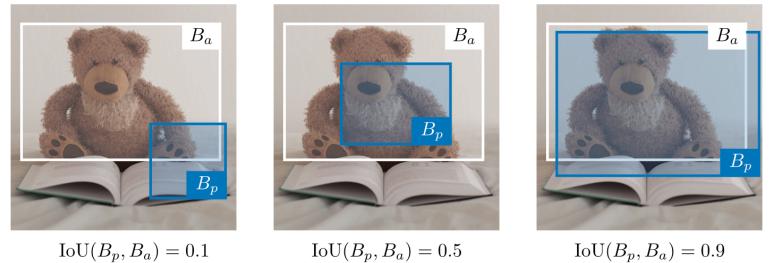
画像分類	位置特定を伴う分類	検出
テディベア	テディベア	テディベア 本
- 画像を分類する - 物体の確率を予測する	- 画像内の物体を検出する - 物体の確率とその位置を予測する	- 画像内の複数の物体を検出する - 複数の物体の確率と位置を予測する
伝統的なCNN	単純されたYOLO, R-CNN	YOLO, R-CNN

□ **検出** – 物体検出の文脈では、画像内の物体の位置を特定したいだけなのかあるいは複雑な形状を検出したいのかによって、異なる方法が使用されます。二つの主なものは次の表でまとめられています：

バウンディングボックス検出	ランドマーク検出
物体が配置されている画像の部分を検出する	- 物体（たとえば目）の形状または特徴を検出する - よりきめ細かい
中心(b_x, b_y)、高さ b_h 、幅 b_w のボックス	参照点(l_{1x}, l_{1y}), ..., (l_{nx}, l_{ny})

□ **Intersection over Union** – Intersection over Union (IoUとしても知られる)は予測された境界ボックス B_p が実際の境界ボックス B_a に対してどれだけ正しく配置されているかを量量化する関数です。次のように定義されます：

$$\text{IoU}(B_p, B_a) = \frac{B_p \cap B_a}{B_p \cup B_a}$$

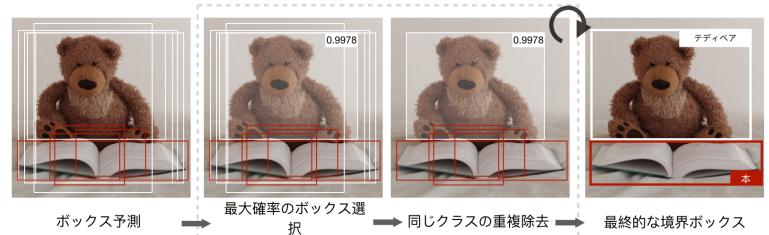


注：常に $\text{IoU} \in [0,1]$ となります。慣例では、 $\text{IoU}(B_p, B_a) \geq 0.5$ の場合、予測された境界ボックス B_p はそこそこ良いと見なされます。

□ **アンカーボックス** – アンカーボクシングは重なり合う境界ボックスを予測するために使用される手法です。実際には、ネットワークは同時に複数のボックスを予測することを許可されており、各ボックスの予測は特定の幾何学的属性の組み合わせを持つように制約されます。例えば、最初の予測は特定の形式の長方形のボックスになる可能性があり、2番目の予測は異なる幾何学的形式の別の長方形のボックスになります。

□ **非極大抑制** – 非極大抑制技術のねらいは、最も代表的なものを選択することによって、同じ物体の重複した重なり合う境界ボックスを除去することです。0.6未満の予測確率を持つボックスを全て除去した後、残りのボックスがある間、以下の手順が繰り返されます：

- ステップ1: 最大の予測確率を持つボックスを選ぶ。
- ステップ2: そのボックスに対して $\text{IoU} \geq 0.5$ となる全てのボックスを破棄する。



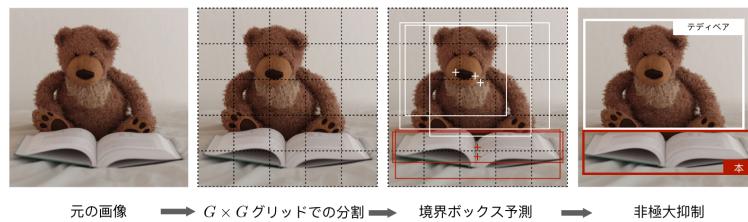
□ **YOLO** – You Only Look Once (YOLO)は次の手順を実行する物体検出アルゴリズムです。

- ステップ1: 入力画像を $G \times G$ グリッドに分割する。
- ステップ2: 各グリッドセルに対して次の形式の y を予測するCNNを実行する:

$$y = \left[\underbrace{p_c, b_x, b_y, b_h, b_w, c_1, c_2, \dots, c_p, \dots}_{k\text{回繰り返す}} \right]^T \in \mathbb{R}^{G \times G \times k \times (5+p)}$$

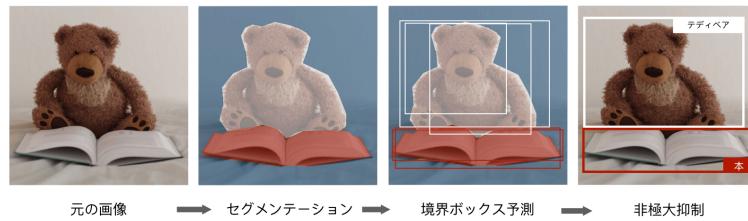
ここで、 p_c は物体を検出する確率、 b_x, b_y, b_h, b_w は検出された境界ボックスの属性、 c_1, \dots, c_p は p 個のクラスのうちどれかが検出されたかのOne-hot表現、 k はアンカーボックスの数です。

- ステップ3: 重複する可能性のある重なり合う境界ボックスを全て除去するため、非極大抑制アルゴリズムを実行する。



注: $p_c = 0$ のとき、ネットワークは物体を検出しません。その場合には、対応する予測 b_x, \dots, c_p は無視する必要があります。

□ **R-CNN** – Region with Convolutional Neural Networks (R-CNN) は物体検出アルゴリズムで、最初に画像をセグメント化して潜在的に関連する境界ボックスを見つけ、次に検出アルゴリズムを実行してそれらの境界ボックス内で最も可能性の高い物体を見つけます。



注: 元のアルゴリズムは計算コストが高くて遅いですが、Fast R-CNNやFaster R-CNNなどの、より新しいアーキテクチャではアルゴリズムをより早く実行できます。

1.6.1 顔認証及び認識

□ **モデルの種類** – 2種類の主要なモデルが次の表にまとめられています:

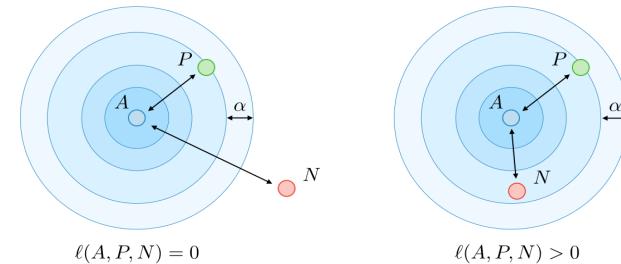
顔認証	顔認識
- これは正しい人ですか? - 1対1検索	- これはデータベース内のK人のうちの1人ですか - 1対多検索

□ **ワンショット学習** – ワンショット学習は限られた学習セットを利用して、2つの与えられた画像の違いを量化解する類似度関数を学習する顔認証アルゴリズムです。2つの画像に適用される類似度関数はしばしば $d(\text{画像1}, \text{画像2})$ と記されます。

□ **シャムネットワーク** – シャムネットワークは画像のエンコード方法を学習して2つの画像の違いを量化解することを目的としています。与えられた入力画像 $x^{(i)}$ に対してエンコードされた出力はしばしば $f(x^{(i)})$ と記されます。

□ **トリプレット損失** – トリプレット損失 ℓ は3つ組の画像 A (アンカー)、 P (ポジティブ)、 N (ネガティブ)の埋め込み表現で計算される損失関数です。アンカーとポジティブ例は同じクラスに属し、ネガティブ例は別のクラスに属します。マージンパラメータを $\alpha \in \mathbb{R}^+$ と呼ぶことによってこの損失は次のように定義されます:

$$\ell(A, P, N) = \max(d(A, P) - d(A, N) + \alpha, 0)$$



1.6.2 ニューラルスタイル変換

□ **モチベーション** – ニューラルスタイル変換の目的は与えられたコンテンツ C とスタイル S に基づく画像 G を生成することです。



□ **活性化** – 層 l における活性化は $a^{[l]}$ と表記され、次元は $n_H \times n_w \times n_c$ です。

□ **コンテンツコスト関数** – $J_{\text{content}}(C, G)$ というコンテンツコスト関数は生成された画像 G と元のコンテンツ画像 C との違いを測定するため利用されます。以下のように定義されます:

$$J_{\text{content}}(C, G) = \frac{1}{2} \|a^{[l]}(C) - a^{[l]}(G)\|^2$$

□ **スタイル行列** – 与えられた層 l のスタイル行列 $G^{[l]}$ はグラム行列で、各要素 $G_{kk'}^{[l]}$ がチャネル k と k' の相関関係を量化解します。活性化 $a^{[l]}$ に関して次のように定義されます。

$$G_{kk'}^{[l]} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_w^{[l]}} a_{ijk}^{[l]} a_{ijk'}^{[l]}$$

注: スタイル画像及び生成された画像に対するスタイル行列はそれぞれ $G^{[l](S)}$ 、 $G^{[l](G)}$ と表記されます。

□ **スタイルコスト関数** – スタイルコスト関数 $J_{\text{style}}(S, G)$ は生成された画像 G とスタイル S との違いを測定するため利用されます。以下のように定義されます:

$$J_{\text{style}}^{[l]}(S, G) = \frac{1}{(2n_H n_w n_c)^2} \|G^{[l](S)} - G^{[l](G)}\|_F^2 = \frac{1}{(2n_H n_w n_c)^2} \sum_{k,k'=1}^{n_c} \left(G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)} \right)^2$$

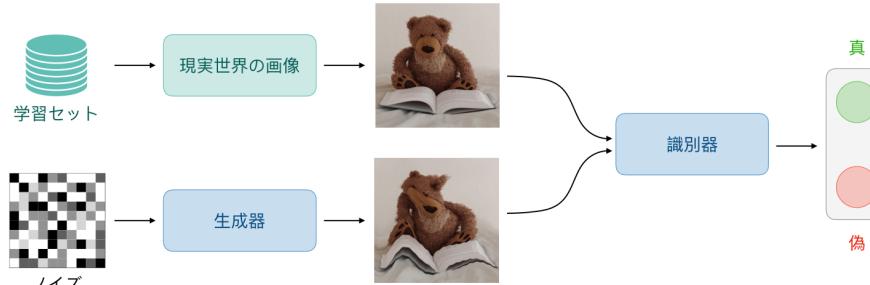
□ **全体のコスト関数** – 全体のコスト関数は以下のようにパラメータ α, β によって重み付けされたコンテンツ及びスタイルコスト関数の組み合わせとして定義されます:

$$J(G) = \alpha J_{\text{content}}(C, G) + \beta J_{\text{style}}(S, G)$$

注: α の値を大きくするとモデルはコンテンツを重視し、 β の値を大きくするとスタイルを重視します。

1.6.3 計算トリックを使うアーキテクチャ

□ **敵対的生成ネットワーク** – 敵対的生成ネットワーク (GANsとも呼ばれる) は生成モデルと識別モデルで構成されます。生成モデルの目的は、生成された画像と本物の画像を区別することを目的とする識別モデルに与えられる、最も本物らしい出力を生成することです。



注: GANsの変種を使用するユースケースにはテキストからの画像生成、音楽生成及び合成があります。

□ **ResNet** – Residual Networkアーキテクチャ (ResNetとも呼ばれる) は学習エラーを減らすため多数の層がある残差ブロックを使用します。残差ブロックは次の特性方程式を有します。

$$a^{[l+2]} = g(a^{[l]} + z^{[l+2]})$$

□ **インセプションネットワーク** – このアーキテクチャはインセプションモジュールを利用し、特徴量の多様化を通じてパフォーマンスを向上させるため、様々な畳み込みを試すことを目的としています。特に、計算負荷を限定するため 1×1 畳み込みトリックを使っています。

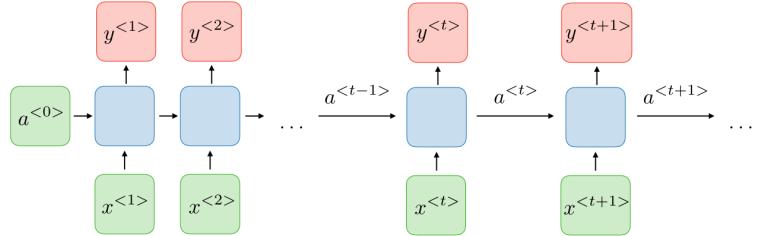
* * *

2 リカレントニューラルネットワーク

浜野秀明・中井喜之訳

2.1 概要

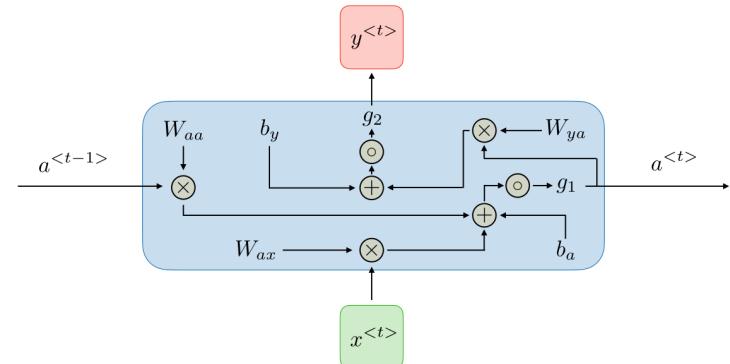
□ **一般的なRNNのアーキテクチャ** – RNNとして知られるリカレントニューラルネットワークは、隠れ層の状態を利用して、前の出力を次の入力として取り扱うことを可能にするニューラルネットワークの一形態です。一般的なモデルは下記のようになります。



それぞれの時点 t において活性化関数の状態 $a^{<t>}$ と出力 $y^{<t>}$ は下記のように表現されます。

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad \text{そして} \quad y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

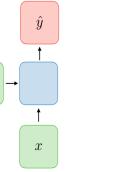
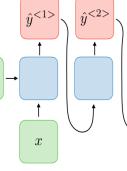
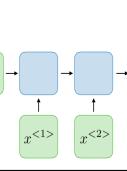
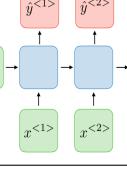
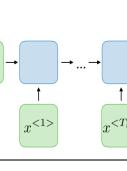
$W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ は全ての時点で共有される係数であり、 g_1, g_2 は活性化関数です。



一般的なRNNのアーキテクチャ利用の長所・短所については下記の表にまとめられています。

長所	短所
<ul style="list-style-type: none"> - 任意の長さの入力を処理できる - 入力サイズに応じてモデルサイズが大きくならない - 計算は時系列情報を考慮している - 重みは全ての時点で共有される 	<ul style="list-style-type: none"> - 遅い計算 - 長い時間軸での情報の利用が困難 - 現在の状態から将来の入力を予測不可能

□ **RNNの応用** – RNNモデルは主に自然言語処理と音声認識の分野で使用されます。以下の表に、さまざまな応用例がまとめられています。

RNNの種類	図	例
一対一 $T_x = T_y = 1$		伝統的なニューラルネットワーク
一対多 $T_x = 1, T_y > 1$		音楽生成
多対一 $T_x > 1, T_y = 1$		感情分類
多対多 $T_x = T_y$		固有表現認識
多対多 $T_x \neq T_y$		機械翻訳

□ **損失関数** – リカレントニューラルネットワークの場合、時間軸全体での損失関数 \mathcal{L} は、各時点での損失に基づき、次のように定義されます。

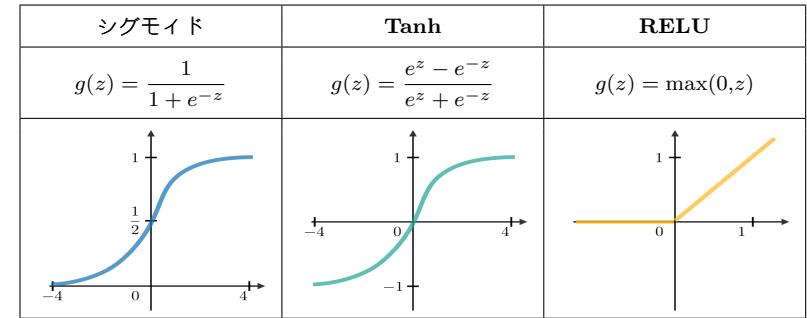
$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}(\hat{y}^{<t>}, y^{<t>})$$

□ **時間軸での誤差逆伝播法** – 誤差逆伝播(バックプロパゲーション)が各時点で行われます。時

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^T \left. \frac{\partial \mathcal{L}^{(T)}}{\partial W} \right|_{(t)}$$

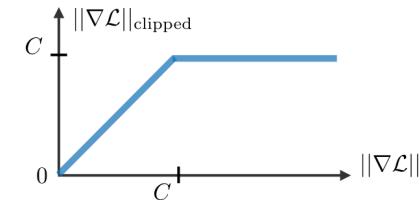
2.2 長期依存関係の処理

□ **一般的に使用される活性化関数** – RNNモジュールで使用される最も一般的な活性化関数を以下に説明します。



□ **勾配消失と勾配爆発について** – 勾配消失と勾配爆発の現象は、RNNでよく見られます。これらの現象が起こる理由は、掛け算の勾配が層の数に対して指数関数的に減少/増加する可能性があるため、長期の依存関係を捉えるのが難しいからです。

□ **勾配クリッピング** – 誤差逆伝播法を実行するときに時折発生する勾配爆発問題に対処するためには、勾配の上限値を定義することで、実際にこの現象が抑制されます。



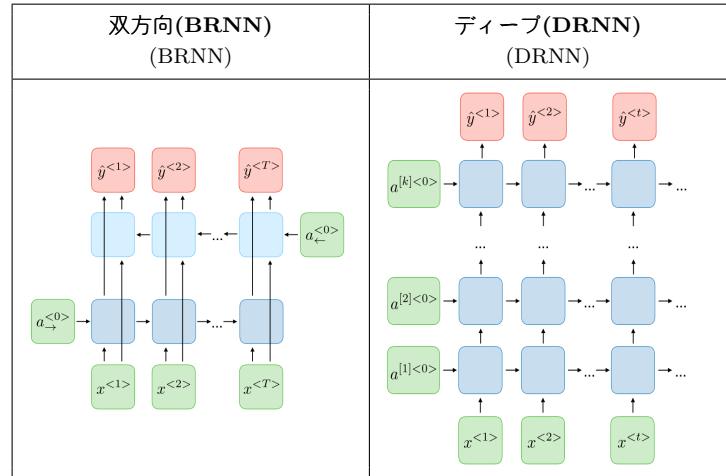
□ **ゲートの種類** – 勾配消失問題を解決するために、特定のゲートがいくつかのRNNで使用され、通常明確に定義された目的を持っています。それらは通常 Γ と記され、以下のように定義されます。

$$\Gamma = \sigma(Wx^{<t>} + Ua^{<t-1>} + b)$$

ここで、 W, U, b はゲート固有の係数、 σ はシグモイド関数です。主なものは以下の表にまとめられています。

ゲートの種類	役割	下記で使用される
更新ゲート Γ_u	過去情報はどのくらい重要ですか？	GRU, LSTM
関連ゲート Γ_r	前の情報を削除しますか？	GRU, LSTM
忘却ゲート Γ_f	セルを消去しますか？しませんか？	LSTM
出力ゲート Γ_o	セルをどのくらい見せますか？	LSTM

□ GRU/LSTM – ゲート付きリカレントユニット (GRU) およびロングショートタームメモリユニット (LSTM) は、従来のRNNが直面した勾配消失問題を解決しようとします。LSTMはGRUを一般化したものです。以下は、各アーキテクチャを特徴づける式をまとめた表です。



2.3 単語表現の学習

この節では、 V は語彙、そして $|V|$ は語彙のサイズを表します。

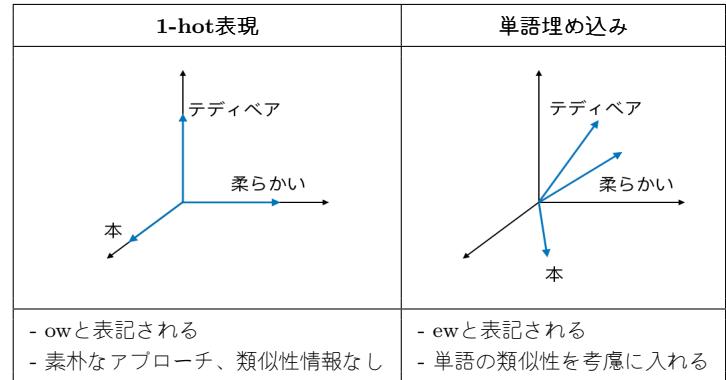
2.3.1 動機と表記

□ 表現のテクニック – 単語を表現する2つの主な方法は、以下の表にまとめられています。

	ゲート付きリカレントユニット (GRU)	ロングショートタームメモリ (LSTM)
$\tilde{c}^{<t>}$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$
$c^{<t>}$	$\Gamma_u \star \tilde{c}^{<t>} + (1 - \Gamma_u) \star c^{<t-1>}$	$\Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>}$
$a^{<t>}$	$c^{<t>}$	$\Gamma_o \star c^{<t>}$
依存関係		

備考：記号 $*$ は2つのベクトル間の要素ごとの乗算を表します。

□ RNNの変種 – 以下の表は、一般的に使用されている他のRNNアーキテクチャをまとめたものです。



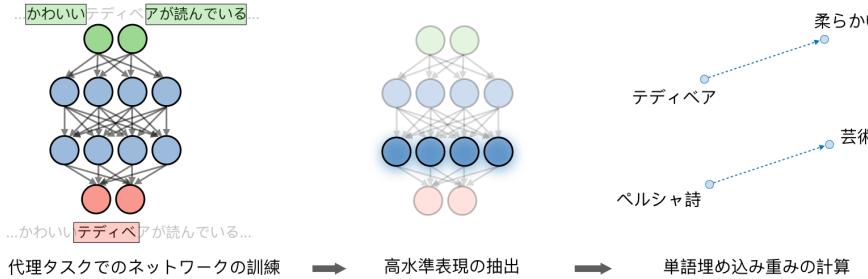
□ 埋め込み行列 – 与えられた単語 w に対して、埋め込み行列 E は、以下のように1-hot表現 o_w を埋め込み行列 e_w に写像します。

$$e_w = E o_w$$

注：埋め込み行列は、ターゲット/コンテキスト尤度モデルを使用して学習できます。

2.3.2 単語の埋め込み

□ **Word2vec** – Word2vecは、ある単語が他の単語の周辺にある可能性を推定することで、単語の埋め込みの重みを学習することを目的としたフレームワークです。人気のあるモデルは、スキップグラム、ネガティブサンプリング、およびCBOWです。



□ **スキップグラム** – スキップグラムword2vecモデルは、あるターゲット単語tがコンテキスト単語cと一緒に出現する確率を評価することで単語の埋め込みを学習する教師付き学習タスクです。tに関するパラメータを θ_t と表記すると、その確率 $P(t|c)$ は下記の式で与えられます。

$$P(t|c) = \frac{\exp(\theta_t^T e_c)}{\sum_{j=1}^{|V|} \exp(\theta_j^T e_c)}$$

注：softmax部分の分母の語彙全体を合計するため、このモデルの計算コストは高くなります。CBOWは、ある単語を予測するため周辺単語を使用する別のタイプのword2vecモデルです。

□ **ネガティブサンプリング** – ロジスティック回帰を使用したバイナリ分類器のセットで、特定の文脈とあるターゲット単語が同時に出現する確率を評価することを目的としています。モデルはk個のネガティブな例と1つのポジティブな例のセットで訓練されます。コンテキスト単語cとターゲット単語tが与えられると、予測は次のように表現されます。

$$P(y=1|c,t) = \sigma(\theta_t^T e_c)$$

注：この方法の計算コストは、スキップグラムモデルよりも少ないです。

□ **GloVe** – GloVeモデルは、単語表現のためのグローバルベクトルの略で、共起行列Xを使用する単語の埋め込み手法です。ここで、各 $X_{i,j}$ は、ターゲットiがコンテキストjで発生した回数を表します。そのコスト関数Jは以下の通りです。

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{|V|} f(X_{i,j})(\theta_i^T e_j + b_i + b'_j - \log(X_{i,j}))^2$$

ここで、fは $X_{i,j} = 0 \implies f(X_{i,j}) = 0$ となるような重み関数です。このモデルでeとθが果たす対称性を考えると、最後の単語の埋め込み $e_w^{(\text{final})}$ は下記のようになります。

$$e_w^{(\text{final})} = \frac{e_w + \theta_w}{2}$$

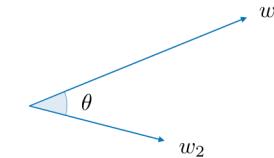
注：学習された単語の埋め込みの個々の要素は、必ずしも解釈可能ではありません。

2.4 単語の比較

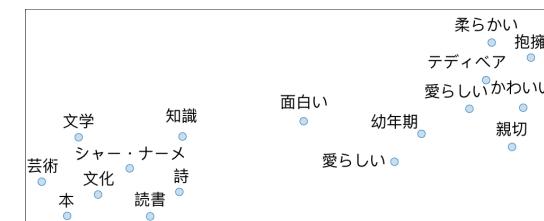
□ **コサイン類似度** – 単語 w_1 と w_2 のコサイン類似度は次のように表されます。

$$\text{similarity} = \frac{w_1 \cdot w_2}{\|w_1\| \|w_2\|} = \cos(\theta)$$

注： θ は単語 w_1 と w_2 の間の角度です。



□ **t-SNE** – t-SNE（t一分布型確率的近傍埋め込み）は、高次元埋め込みから低次元埋め込み空間への次元削減を目的とした手法です。実際には、2次元空間で単語ベクトルを視覚化するために使用されます。



2.5 言語モデル

□ **概要** – 言語モデルは文の確率 $P(y)$ を推定することを目的としています。

□ **n-gramモデル** – このモデルは、トレーニングデータでの出現数を数えることによって、ある表現がコーパスに出現する確率を量量化することを目的とした単純なアプローチです。

□ **バープレキシティ** – 言語モデルは一般的に、PPとも呼ばれるバープレキシティメトリックを使用して評価されます。これは、単語数Tにより正規化されたデータセットの逆確率と解釈できます。バープレキシティは低いほど良く、次のように定義されます。

$$\text{PP} = \prod_{t=1}^T \left(\frac{1}{\sum_{j=1}^{|V|} y_j^{(t)} \cdot \hat{y}_j^{(t)}} \right)^{\frac{1}{T}}$$

注：PPはt-SNEで一般的に使用されています。

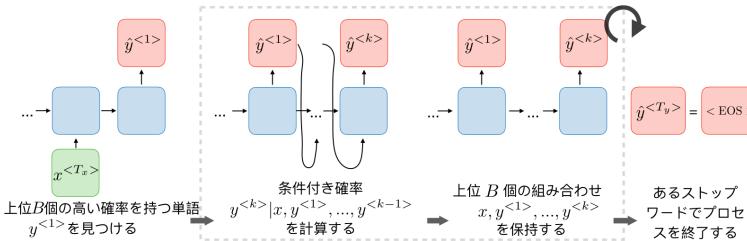
2.6 機械翻訳

□ **概要** – 機械翻訳モデルは、エンコーダーネットワークのロジックが最初に付加されている以外は、言語モデルと似ています。このため、条件付き言語モデルと呼ばれることもあります。目的は次のような文yを見つけることです。

$$y = \arg \max_{y^{<1>} \dots, y^{<T_y>}} P(y^{<1>} \dots, y^{<T_y>} | x)$$

□ ビーム検索 – 入力 x が与えられたとき最も可能性の高い文 y を見つけるために、機械翻訳と音声認識で使用されるヒューリスティック探索アルゴリズムです。

- ステップ1：上位 B 個の高い確率を持つ単語 $y^{<1>}$ を見つける。
- ステップ2：条件付き確率 $y^{<k>}|x, y^{<1>}, \dots, y^{<k-1>}$ を計算する。
- ステップ3：上位 B 個の組み合わせ $x, y^{<1>}, \dots, y^{<k>}$ を保持する。



注意：ビーム幅が1に設定されている場合、これは単純な貪欲法と同等です。

□ ビーム幅 – ビーム幅 B はビーム検索のパラメータです。 B の値を大きくするとより良い結果が得られますが、探索パフォーマンスは低下し、メモリ使用量が増加します。 B の値が小さいと結果が悪くなりますが、計算量は少くなります。 B の標準値は10前後です。

□ 文章の長さの正規化 – 数値の安定性を向上させるために、ビーム検索は通常次のように正規化（対数尤度正規化）された目的関数に対して適用されます。

$$\text{Objective} = \frac{1}{T_y^\alpha} \sum_{t=1}^{T_y} \log [p(y^{<t>}|x, y^{<1>}, \dots, y^{<t-1>})]$$

注：パラメータ α は緩衝パラメータと見なされ、その値は通常0.5から1の間です。

□ エラー分析 – 予測された \hat{y} の翻訳が良くない場合、以下のようなエラー分析を実行することで、なぜ y^* のような良い翻訳を得られなかったのか考えることが可能です。

症例	$P(y^* x) > P(\hat{y} x)$	$P(y^* x) \leq P(\hat{y} x)$
根本原因	ビーム検索の誤り	RNNの誤り
改善策	ビーム幅の拡大	-さまざまなアーキテクチャを試す -正則化 -データをさらに取得

□ Bleuスコア – Bleu (Bilingual evaluation understudy) スコアは、 n -gramの精度に基づき類似性スコアを計算することで、機械翻訳がどれほど優れているかを定量化します。以下のように定義されています。

$$\text{bleu score} = \exp \left(\frac{1}{n} \sum_{k=1}^n p_k \right)$$

ここで、 p_n は n -gramでのbleuスコアで下記のようにだけ定義されています。

$$p_n = \frac{\sum_{\substack{n-\text{gram} \in \hat{y}}} \text{count}_{\text{clip}}(\text{n-gram})}{\sum_{\substack{n-\text{gram} \in \hat{y}}} \text{count}(\text{n-gram})}$$

注：人為的に水増しされたブルースコアを防ぐために、短い翻訳評価には簡潔さへのペナルティが適用される場合があります。

2.7 アテンション

□ アテンションモデル – このモデルを使用するとRNNは重要であると考えられる入力の特定部分に注目することができ、得られるモデルの性能が実際に向上します。時刻 t において、出力 $y^{<t>}$ が活性化関数 $a^{<t>}$ とコンテキスト $c^{<t>}$ とに払うべき注意量を $\alpha^{<t,t'>}$ と表記すると次のようになります。

$$c^{<t>} = \sum_{t'} \alpha^{<t,t'>} a^{<t'>} \quad \text{および} \quad \sum_{t'} \alpha^{<t,t'>} = 1$$

注：アテンションスコアは、一般的に画像のキヤブション作成および機械翻訳で使用されています。



かわいいティベアがベルシャ文学を読んでいます



かわいいティベアがベルシャ文学を読んでいます

□ アテンションの重み – 出力 $y^{<t>}$ が活性化関数 $a^{<t>}$ に払うべき注意量 $\alpha^{<t,t'>}$ は次のように計算されます。

$$\alpha^{<t,t'>} = \frac{\exp(e^{<t,t'>})}{\sum_{t''=1}^{T_x} \exp(e^{<t,t''>})}$$

注：この計算の複雑さは T_x に関して2次です。

* * *

3 ディープラーニングのアドバイスやコツ

カムエララウ・中井喜之・森浩貴訳

3.1 データ処理

□ **Data augmentation (データ拡張)** – 大抵の場合は、深層学習のモデルを適切に訓練するには大量のデータが必要です。Data augmentation という技術を用いて既存のデータから、データを増やすことがよく役立ちます。以下、Data augmentation の主な手法はまとまっています。より正確には、以下の入力画像に対して、下記の技術を適用できます。

元の画像	反転	回転	ランダムな切り抜き
- 何も変更されていない画像	- 画像の意味が変わらない軸における反転	- わずかな角度の回転 - 不正確な水平線の校正 (calibration) をシミュレートする	- 画像の一部へのランダムなフォーカス - 連続して数回のランダムな切り抜きが可能

カラーシフト	ノイズの付加	情報損失	コントラストの修正
- RGBのわずかな修正 - 照らされ方によるノイズを捉える	- ノイズの付加 - 入力画像の品質のばらつきへの耐性の強化	- 画像の一部を無視 - 画像の一部が欠ける可能性を再現する	- 明るさの変化 - 時刻による露出の違いをコントロールする

□ **Batch normalization** – ハイバーパラメータ γ, β によってバッチ $\{x_i\}$ を正規化するステップです。修正を加えたいバッチの平均と分散を μ_B, σ_B^2 と表記すると、以下のように行えます。

$$x_i \leftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

より高い学習率を利用可能にし初期化への強い依存を減らすことを目的として、基本的には全結合層・畳み込み層のあとで非線形層の前に行います。

3.2 ニューラルネットワークの学習

3.2.1 定義

□ **エポック** – モデル学習においてエポックとは学習の繰り返しの中の1回を指す用語で、1エポックの間にモデルは全学習データからその重みを更新します。

□ **ミニバッチ勾配降下法** – 学習段階では、計算が複雑になりすぎるため通常は全データを一度に使って重みを更新することはありません。またノイズが問題になるため1つのデータポイントだけを使って重みを更新することもありません。代わりに、更新はミニバッチごとに行われます。各バッチに含まれるデータポイントの数は調整可能なハイバーパラメータです。

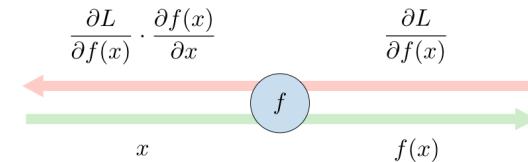
□ **損失関数** – 得られたモデルの性能を数値化するために、モデルの出力 z が実際の出力 y をどの程度正確に予測できているかを評価する損失関数 L が通常使われます。

□ **交差エントロピー誤差** – ニューラルネットワークにおける二項分類では、交差エントロピー誤差 $L(z, y)$ が一般的に使用されており、以下のように定義されています。

$$L(z, y) = - \left[y \log(z) + (1 - y) \log(1 - z) \right]$$

3.2.2 最適な重みの探索

□ **誤差逆伝播法** – 実際の出力と期待される出力の差に基づいてニューラルネットワークの重みを更新する手法です。各重み w に関する微分は連鎖律を用いて計算されます。

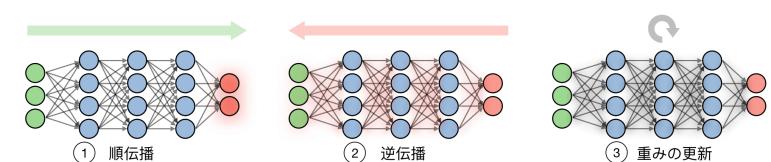


この方法を使用することで、それぞれの重みはそのルールにしたがって更新されます。

$$w \leftarrow w - \alpha \frac{\partial L(z, y)}{\partial w}$$

□ **重みの更新** – ニューラルネットワークでは、以下の方法にしたがって重みが更新されます。

- ステップ1：訓練データのバッチを用いて順伝播で損失を計算します。
- ステップ2：損失を逆伝播させて各重みに関する損失の勾配を求めます。
- ステップ3：求めた勾配を用いてネットワークの重みを更新します。



3.3 パラメータチューニング

3.3.1 重みの初期化

□ **Xavier初期化** – 完全にランダムな方法で重みを初期化するのではなく、そのアーキテクチャのユニークな特徴を考慮に入れて重みを初期化する方法です。

□ **転移学習** – 深層学習のモデルを学習させるには大量のデータと何よりも時間が必要です。膨大なデータセットから数日・数週間をかけて構築した学習済みモデルを利用し、自身のユースケースに活かすことは有益であることが多いです。手元にあるデータ量次第ではありますが、これを利用する以下の方があります。

学習サイズ	図	解説
小		全層を凍結し、softmaxの重みを学習させる
中		大半の層を凍結し、最終層とsoftmaxの重みを学習させる
大		学習済みの重みで初期化して各層とsoftmaxの重みを学習させる

3.3.2 収束の最適化

□ **学習率** – 多くの場合 α や時々 η と表記される学習率とは、重みの更新速度を表しています。学習率は固定することもできる上に、適応的に変更することもできます。現在もっとも使用される手法は、学習率を適切に調整するAdamと呼ばれる手法です。

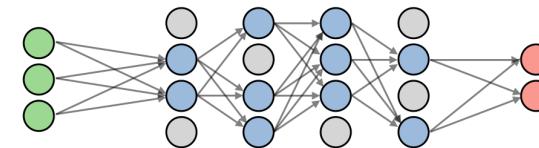
□ **適応学習率法** – モデルを学習させる際に学習率を変動させると、学習時間の短縮や精度の向上につながります。Adamがもっとも一般的に使用されている手法ですが、他の手法も役立つことがあります。それらの手法を下記の表にまとめました。

手法	解説	wの更新	bの更新
Momentum (運動量)	- 振動を抑制する - SGDの改良 - チューニングするパラメータは2つ	$w \leftarrow w - \alpha v_{dw}$	$b \leftarrow b - \alpha v_{db}$
RMSprop	- 二乗平均平方根のプロパゲーション - 振動をコントロールすることで学習アルゴリズムを高速化する	$w \leftarrow w - \alpha \frac{dw}{\sqrt{s_{dw}}}$	$b \leftarrow b - \alpha \frac{db}{\sqrt{s_{db}}}$
Adam	- Adaptive Moment estimation - もっとも人気のある手法 - チューニングするパラメータは4つ	$w \leftarrow w - \alpha \frac{v_{dw}}{\sqrt{s_{dw}} + \epsilon}$	$b \leftarrow b - \alpha \frac{v_{db}}{\sqrt{s_{db}} + \epsilon}$

備考：他に Adadelta, Adagrad, SGD などの手法があります。

3.4 正則化

□ **ドロップアウト** – ドロップアウトとは、ニューラルネットワークで過学習を避けるために $p > 0$ の確率でノードをドロップアウト（無効化）する手法です。モデルが特定の特徴量に依存しそぎることを避けるよう強制します。

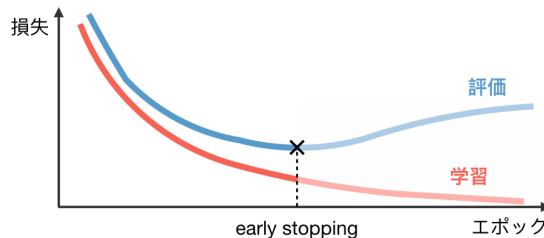


備考：ほとんどの深層学習のフレームワークでは、ドロップアウトを 'keep' というパラメータ ($1 - p$) でパラメータ化します。

□ **重みの正則化** – 重みが大きくなりすぎず、モデルが過学習しないようにするために、モデルの重みに対して正則化を行います。主な正則化手法は以下の表にまとめられています。

LASSO	Ridge	Elastic Net
- 係数を0へ小さくする - 変数選択に適している	係数を小さくする	変数選択と小さい係数のトレードオフ
$\dots + \lambda \theta _1$ $\lambda \in \mathbb{R}$	$\dots + \lambda \theta _2^2$ $\lambda \in \mathbb{R}$	$\dots + \lambda [(1 - \alpha) \theta _1 + \alpha \theta _2^2]$ $\lambda \in \mathbb{R}, \alpha \in [0, 1]$

- **Early stopping** – バリデーションの損失が変化しなくなるか、あるいは増加し始めたときに学習を早々に止める正則化方法



3.5 おすすめの技法

- **小さいバッチの過学習** – モデルをデバッグするとき、モデル自体の構造に大きな問題がないか確認するため簡易的なテストが役に立つことが多いです。特に、モデルを正しく学習できることを確認するため、ミニバッチをネットワークに渡してそれを過学習できるかを見ます。もしできなければ、モデルは複雑すぎるか単純すぎるかのいずれかであることを意味し、普通サイズの学習データセットはもちろん、小さいバッチですら過学習できないのです。

- **Gradient checking (勾配チェック)** – Gradient checking とは、ニューラルネットワークの逆伝播を実装する際に用いられる手法です。特定の点で解析的勾配と数値的勾配とを比較する手法で、逆伝播の実装が正しいことを確認できます。

	数値的勾配	解析的勾配
公式	$\frac{df}{dx}(x) \approx \frac{f(x+h) - f(x-h)}{2h}$	$\frac{df}{dx}(x) = f'(x)$
コメント	<ul style="list-style-type: none"> - 計算コストが高い、損失を次元ごとに2回計算する必要がある - 解析的実装が正しいかのチェックに用いられる - hを選ぶ時に小さすぎると数値不安定になり、大きすぎると勾配近似が不正確になるというトレードオフがある 	<ul style="list-style-type: none"> - 「正しい」結果 - 直接的な計算 - 最終的な実装で使われる

* * *