はじめに

近年は、計算機の発展に伴いディープラーニング（深層学習）の研究が盛んになってきました。特徴量を手動で設計して分類や回帰を行う従来の機械学習とは異なり、特徴量を「学習」させることが出来る点がディープラーニングの大きな強みです。数年前に登場した畳み込みニューラルネットワークにはじまり、様々なモデルが画像認識分野で目覚ましい精度を発揮するようになりました。また、敵対的生成ネットワークの登場により「絵を描く」などの創造的なタスクもこなせるようになりました。自然言語処理の分野では、ニューラルネットワークを用いて言語を翻訳するモデルも登場しました。音声処理の分野では、音楽を生成するようなモデルも存在します。このように、ディープラーニングはあらゆる場面で欠かせない存在となりつつあります。それに伴い、様々なデータやモデルを活用できる人材も求められています。

これらの分野に興味のある大学生は多いのですが、いざ本格的な本を開いてみると複雑な数式に心を折られて本棚に戻してしまう人が多いのではないでしょうか。そこで、同じ志をもつ学生が一緒に勉強できる場所を作ろうと思い立ち上げたのが京都大学人工知能研究会 KaiRA です。

KaiRA は理系文系・専攻・回生・大学を問わず様々なバックグラウンドの学生が一丸となって活動しています。KaiRA のメンバーは毎週一回一緒に集まり、専門書籍の輪読会を行っています。難解な書籍も、発表者のスライドや質疑応答を通じて理解を深めることができます。これに加え、勉強した知識を活かして各自アプリケーションの開発も行っています。NF では、これまでの勉強会の内容を簡単にまとめたポスターや、実際に作成したデモを展示しております。会長の私自身まだまだ勉強途中の未熟者ですが、この NF の展示で機械学習や深層学習の面白さが伝われば幸いです。

京都大学人工知能研究会
KaiRA 会長
金子　英樹

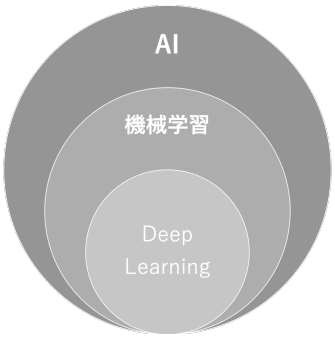
目次

第 1 章

Deep Learning 入門

1.1 Deep Learning の話の前に

AI という言葉をよく耳にしますね. 「知的な作業を人にかわって自動で行う」という意味合いで広く使われています.



人が細かく指示を出さなくとも機械がある問題の解決方法を独自に学習する「機械学習」はその一部ですので AI の意味するところが必ずしも「学習」と関係があるわけではないことに多少の注意が必要です.

機械学習システムは問題に関する**大量のデータ**を与えることで**訓練**し, その問題の解決方法を学習させます. 一度解決方法を学習した機械学習システムは初見のデータにも対応できるようになります.
例えば, 犬と猫の画像を自動的に分類してタグ付けする問題を考えてみます. この場合, 機械学習システムを訓練するときに与えるのは

1. 犬と猫についての大量のデータ
2. それぞれのデータが犬, 猫のどちらなのかを表す正解データ

の 2 つであり, 最終的に機械は犬と猫を分類する方法を学習し, 自動で分類できるようになります.

訓練している間, 機械学習システムは 2. で与えられた正解データ (期待される出力) と現段階での自分の出力との誤差を小さくするようにして入力 (犬や猫) を意味のある出力 (犬か猫かのタグ付け) に変換できるようにしています.
ただし, 入力画像を機械学習の各々の手法に適したデータに変換する作業 (特徴エンジニアリング) は人の手によって行われなければなりません.

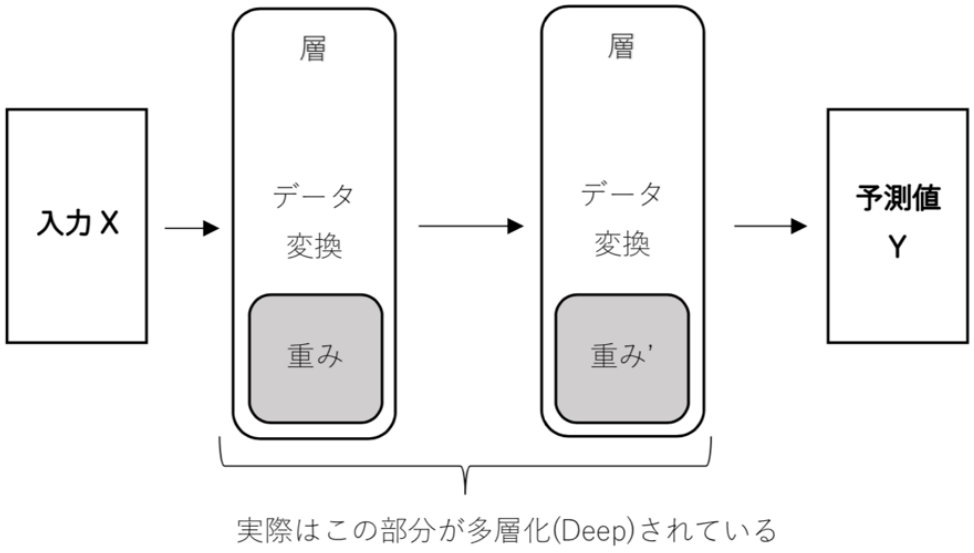
Deep Learning では特徴エンジニアリングが自動化されています。つまり, 画像分類の問題を学習させるのであれば画像データを入力として与えることで機械が画像の特徴を抽出して, それに基づいて分類するようになります.

Deep Learning で用いられているアルゴリズムは約 30 年前にできています. Deep Learning が最近になって注目されるようになった理由は, 高速な並列演算が可能な GPU の進歩や, インターネット上で大量にデータを入手できるようになったことです.

1.2　Deep Learning の概要

Deep Learning で用いられているニューラルネットワークというモデルはたくさんの「**層**」が積み重なってできています．それぞれの層では入力データを変換して情報抽出が行われており, 最終的に求められる出力がされるようになります．

Deep Learning の中身を知る前に, ここでは Deep Learning の学習の大まかな流れを見ていきたいと思います．



各層が, 入ってきたデータに対してどのような変換を施すのかはその層の持つ**重み** (パラメータ) によって決まります．この場合, 学習とは正しい予測を得られるような重みをすべての層にわたって見つけ出すことを意味します．

重みの値を調整する過程ではモデルの (学習の各段階での) 予測値と正解値とのずれ (誤差) を指標にします．そのずれを計算する関数を**損失関数** (目的関数とも呼ぶ) といい, 算出されたずれを損失率といいます．

次節以降で紹介する**誤差逆伝播法**という手法を用いて損失率をもとに重みの値を少しずつ調整します．損失率を計算してそれをもとに重みの値を調整することは学習において何度も繰り返されます (訓練ループ)．最初はとても大きな損失値を出しますが, 訓練が進むにつれて損失率が低くなっていきます．(下の図を参照してください)

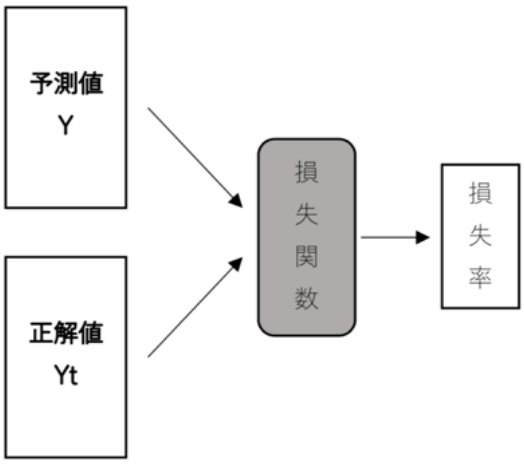


図 1.1　予測値と正解値から損失率が出てくる

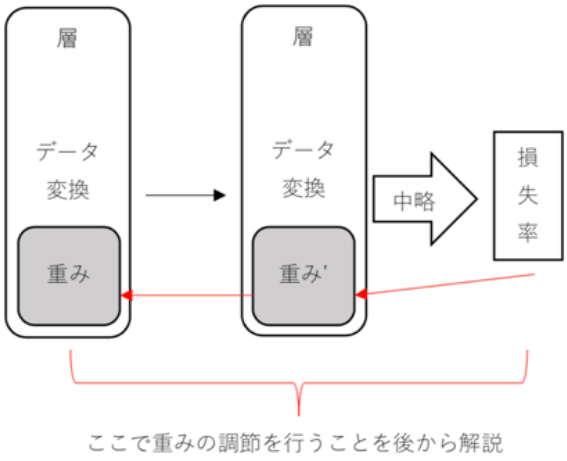


図 1.2　損失率をもとに重みを調整するイメージ

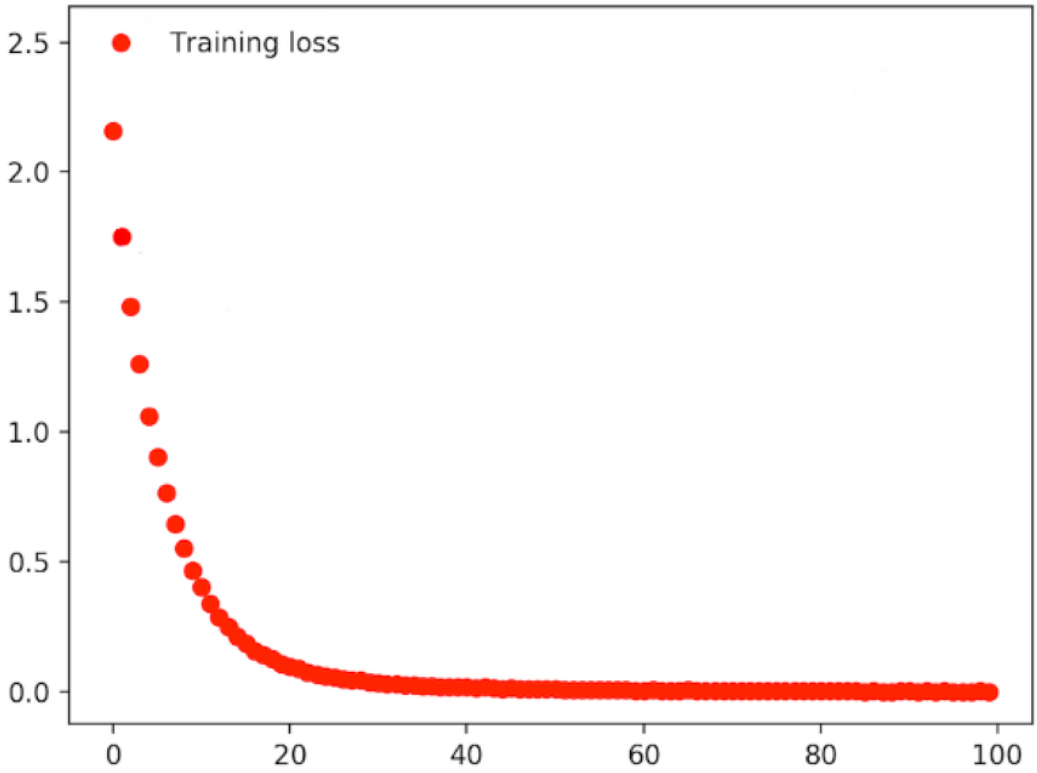
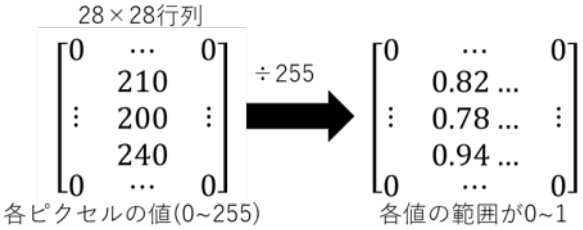


図 1.3　訓練が進むにつれて損失率が下がって行く様子.(横軸は訓練データセットを何周したか, 縦軸は loss の値を表す.)

1.3　学習を行う前に

深層学習は主に画像処理や音声処理、自然言語処理などに利用されますが、その際、画像データや音声データ、テキストデータを学習に用います。そこで必要なデータを集めるわけですが、集めてきたデータをそのままは扱えません。いくらかの**前処理**を行う必要があります。例えば以下のような画像データを用いるとします。

この画像は 28px × 28px の白黒の画像で、各ピクセルが 0〜255 の値（どれだけ黒いか）を持っています。つまり、28 × 28 の行列でこの画像を表現することができるということです。ここで、それぞれの値を 255 で割ります。すると、以下のように各値が 0〜1 の間に収まるようになります。



また、カラー画像についても考えてみましょう。カラー画像では各ピクセルがそれぞれ RGB（赤緑青）の 0〜255 の値を持ちます。このような、各入力が持つ成分の数をチャンネルと呼びます。つまり、白黒画像は 1 つのチャンネルを持ち、カラー画像は 3 つのチャンネルを持つ、と言います。

以上のような、データを特定の範囲に変換する処理を**正規化**と呼びます。ここでは前処理としてシンプルな正規化を行っただけですが、一般に前処理を行うと学習の性能が向上することが知られています。他に学習の前にやらなければならないこととして、データを学習に用いる際、訓練に用いるデータ（訓練データ）とその性能のテストに用いるデータ（テストデータ）に分割する必要があります。なぜそんなことをする必要があるかというと、訓練データをテストに使ってしまうと**過学習**が起こってしまうからです。

過学習とは、モデルがあるデータセットに過剰に適合してしまうことを指します。つまり、あるデータセットで学習を行ってそのデータセットに対して答えを予測する精度が高くなっても、他のデータに対しては精度が低いままになってしまうことです。機械学習を行う目的は実際の画像や音声に対して、それが何なのかという答えを予測することなので、未知のデータに対しても正しい答えを与えられる汎用性（**汎化性能**）が求められます。そこで、テストデータをあらかじめ訓練データと分離しておくと、テストデータはあくまで学習を終えたモデルの評価にのみ用いられ、学習に用いられることはありません。そのため、過学習を防ぐことができ、モデルに対して正しい評価をすることができます。

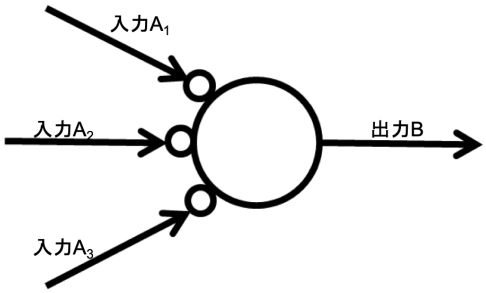
以上で学習の前に行う処理の概要について説明しました。次は実際に学習の中身、ニューラルネットワークについて見ていきましょう。

1.4　パーセプトロン

脳では神経細胞（ニューロン）から神経細胞に電気刺激を伝えます．これをモデル化 (数学的に表せる形にすること) したものがパーセプトロンです．パーセプトロンは次のような模式図で表されます．

神経細胞が刺激を伝達する際の特徴は次の 2 つです．

1. 神経細胞は一定以上の刺激を受けると次の神経細胞に刺激を伝える
2. 神経細胞どうしの繋がり具合により，電気刺激の強さが異なる



これらをパーセプトロンでは数学的に表現します．1. を表現するのが**活性化関数**，2. を表現するのが**重み**です．

活性化関数

一定以上の刺激を受けとったときに，信号を次に伝える役割が活性化関数でした．従って，あるしきい値以下の値を代入すると 0 を返し，それより大きな値を代入すると 1 を返すような関数が考えられます．

$$h(x) = \begin{cases} 0 & (x \leqq \text{しきい値}) \\ 1 & (x > \text{しきい値}) \end{cases} \tag{1.1}$$

このような関数をステップ関数といいます．

この他に，「小さな値を受け取ると 0 に近い値を返し，大きな値を受け取ると大きな値を返す」という性質をもつ関数

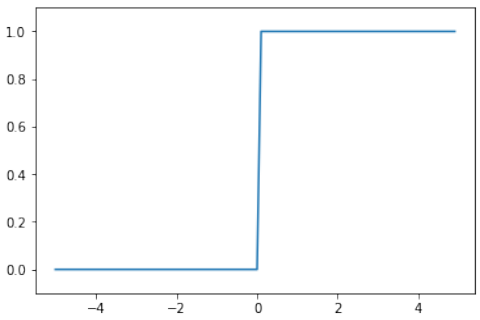


図 1.4　ステップ関数，しきい値が 0 の場合

も活性化関数として用いることができます．そのような例を見ていきましょう．

$$h(x) = \frac{1}{1 + e^{-x}} \tag{1.2}$$

$$h(x) = \begin{cases} x & (x > 0) \\ 0 & (x \leqq 0) \end{cases} \tag{1.3}$$

(1.2) をシグモイド関数，(1.3) を ReLU 関数といい，これらのグラフはそれぞれ次のようになります．

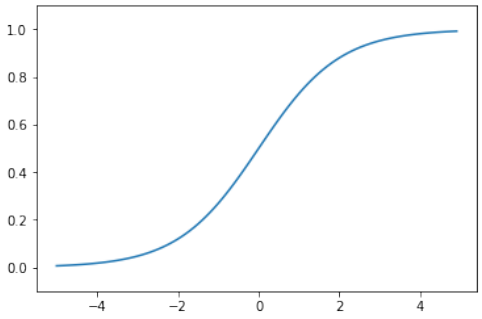


図 1.5　シグモイド関数

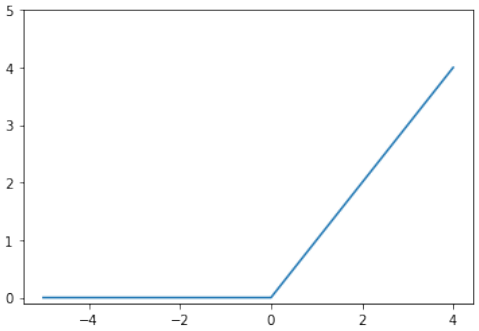


図 1.6　ReLU 関数

これらの活性化関数が近年のニューラルネットワークで有効であることがわかっています．

重み

神経細胞どうしの繋がり方により，次の神経細胞に伝わる刺激の強さが異なり，これを表現するのが**重み**でした．重要な入力を重く受け止め，さほど重要でない入力に対しては軽く受け止める，ということです．数学的に表現するには，重要な入力には大きな数をかけておき，重要でない入力には小さな数をかけるということをします．

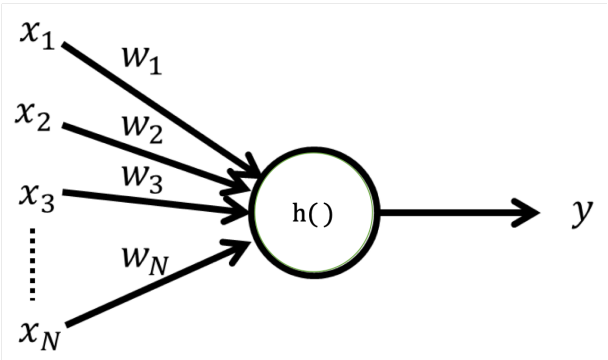


図 1.7　ニューロンのモデル

これらの要素を取り入れた模型が左図のようになります．
入力が $x_1 \ldots x_N$, 各重みが $w_1 \ldots w_N$, 活性化関数が $h(x)$, 出力が y です．このとき，これらは

$$y = h(w_1x_1 + w_2x_2 + \ldots + w_Nx_N)$$

と表されます．

1.5　ニューラルネットワーク

上述のパーセプトロンの構造を組み合わせると，あらゆる関数の近似を表現できるようになります．特に，下図のように幾層にもこのモデルを重ねたものを，**ニューラルネット**といいます．

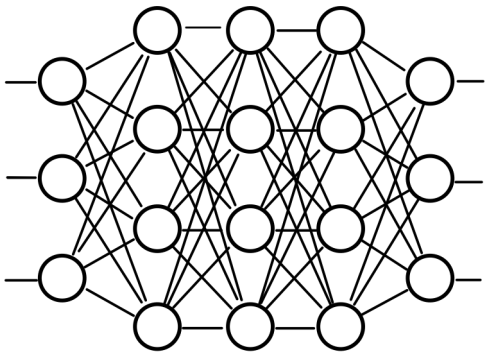


図 1.8 多層のニューラルネット

ニューラルネットではあらゆる関数の近似ができると述べました．例えば目に見えた物と，その物の名前を対応させるような関数にニューラルネットを近似させることができれば，これを画像認識の人工知能として用いることができます．このようにニューラルネットを適切なものにしていくには，層の数や，各層のノードの数，活性化関数として何を用いるか，そして重みを適切に設定しなければなりません．特に，重みの数は巨大なニューラルネットになると膨大で，とても人間が設定できる数ではありません．ところが，なんと適切な重みをデータから自動的に決定する方法が存在します．これこそが近年の深層学習の素晴らしい点なのですが，この方法について，次章で解説します．

1.6 学習とは

これまで，複数層のニューラルネットに入力となるデータを与えることで出力となるデータが計算されることを学びました．各ノードの重みパラメータをうまく調節することで，より性能の良い (より正確にものを認識したりできる) ニューラルネットを作ることができます．このパラメータを調節する過程のことを**学習**といいます．Deep Learning の画期的な点は，データを与えると機械が自動で重みパラメータを決定してくれる点にあります．つまり，教師となるデータがあれば機械が勝手に学習をしてくれるのです．近年，人工知能が世界的なブームとなったきっかけはこの点にあります．ここでは，どのように機械が自動で学習をするかを学びます．

ニューラルネットを訓練するための教師となるデータは，問題と解答がセットとなった問題集のようなものです．例えば，画像の分類をするニューラルネットを作るための教師データは，たくさんの写真と，それに何が写っているかの情報のセットになります．Deep Learning の学習の大まかなイメージは次の通りです．学習中のニューラルネットに問題データを入力すると，答えを出力します．この機械の答案と，正しい答えとの「ズレ」を数値化し，重みパラメータを変化させることで「ズレ」を減らします．これを繰り返し，「ズレ」が最も小さくなる重みを発見します．答え合わせを繰り返して少しずつ精度を上げていくイメージですね．重みが適切であれば，今後，教師以外のデータを与えても正しい答えを返すようになります．

では学習の過程を具体的に見ていきましょう．手書き文字認識をするための教師データセット”MNIST”を用いて手書き文字を認識するニューラルネットをつくることを考えます．MNIST では手書き数字のモノクロ画像 (問題) と，その画像が 0〜9 のどれなのかの情報 (解答) がセットになっています．入力となる画像データは $28 \times 28 (= 784)$ ピクセルのモノクロ (白から黒までを 256 段階で表現します) のデータですので，この画像データは 784 個の数字の集まりだと考えることができます．これを 784 次元のデータであるといいます．出力は，入力画像が 0〜9 のどれであるかについてのそれぞれの確率です．つまり入力が 0 である確率が 0.12，1 である確率が 0.03，2 である確率が 0.04・・・とい

うように，10 個の確率を出力として返します．これは 10 個の数字の集まりなので 10 次元のデータです．したがって，このニューラルネットは，784 次元の画像データを入力すると，10 次元の出力を返す関数だと考えることができますね．このような関数のことをモデルといいます．このモデルでは出力となる確率のうち，一番大きなものと対応する数字が，画像認識された数字だというわけです．

1.7 損失関数

正しい答えと，このニューラルネットモデルの導いた答えの「ズレ」はどのように数値化すれば良いでしょうか．モデルの答案は

$$[0, 0.1, 0.6, 0, 0.05, 0.1, 0, 0.05, 0.1, 0]$$

のように出力されます．左から順に $0, 1, 2, \ldots$ と対応していることにすれば，このモデルは入力画像を 2 であると考えているようですね．これに対し，入力画像が実際に 2 のとき，模範解答は

$$[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]$$

となります．これらのズレが数値化できたら良いわけです．モデルの答案と模範解答のズレを数値化する関数を**損失関数**といい，ニューラルネットでは次の 2 つがよく用いられます．

$$E = \frac{1}{2}\sum_{k=0}^{9}(y_k - t_k)^2 \tag{1.4}$$

$$E = -\sum_{k=0}^{9} t_k \log y_k \tag{1.5}$$

E はズレ ($\boldsymbol{E}rror$),t_k は教師データ (模範解答)，y_k はモデルの出した答案になります．今回は 9 次元のデータなので $k = 0, 1, \ldots, 9$ です．これらの式は，モデルの予測結果と教師データが一致しているときに 0 になり，ズレが大きくなるにつれ E も大きくなることを確認してください．(1.4) を 2 乗和誤差，(1.5) を交差エントロピー誤差といいます．誤差を算出する方法はたくさんありそうですが，これらはニューラルネットをるくるのに都合の良い性質を備えているのでよく用いられるようです

損失関数の最小値を求めて

ニューラルネット内の重みを微調整し，損失 E を減らしていくことが学習だと述べました．つまり学習後のモデルの E は重み W の関数だと考えられます．従って，

$$E = f(W)$$

と考えられ，この $f(W)$ の最小値を求めることが学習することだと言えます．ただ，この W というのはニューラルネットのたくさんの重みのことなので，f は多変数の関数です．では，簡単なニューラルネットを例に W を調整し，損失関数を最小化する方法を学びましょう．

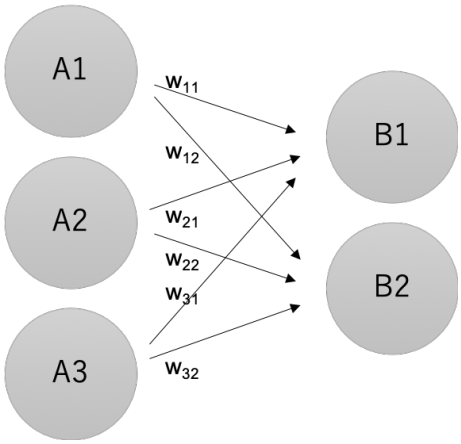


図 1.9 1 層のニューラルネット

左図は 1 層の単純なニューラルネットの例です．入力が $A_1 \sim A_3$，出力が B_1, B_2 で，それぞれに重みを持っています．つまりこのモデルは 3 次元の入力，2 次元の出力を持ちます．活性化関数が $h(x)$ であるとすると，

$$\begin{cases} B_1 = h(w_{11}A_1 + w_{21}A_2 + w_{31}A_3) \\ B_2 = h(w_{12}A_1 + w_{22}A_2 + w_{32}A_3) \end{cases} \tag{1.6}$$

のように表されます．
模範解答を $(\tilde{B_1}, \tilde{B_2})$ であるとし，損失関数として (1.4) 式の 2 乗和誤差を用いたとすると E は

$$E = \frac{1}{2}\{(B_1 - \tilde{B_1})^2 + (B_2 - \tilde{B_2}^2)\} \tag{1.7}$$

従って，このニューラルネットにおける学習とは，(1.7) 式に (1.6) 式を代入し，適切な w_{ij} を決めて E の最小値を求めることに他なりません．この場合 6 変数の最小値を求めることになります．

1.8 勾配法

残念ながら，多変数関数の最小値を求めることは困難です．実際のニューラルネットでは重みは数千万個もあったりするので，真の最小値を求めることは不可能でしょう．しかし，局地的に小さいような値，1 変数関数で例えるならちょうど極小値のような値を求めることなら可能です．ですので，学習においてはこの局地的な谷底を目指すことにします．

w_{11} 以外の重みを固定し，w_{11} のみを変化させる場合を考えます．
w_{11} を小さな正値 Δw_{11} だけ増やし，E が $E + \Delta E$ になったとすると，E の w_{11} 方向の傾きは $\frac{\Delta E}{\Delta w_{11}}$ であると言えます．E を小さくするには，この傾きが正ならば w_{11} を少し減らし，負ならば w_{11} を増やせば良いです．

このようにして，すべての重み w_{ij} について傾き $\frac{\Delta E}{\Delta w_{ij}}$ を考えることができます．傾き $\frac{\Delta E}{\Delta w_{ij}}$ が急なものほど w_{ij} を動かした際 E に強い影響を与えるので，重み w_{ij} を調節する量は，$\frac{\Delta E}{\Delta w_{ij}}$ に比例した量にします．(比例定数を η とします)
従って，更新後の重みを $\tilde{w_{ij}}$ とすると

$$\tilde{w_{ij}} = w_{ij} - \eta \frac{\Delta E}{\Delta w_{ij}}$$

となります．このような重みの更新を繰り替えすと，いずれ E の極小値にたどり着きます．
この方法を**勾配法**といいます．

実際のニューラルネットでは教師データの中から一部をランダムに選び，そのデータ群の損失関数を減らすように重みを更新する，ということを繰り返します．これは，**確率的勾配降下法**といいます．
さらに，膨大な数の重みの傾きを効率よく求めるために，**誤差逆伝播法**というアルゴリズムが活用されています．

偏微分

多変数関数 $f(X), (X = \{x_1, x_2, \ldots\})$ のある変数 x_i 以外の変数を固定し，f を x_i について微分することを**偏微分**といい，

$$\frac{\partial f}{\partial x_i}$$

のように表します

1.9 誤差逆伝播法

誤差逆伝播法によって全体の重みのうちある一部が変化したときに最終的な出力がどのくらい変化するか, つまり dL/dw が各重みについて算出されます．(L は損失関数の値とします．) このアルゴリズムの優れている点を逆伝播の具体例とともに見ていきましょう．

逆伝播を考えるための簡単な例としてある試験の合計点を考えてみます．この試験の合計点は英語の素点がそのまま, 数学は素点の 2 倍が加算されることで決まります．この状況を図示すると以下のようになります

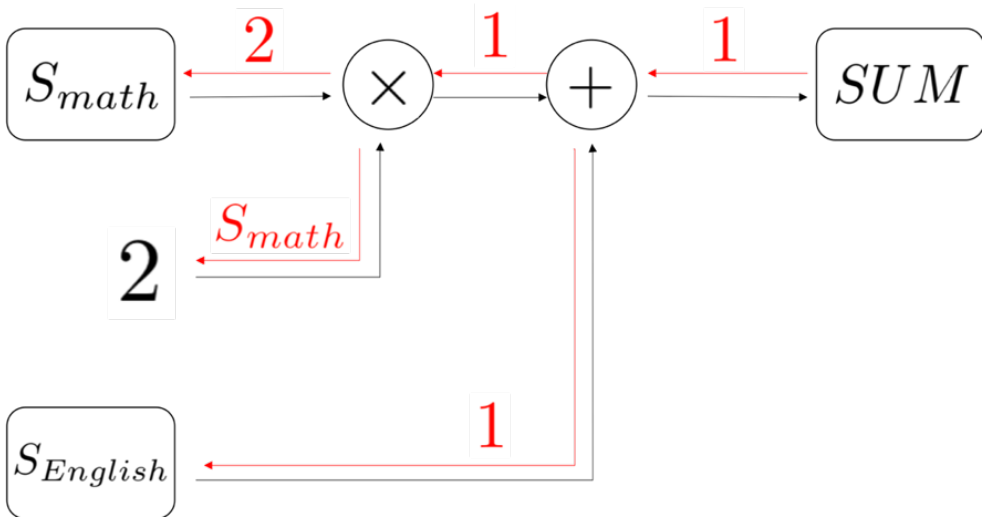


図 1.10 英語の素点と数学の得点 ×2 が加算される． 逆伝播がなぜその値になるかはのちに説明する．

数学の点数が 1 点変化したときの合計点への影響は 2 点であることがわかります．これは簡単な例なので暗算で求まるかもしれませんが, 上の図のように全体で行われている計算を段階ごとに微分可能な演算に分解して逆伝播を考えることでも求めることができます．

数学の点数の合計点への影響を考えるのに着目していたのは掛け算の段階での逆伝播の値ですね．合計点の変化を調

べるのに一旦英語の点数ことは忘れていても良いのです．逆伝播の優れているのはこのように**局所的な演算のみに着目して良い点**です．

前の段階で算出された微分の値 (この例だと足し算の段階から掛け算の段階へと流れている 1 という値) がその次の段階で用いられていることにも注目しましょう．逆伝播を利用するとそれまでの計算結果が次に使えるのが良い点です．

以上より逆伝播の優れた点として

1. 局所的な演算に着目することでそこでの微分の値が求まる
2. 途中の結果を利用できるので効率よく複数の微分を計算できる

ということがわかりました．ニューラルネットワークの膨大な数の計算を行うときに逆伝播の優れた点を活かすことが出来ます．

逆伝播をニューラルネットワークに適用するということは, 各層での重みごとにその値の変化が最終的な出力である「誤差」の増減にどれだけ影響を与えるかを算出するということです．

逆伝播について詳しく見ていきましょう．各演算の段階でそれまでの計算結果を用いて微分を計算する方法を説明します．

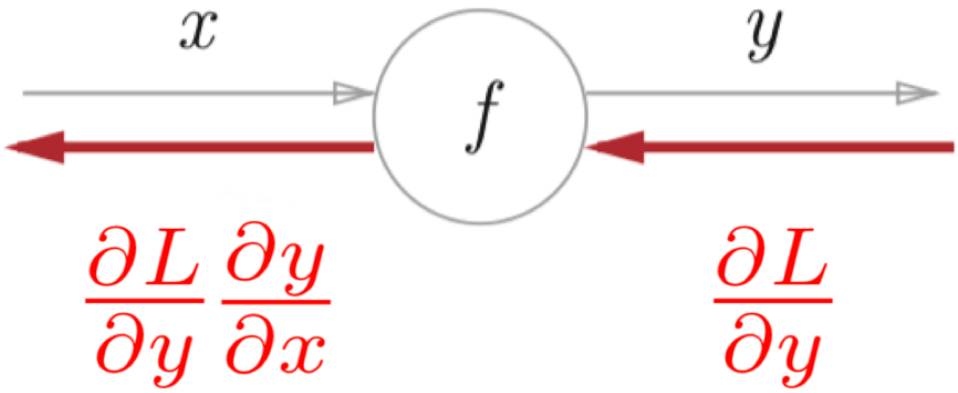


図 1.11　逆伝播の微分計算

この図は損失関数が L であるニューラルネットワークの一部を一般的に表したものです．f は微分可能な演算を表します．f への入力 x についての微分とは x が微小変化したときの L の変化の度合いで,

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x} \tag{1.8}$$

で表されます．局所的な微分 $\partial L/\partial x$ は その演算までの計算結果である $\partial|/\partial y$ と着目している演算の微分である $\partial y/\partial x$ を用いて表されていることがわかります．

式 (1.8) で用いられている合成関数の微分の性質を連鎖律といいます

連鎖率

合成関数の微分は合成関数を構成するそれぞれの関数の微分の積で表される.

例えば　$z = t^3, t = 2x + y$ の時

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial x} = 6(2x+y)^2$$
$$\frac{\partial z}{\partial y} = \frac{\partial z}{\partial t}\frac{\partial t}{\partial y} = 3(2x+y)^2$$

上式が成立します. ∂t の部分が打ち消されると思えば覚えやすいです.
この性質によりニューラルネットワークでは

$$\frac{\partial L}{\partial x_n} = \frac{\partial L}{\partial L}\frac{\partial L}{\partial x_1}\frac{\partial x_1}{\partial x_2}\cdots\frac{\partial x_{n-2}}{\partial x_{n-1}}\frac{\partial x_{n-2}}{\partial x_n}$$

のようにいま計算したい微分の値を各段階での微分の値の積として算出できます. ここでは $\partial x_{n-1}/\partial x_n$ が着目している演算の微分で $\partial L/\partial x_{n-1}$ と書き直せる部分がその演算までの計算結果ですね.

微分可能な演算として足し算と掛け算を例にあげ, 上に出したある試験の例における逆伝播を説明します.

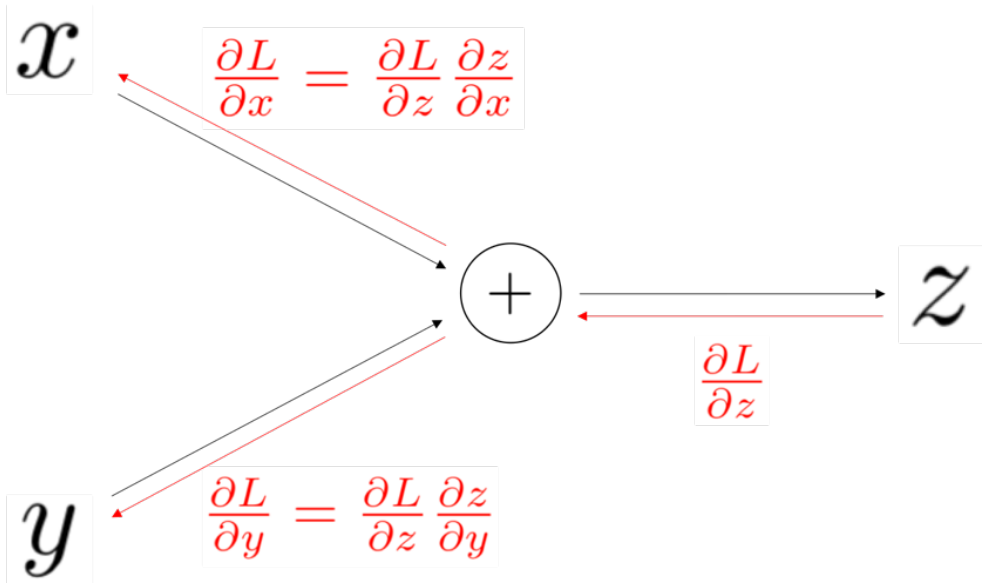


図 1.12　英数試験得点の計算グラフ

図 1.9 に示すように $x + y = z$ を考えると

$$\frac{\partial z}{\partial x} = 1, \ \frac{\partial z}{\partial y} = 1$$

なので

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\cdot 1, \ \frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\cdot 1$$

となります.

上の例で英語の点数と数学の点数を足す演算において逆伝播を考えると

$$2S_{math} + S_{english} = S_{sum}$$

なので

$$\frac{\partial S_{sum}}{\partial 2S_{math}} = 1$$

と

$$\frac{\partial S_{sum}}{\partial S_{english}} = 1$$

がつぎの計算へと逆伝播していきます.

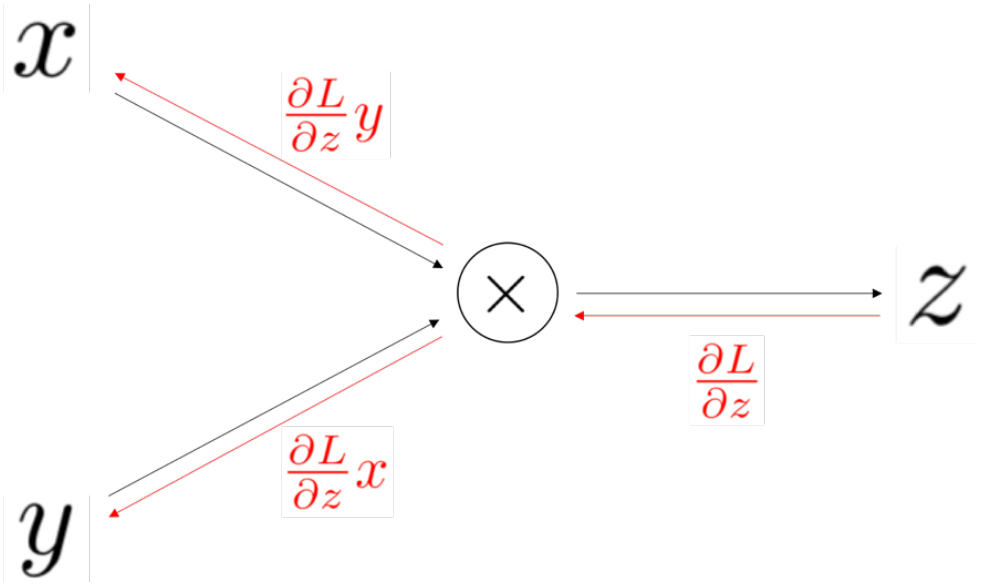


図 1.13　試験得点の逆伝播

図 1.9 に示すように $xy = z$ を考えると $(\partial z/\partial x) = y, (\partial z/\partial y) = x$ なので
$(\partial L/\partial x) = (\partial L/\partial z)y, (\partial L/\partial y) = (\partial L/\partial z)x$ となります.

上の例で数学の点数を 2 倍するところにおける逆伝播を考えると $2S_{math} = 2 * S_{math}$ なので $\partial(2S_{math})/\partial(S_{math}) = 2$　であることより

$$\begin{aligned}\frac{\partial S_{sum}}{\partial S_{math}} &= \frac{\partial S_{sum}}{\partial 2S_{math}} \frac{\partial 2S_{math}}{\partial S_{math}} \\ &= 1 \cdot 2 \\ &= 2\end{aligned}$$

実際のニューラルネットワークでの逆伝播をみていきましょう式 (1.6) の重みと入力の積は行列の積として $y = Aw$ と表現できます. $A = (A_1, A_2, A_3)$, $w = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{pmatrix}$ です.

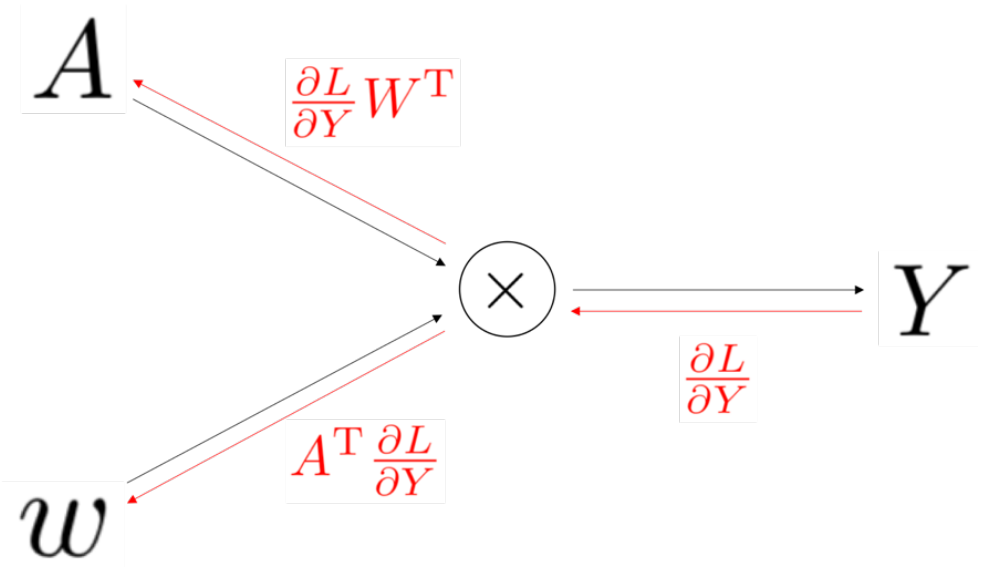


図 1.14 実際のニューラルネット誤差逆伝播

図 1.9 で示す演算は重みと入力の掛け算 (行列の積) なので, 重みに関する損失関数の勾配は以下の式で求めることが出来ます.

$$\frac{\partial L}{\partial w} = A^{\mathrm{T}} \frac{\partial L}{\partial Aw}$$

ここでは $\partial L/\partial Aw$ がこの演算までの逆伝播の値で A^{T} がこの演算における微分の値です.

実際に使われているニューラルネットワークは多くの種類の演算の組み合わせでできています. それらは微分可能な演算なので誤差逆伝播法をニューラルネットワーク全体に渡って実行することができます.

学習の手引き

ここまでディープラーニングについての大まかな解説を行ってきましたが、使われている技術や理論についてもっと詳しく知りたいという方や、実際に自分で AI を作ってみたい！という方もいらっしゃるかもしれません。そこで、ディープラーニングについて学ぶために必要な知識や参考になる書籍をいくつか紹介していこうと思います。

参考書籍

- 『ゼロから作る Deep Learning』斎藤康毅［著］オライリー・ジャパン
  ディープラーニングを基礎から学びながらライブラリに頼らず実装していく本。初学者でも理解できる名著。の自然言語処理編も出版されておりこちらも良書。
- 『Python と Keras によるディープラーニング』Francois Chollet［著］株式会社クイープ［訳］巣籠悠輔［監訳］マイナビ出版
  Keras を用いて手を動かしながらディープラーニングについての理解を深める本。著者は Keras の作者である。コンペも視野に入れた本なので実践的な使い方まで知ることができる。

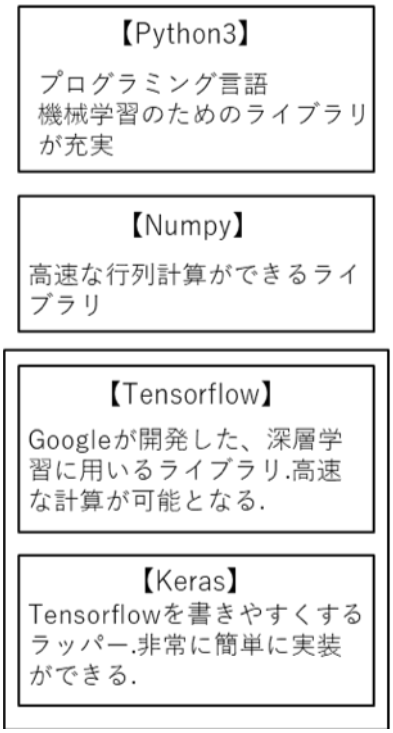


図 1.15　実装

理論

機械学習を理論から理解するには数学や統計についての知識が必要。多くの書籍で高校数学は前提となっています、頑張りましょう！

- 「ゼロから作る Deep Learning」斎藤康毅［著］オライリー・ジャパン
  最低限必要となる行列や偏微分について軽く説明してあります。機械学習についてもっと理解したいなら他の書籍も読みましょう。
- 「パターン認識と機械学習」　C.M. ビショップ［著］元田浩［訳］
  ベイズ確率の視点から機械学習の理論について解説した本。

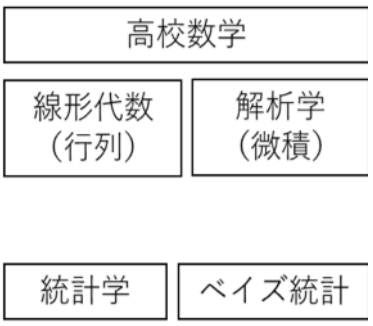


図 1.16　キャプション

第 2 章

コンピュータビジョン (画像認識)

コンピュータビジョンというのは, **人間の視覚が行うことを自動化することを目指す分野**です. その例として, 画像や映像の中に写っている物体の識別や分類をコンピュータに行わせるということが挙げられます. 具体的に言うと, iPhone の’写真’アプリでは保存されている画像が「食べ物」「動物」「スポーツ」といったカテゴリに自動で分類されるようになっていたり, 人物の顔を自動で見分けて同じ人の顔をまとめて表示できるようになっていたりします.

畳み込みニューラルネットワーク (CNN) はコンピュータビジョンにおいてよく用いられるディープラーニングモデルです. ここでは画像を分類する際に用いられる CNN について見ていきたいと思います.

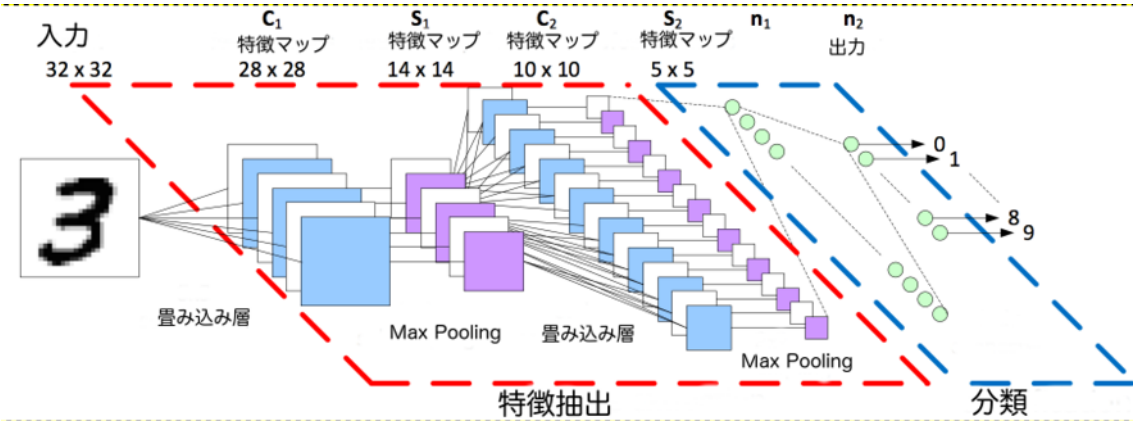


図 2.1　CNN の構造を表す図

CNN はネットワークの前半で**入力画像の特徴量を抽出**し, 後半のでは抽出された**特徴量をもとに画像を分類**にかけます. 後半の分類器については前節で説明したニューラルネットワークと一緒なのでここでは前半の特徴抽出について見ていきましょう.

2.1　特徴抽出

畳み込み層が学習するのは局所的なパターンです. サイズが 3 ピクセル× 3 ピクセル程度の重み行列（ウィンドウ) が画像上の各場所でその場所の画像の行列との内積を計算しながら移動し, 最終的に各場所での内積の値がそれぞれのピ

クセルに格納された**出力特徴マップ**を返します.

下の画像の例を説明します. 学習された重みは中央の 3 × 3 の行列であり, もともとの画像 (左側の×印) からウィンドウの検出するパターンが右側の特徴マップに反映されていることがわかります. つまりこの例の処理においては画像中における左上から右下にかけての斜め方向のエッジの検出をしており, それを×印の 1 つの特徴として返していることになります.

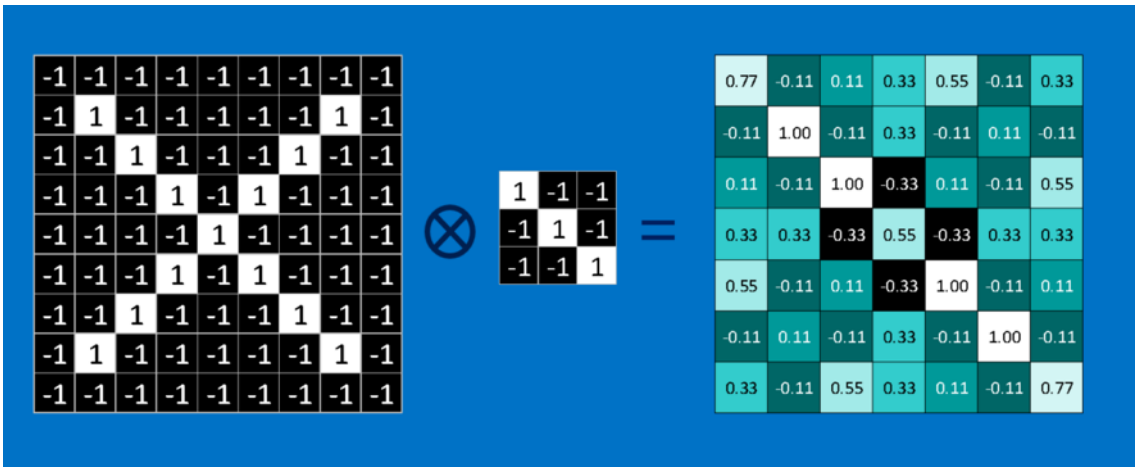


図 2.2　どのようにして局所的なパターンが検出されるのか

1 つの畳み込み層の中にこうした重み行列は何枚か存在します. うまく様々な特徴が検出できるような**重み行列**は学習によって自動で獲得されます.

元々の画像に RGB という 3 つのチャンネルがあるように, 出力特徴マップにもそれぞれが画像の特定の側面を保持している重みの数に対応した個数のチャンネルがあります. 図 2.2 の特徴マップも 1 つのチャンネルの例) 各チャンネルが保持しているのは RGB の情報ではなく**画像の各場所における特定のパターンの有無**です. 特定のパターンの例としてあげられるのは, ある方向へのエッジやエッジの組み合わせでできる**模様**です. さらにその模様が組み合わさってできる**人の顔というパターン**もそれに含まれます.

図 2.3　CNN の最初の畳み込み層において学習された重みを可視化したもの

CNN の特徴として, 層が深くなるにつれて学習するパターンが大きなものになるということがあげられます. 1 つ目の畳み込み層は画像中の**エッジや色の塊**といった小さな局所的なパターンを学習し, その次の層で**エッジの組み合わせでできる局所的な物体**の検出を学習し, より深い層になると局所的な物体の特徴の組み合わせから**「人」や「猫」の顔**などの高レベルな概念を学習するようになります. これによって複雑な画像の漸進的な学習が可能となっています.

また, 出力された特徴マップに対して Max Pooling と呼ばれる演算を行うことで処理するべき特徴マップを小さくすることと 1 つのウィンドウのカバーする (元の画像に対する) 範囲を大きくします. 画像の漸進的な処理を効率的に計算できるのはこのためです.

このような畳み込み演算と Max Pooling を用いた特徴マップの縮小を繰り返して画像の特徴量が抽出されます.

2.2　神経科学と畳み込みネットワークの関連性

畳み込みニューラルネットワークのいくつかの設計原理は神経科学の分野から影響を受けたものです.

神経科学者の Hubel と Wiesel は猫の前に置かれたスクリーン上に投影された画像に対して猫の脳がどのように反応するかを観察しました. その結果, 視覚システムのはじめの部分ではニューロンはある方向を向いた棒状のものなど光の特定のパターンに強く反応するがその他のパターンにはほぼ反応しないということがわかりました.

脳の一次視覚皮質 (V1 と呼ばれる) に注目します. V1 は視覚的入力に対して処理を行う脳の最初の領域です. 概要としては目から入った光が網膜を刺激して, 網膜上のニューロンが画像に前処理を加えます. その画像は視神経と外側膝状体と呼ばれる脳領域を通過して V1 へといきます. V1 の特徴として

1. 網膜上の画像の構成を配置するような二次元構造になっている
2. 多くの単純細胞が含まれている
3. 多くの複雑細胞が含まれている

畳み込みニューラルネットワークは V1 のこうした特徴を捉えるように設計されています.

具体的には畳み込みネットワークの出力する特徴マップそれぞれのチャンネルは縦横の二次元マップであること (1. に対応), 畳み込みネットワークのパターン検出ウィンドウは単純細胞の働きを模倣していること (2. に対応), 複雑細胞の働きは畳み込みネットワークの Max Pooling に着想を与えていること (3. に対応) です.

V1 のこうした特徴は脳内の視覚システムの他の領域でも成り立つと考えられます. これは CNN において**ウィンドウによるパターンの検出**, Max Pooling が画像に対して繰り返し適用されることと捉えることもできます.

V1 の単純細胞が反応する画像の局所的なパターンを視覚化することが可能です. 逆相関法と呼ばれる手法なのですが, まず電極をニューロンに取り付けます. 次に動物の網膜の前に白色ノイズの画像をいくつか見せ, それらがどのようにニューロンを活性化するのか調べることでニューロンが反応するパターン (**ニューロンの持つ重み**) を知ることができます.

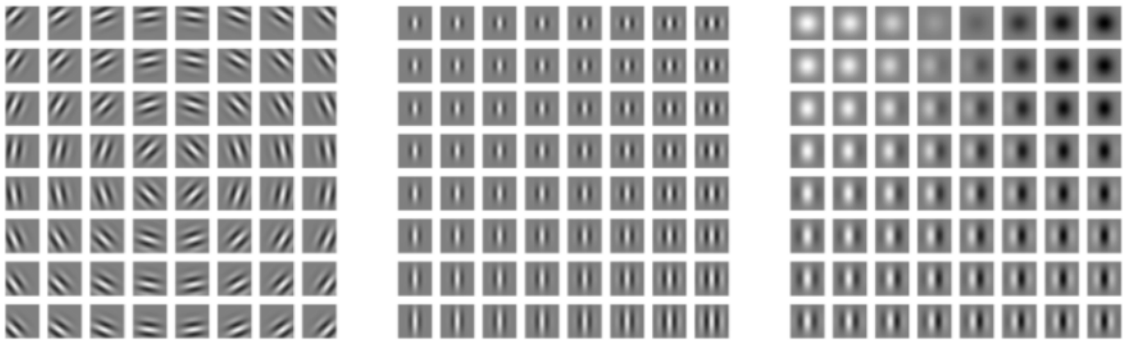


図 2.4　ニューロンの持つ重みを可視化した画像

ニューロンの持つ重みはガボール関数と呼ばれる関数で近似されることが知られており, それを様々なパラメータ設定で図示すると上の画像のようになります. 単純細胞は特定の場所での特定の方向への輝度の特定の空間周波数に反応することがわかります.

図 2.3 と図 2.4 を見てわかるように,CNN **で学習した重みと** V1 **の細胞で用いられている重み**を比較すると V1 は視覚システムの最初の段階であり, そこで検出されるパターンは CNN の場合と同様に局所的なエッジということがわかります. これによって神経科学と機械学習の対応関係が明らかになりますね.

第 3 章

自然言語処理

自然言語処理は、コンピュータに我々が使用する言語を理解させようとする、人工知能研究の一分野です。古くからそのような取り組みは行われてきましたが、近年ディープラーニングによって性能の大きな向上を遂げています。ここでは、自然言語処理についての大まかな説明を行おうと思います。

3.1 理論のあらまし

単語の意味をコンピュータに理解させようとするとき、古くからの手法では人力によって単語をグループごとに分類して「辞書」を作ろうとしてきました。しかし、この手法では単語の細かいニュアンスを表現できない、莫大なマンパワーが必要、新しい単語に対応できない、などの問題点がありました。

そこで、単語ごとの意味を表現するために用いられた考えが**分布仮説**です。分布仮説とは、単語の意味はその周りに現れる単語によって表現されるという考えです。例えば、

- I like dogs.
- I like apples.
- I study English.

ような文があったとして、各単語の周りにある単語を行列にして表にすると（これを**共起行列**と言います）

count	I	like	dogs	apples	study	English
I	0	2	0	0	1	0
like	2	0	1	1	0	0
dogs	0	1	0	0	0	0
apples	0	1	0	0	0	0
study	1	0	0	0	0	1
English	0	0	0	0	1	0

これを見て単語の意味が表現されているとは感じられないと思いますが、用いる文を多くすればより正確な表現が可

能となります。一方で、文が多くなればそれだけ用いられる単語も増えていき、共起行列が非常に大きくなります。また、見てわかる通りこの行列には 0 がとても多いです。つまり無駄な部分が多いと言い換えることができます。そこで、より密な表現で単語を表したいと思います。

3.2 Word2vec

Word2vec は単に周りの単語を数えるのではなく周りの単語からその間にある単語を推論することで単語をベクトルとして表そうとする手法です。word2vec には CBOW モデルと skip-gram モデルが存在しますがここでは skip-gram モデルを紹介します。
例文として I want to play baseball. を用いるとすると、play からその周りの単語を推測しようとするモデルです。

ここでニューラルネットワークが登場します。このときの入力は one-hot **表現**で表されるベクトルです。one-hot 表現とは特定の単語に対応する要素だけ 1 で他の要素は 0 となるベクトルです。

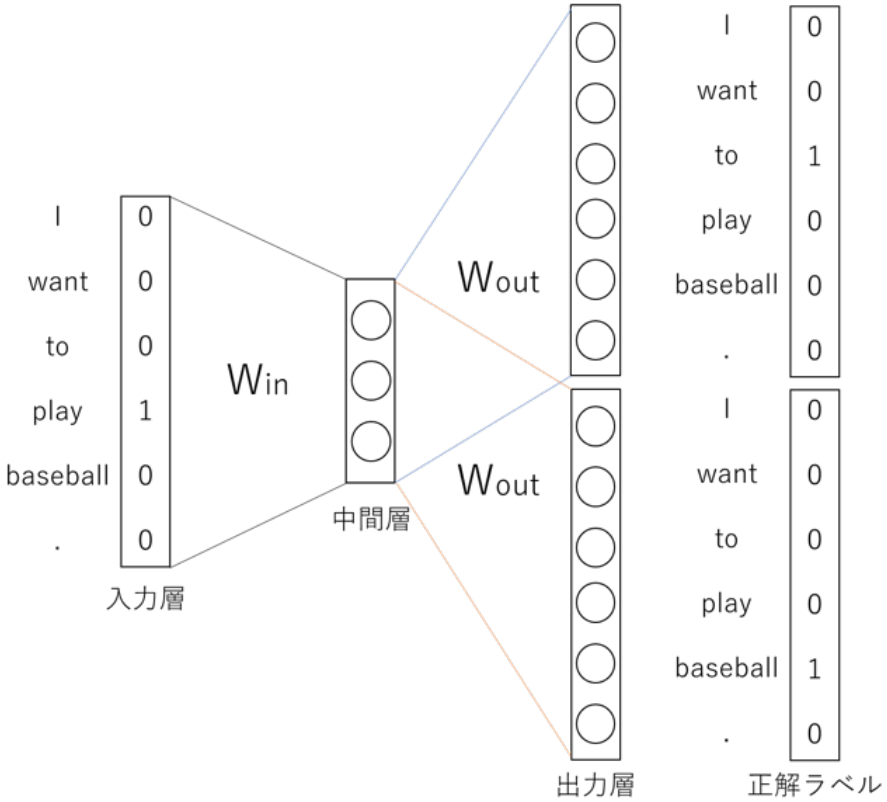


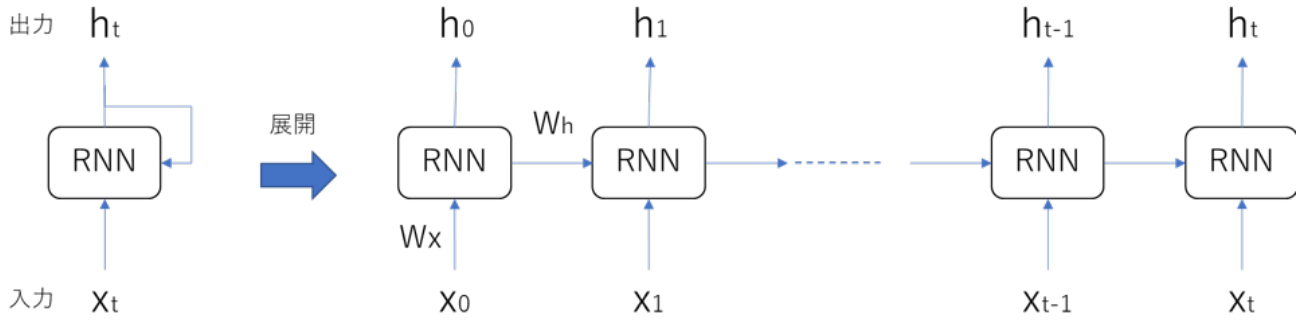
図 3.1 play に対応する one-hot 表現 $[000100]$

出力は各単語が [?] に入る確率のベクトルになります。正解の単語 (to, baseball) に対応する出力の要素が 1 に近づくように学習を行うと、学習した重み（Win）がその単語（play）の意味を表すベクトルとなります。おもしろいの

は学習した単語のベクトルは加算減算が行えるということで、例えば「king　man + woman = queen」という計算が成り立ちます。

3.3　RNN

文章生成や機械翻訳などのタスクを行う際には時系列データを扱う必要があります。しかし、これまでのニューラルネットはフィードフォーワード（流れが一方向）であり、単語の並び方を考慮できていませんでした。また、文章を入力とするとき、その長さは一定ではなく可変長でありそれを解決するためのネットワークが RNN（Recurrent Neural Network）です。Recurrent は日本語では「再発する」「周期的に起こる」「循環する」というような訳が当てられます。詳しく説明すると長くなるのでここでは図でざっくりとした説明をします。



入力は t 個の単語からなる文です。各 RNN レイヤで h が出力されると同時に次のレイヤにもその出力を渡します。h は「状態」を表し、レイヤを伝わるにつれ次々と更新していると言えます。この出力 h のことを**隠れ状態ベクトル**と呼びます。これにより RNN では過去の単語を記憶しているのです。

自然言語処理の手法についてここでは軽く説明するにとどめましたが、実際はもっと複雑だったりするので興味がある人は書籍等で勉強してみるといいと思います。

第 4 章

深層学習と自動会話文生成

前章「自然言語処理」では、人間の言葉を分散表現にする Word2Vec や、状態推移を学習できる RNN を紹介しました。それでは、実際に深層学習を用いて会話文を生成するにはどうすればいいのでしょうか？これは、以下のような問題とみなすことができます。

タスク　N 個の単語から成る文章 $X = \{x_1, x_2 \dots x_N\}$ が与えられたとき、文章 $Y = \{y_1, y_2 \dots y_N\}$ を予測する。ただし、文章中のそれぞれの単語は Word2Vec などで生成したベクトル、もしくは単に単語に ID を振ったスカラーとする。

人が文章を書くとき、そこには規則性があります。例えば、「私」という単語の後には「は」が 60% の確率で出てきて、「京大」が 1% の確率で出るとします。これは、

$$p(\text{は} \mid \text{私}) = 0.6$$

$$p(\text{京大} \mid \text{私}) = 0.01$$

のような条件付き確率として表すことができます。文章はさらに長い単語列で表されているので、その場合は

$$p(\text{京大生} \mid \text{私}, \text{は})$$

のように表現できます。この確率は単語の選び方によって遷移してゆき、それっぽい文章を生成するには、この確率が高くなるように単語を選べば良いのです。過去に出現した単語の列から新たな単語を予測するという意味で、文章は時系列データとみなすことができます。

深層学習を用いて時系列データを学習するには、前章で触れた RNN が良く用いられます。特に文章を生成するモデルとして Seq2Seq というものが有名で、これは与えられた文章をモデルの内部表現に変換する「エンコーダー」と、その値を受け取って新しく文章を生成していく「デコーダー」から成り立っています（図 4.1）。

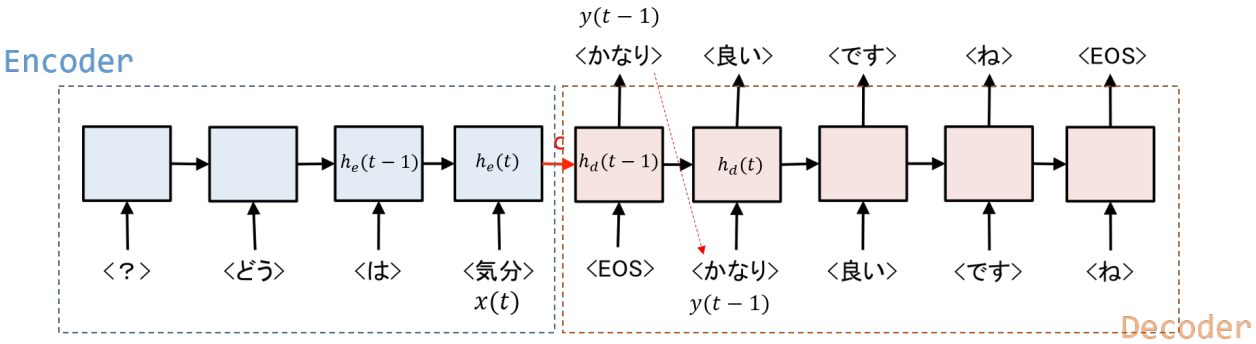


図 4.1　Seq2Seq モデル 注) 誤：中央付近の <EOS>　正：<BOS>

まず、エンコーダーの RNN には内部状態（ステート、図 4.1 では c）を保持する機能を持つ「LSTM」を用います。t 番目の単語 x_t を順伝播させる際、一つ前の単語 x_{t-1} も入力として利用し、このような具合で今まで登場した単語を再帰的に入力していきます。この際出力されるものは無視して、最終的なステートを取り出し、デコーダー側に伝えてあげます。

デコーダー側でも LSTM を用いるのですが、最初の入力は文章の開始を示す記号（BOS）にします。この結果、p(かなり | BOS) $= 0.6, p$(あまり | BOS) $= 0.3$ のような確率をたくさん得ることができます。一番高い確率を再度 LSTM に入力すると次の単語が予測されるので、これを繰り返していくと最終的に文章が生成されるというわけです。しかし、毎回一番確率の高い単語を選べばよいというわけではありません。なぜなら、各単語の確率を掛け合わした結果、途中でほかの単語を選んだ方が最終的な確率が高かったかもしれないからです。このような問題に対処するため、毎回確率の高い単語 N 個を選んで文章を生成していく探索方法が考えられます。ここでのアルゴリズムはビームサーチと呼ばれるものが採用されているので、詳しくは調べてみてください。

さて、このようなシステムを実装する場合、次の作業が必要になります。

1. 会話のデータセットの用意
   これが一番難題です。英語のデータセットならまだしも、日本語のデータセットを探すのは至難の業です。昔は Twitter の Userstream API などで自分で取得することができましたが、この API が廃止されてしまいました。
2. 文章を単語に分解する「わかち分け」
   これは形態素解析ソフト「Juman」や「MeCab」などで簡単にできます。
3. モデルの構築
   Pytorch や Chainer、Tensorflow などでモデルを構築します。ネットで検索すれば沢山実装例が出てくるので、がんばってください。
4. 計算資源の用意と計算
   深層学習で学習を行う際、処理の高速化のために GPU が不可欠です。これを搭載した PC を用意するか、AWS や Azure などのクラウドサーバーを借りましょう。

Seq2Seq を利用すると、会話を生成するだけでなく機械翻訳も可能となります。入力から出力を推測するというタスクは全く同じで、データセットを変えれば比較的簡単に動かすことができます。興味があればぜひ挑戦してみてください。

第 5 章

量子コンピュータ

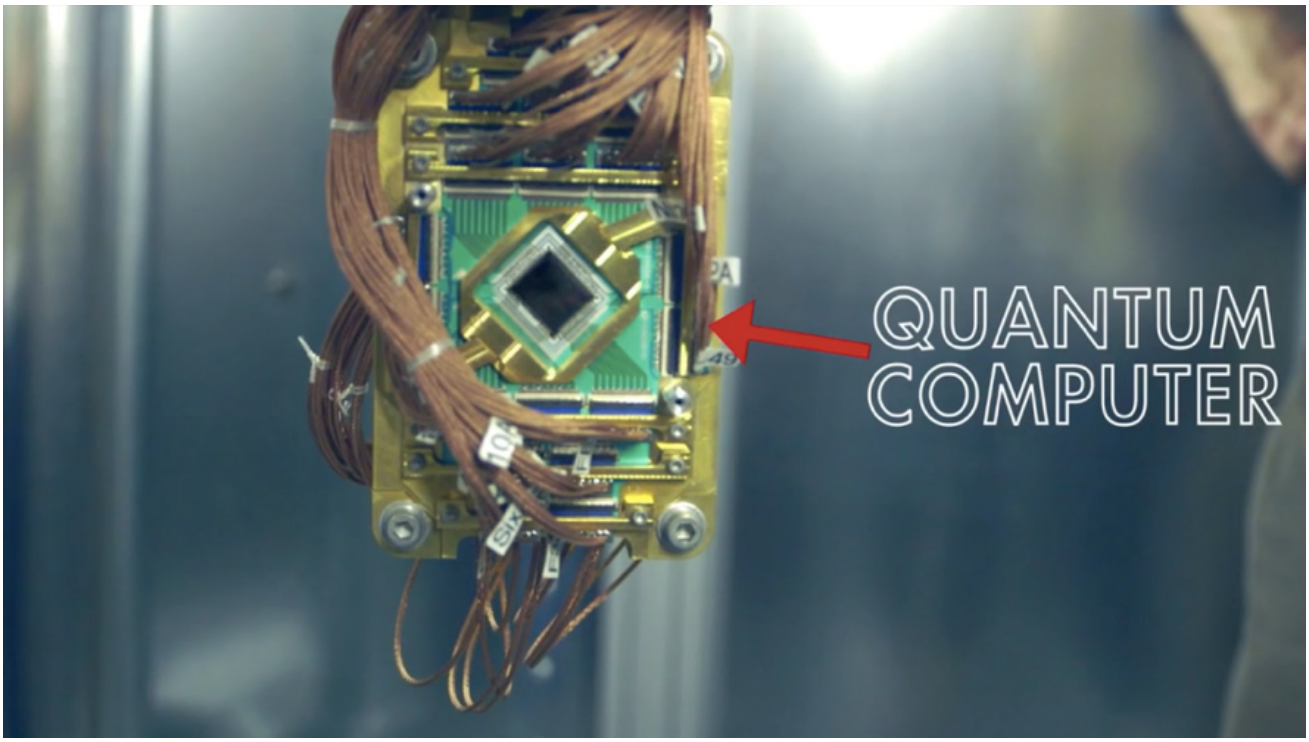


図 5.1　引用元 (https://www.youtube.com/watch?v=w_-_H9eBte8T)

5.1　はじめに

現在、二つのものがテクノロジーのフロンティアとして注目を集めている。

一つは人工知能である。特に最近ではハードウェアの進歩によりディープラーニングというモデルが実装可能になり様々な分野で活躍している。そのあまりの威力に社会問題を引き起こすのではないかと一部で懸念されるほどである。

もう一つが量子コンピュータである。長い間研究されていた量子ゲート式とは違う量子アニーリング方式の量子コンピュータは Google、NASA、Volkswagen など様々な機関や企業に導入されている [田中宗, 2017]。また量子ゲート式でもより多くの量子ビットを実装できたというようなニュースを目にすることも多い。

この二つのフロンティアが結び付くとどの様なことができるのか。研究段階のためまだわからないことは多いがいくつか期待されている例を紹介していきたい。なお高校生レベルの科学知識を持った人を読者として想定しているので量子力学などあまりなじみがないであろう分野はできるだけ丁寧に解説した（つもりである）。また本稿を書き上げる際『量子コンピュータが人工知能を加速する（日経 PB 社）』を参考にしている。量子アニーリングについて数学や物理の知識がなくても読めるので興味がある人は是非。

5.2　量子コンピュータの歴史

量子コンピュータとは大雑把に言えば量子力学的な現象や法則を利用して計算するコンピュータのことである。量子力学とは原子や分子などのミクロな世界を記述するための力学体系である。そこで記述される粒子の振る舞いは日常生活の感覚では理解できない不思議なものである（詳しくは後で解説）。

そんな不思議な性質を利用した計算機の必要性を訴えたのは物理学者ファインマンである。1982 年、彼はある種の量子力学的現象は古典計算機では効率的にシミュレートすることができないが、量子力学的性質を取り込んだ計算機ならば計算可能であることを示した。1985 年にイギリスの物理学者ドイチェが「チューリングマシン」という計算モデルを再定式化した「量子チューリングマシン」を提案する。さらに 1994 年にショアが「素因数分解」の「多項式時間量子アルゴリズム」を示した。大きな数の素因数分解が古典計算機では難しいことは昔から有名だった。実際、いくつかの大きい素数を掛け合わせることは簡単だが掛け合わされたものから元の素数を求めるのは簡単ではないことは実際に手計算でやってみてもらえばわかると思う。注目すべきは、この非可逆性が RSA 暗号という現在インターネットなど広く利用されている暗号の基本原理になっていることである。量子コンピュータがこの暗号を解く可能性が示されたことにより、社会への影響力は高く評価され多くの関心を集めた。

このように理論的研究は進んだがハードウェアの実装は非常に困難でなかなか進まなかった。しかしながらここ最近ハード面においても成果が出てき始めている。量子ゲート式は長い間研究され続け、50 量子ビットのものが開発されている（量子ビットについてはのちに解説）。また量子ゲート式とは違う量子アニーリング方式の量子コンピュータを D-Wave が開発し、2000 量子ビットを超えるものが出てきている。本稿では量子アニーリング方式について解説したいがそのためにも量子力学の基礎知識が必要となる。つぎに必要最低限の量子力学の説明をしよう。

5.3　重ね合わせ量子ビット

量子力学で重要な考え方である「重ね合わせ」について解説していきたいと思う。そのためにも前期量子論といわれている量子力学が誕生するまでの話をする。そして「重ね合わせ」の話をし、その性質を利用した「量子ビット」を解説する。量子ビットは量子コンピュータの基本単位である。

5.3.1　波と粒子

量子力学黎明期にとってまず重要になるのは Einstein の光の粒子説である。これは角振動数 ω 、波数ベクトル k の光はエネルギー $E = \hbar\omega$、運動量 $p = \hbar k$ の粒子の集まりであるという考え方である。これはこれまで波であると考え

られてきた光を粒子として捉える変わった考えだったが、これによって Compton 効果や光電効果などの物理現象を説明することができた。

ならば逆に粒子である電子や陽子も波として考えられるのではないだろうか。そうして de Broglie が考えたのが物質波である。実際に電子での「ヤングの実験」で電子波の存在が確かめられている。この実験は電子銃から電子をスクリーンに向けて照射するのだが、スクリーンと電子銃の間に二つのスリットを持った板を置いておく。電子を一つ照射すると電子はスリットを通ってスクリーンに当たり、当たった場所がスクリーンに跡として残る。はじめは電子が当たった跡が不規則な点の集合としか見えないが、実験を繰り返すうちに縞模様が現れる。光でも同じような実験ができる。このときも縞模様が浮かんでくるがこれは右のスリットを通った光と左のスリットを通った光が干渉しあってできる干渉縞である。干渉というのは波に特有の性質である。このことから電子にも波のような性質があることがわかる。

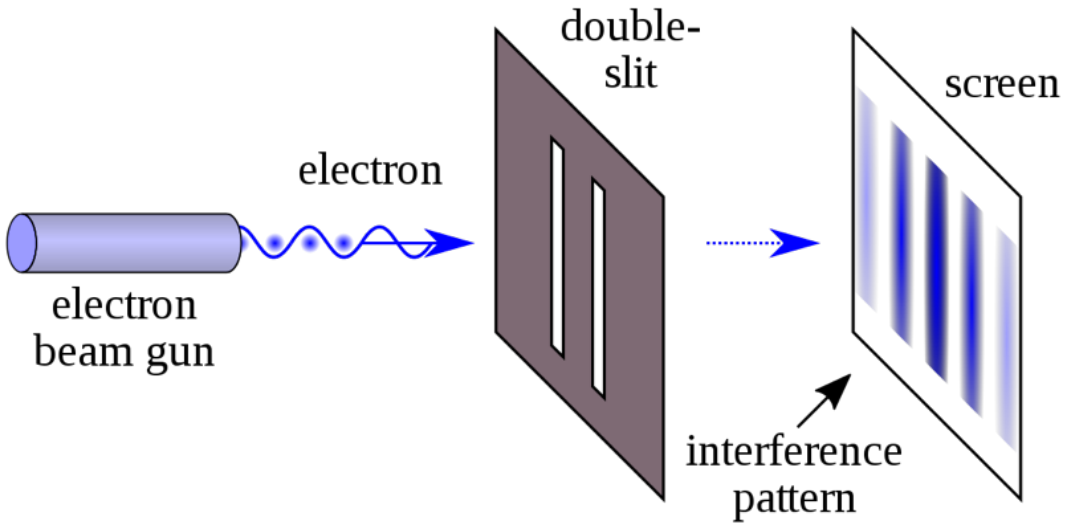


図 5.2　引用元 (https://en.wikipedia.org/wiki/Double-slit_experiment)

5.3.2　重ね合わせ

しかしセンサーで電子が左右どちらのスリットを通ったのかを測定すると不思議なことが起きる。なんと干渉縞は消えてしまうのだ。この出来事を説明するために二つ以上の状態が同時に存在するという「重ね合わせ」の考え方が出てくる。つまりこの考えが説明することには、干渉縞ができるのは電子が「右のスリットを通る」という状態と「左のスリットを通る」という状態が同時に存在しそれぞれが干渉しあうことによって縞模様ができる。それに対して左右どちらのスリットを通ったのかを測定してしまえば状態がひとつに確定していまい重ね合わせがなくなる。従って干渉縞がきえるというのである。

重ね合わせの考え方になじむためにも「シュレディンガーの猫」という重ね合わせについての有名な思考実験の例を挙げよう。箱の中に放射性物質であるラジウム、生きた猫、ガイガーカウンター、青酸ガス発生装置をいれておく。青酸ガス発生装置はガイガーカウンターが α 波を観測したら作動するようにしておく。ラジウムが一時間のうちに α 波を出す確率を 50% とし、実験開始から一時間後の状態を考えよう。α 波の発生というのは原子核が関わる量子力学的な現象である。従って一時間後には「α 波が発生した状態」と「α 波が発生しなかった状態」との五分五分の重ね合わ

せとなる。またこのことから実験の設定により「猫の生死」も五分五分の重ね合わせとなる。これは我々の直感では理解できないことである。α 波は原子核から発生するミクロな世界のものであるからまだ受け入れる余裕があるが、それが猫の生死というマクロな事象に拡大すると我々の直感と矛盾したものになるというのがこの話の言いたいことである。シュレディンガー自身はこれをパラドックスとしている。このような話は量子測定理論という分野で今なお研究が続いている。

5.3.3　量子ビット

さて、長々と量子力学の話を続け、「重ね合わせ」が何かを説明したが（あまり上手に説明できた気がしないが)、量子コンピュータは「重ね合わせ」の状態をもつ「量子ビット」を構成基本単位として持つ。古典計算機では情報を 0 と 1 の数列で表し、情報の大きさを 0 と 1 が 5 個並んでいたら 5 ビット 10 個並んでいたら 10 ビットと並んでいる数字の個数で表す。この「ビット」のアナロジーとして「量子ビット」というのが考えられる。基本は「ビット」と変わらないが、重ね合わせがあるという点で違いがある。ビットは同時に 0 か 1 か一つの状態しか持つことができないが、量子ビットは 0 と 1 の状態を重ね合わせとして同時に持つことができる。またこの同時に持つことができる状態の数は量子ビット数に対し指数関数的に増加する。具体的には N 量子ビットならば 2^N 個の状態を持つことができる。

5.3.4　まとめ

さて、量子力学と量子ビットの不思議な話をして来たがここまで読んで難しい印象を持った人も多いと思う。ファインマン曰く、「量子力学がわかったと言っている奴は量子力学がわかってない」そうなので何も問題はない。量子力学は原理としている部分が摩訶不思議でややこしいので入り口のハードルは高いが枠組み自体は理路整然かつ明瞭な議論で構築されたものなので具体的な数式や方程式の話になれば難しくない（簡単だとは言ってない)。ただ量子ビットは二つの状態を同時に持つことができるということだけは押さえてほしい。

5.4　量子アニーリング

量子アニーリングというのは進化的アルゴリズムと同じように自然現象を参考にしたアルゴリズムであり、「アニーリング」を参考にした最適化問題を解くためのものである。

アニーリングを参考にしているといったが、そもそも「アニーリング」とは何であろうか。これは日本語では「焼きなまし」という。焼きなましとは金属を熱した後ゆっくり冷やすことで金属のひずみをなくし均質化することである。この時の系の状態の振る舞いを参考にしたアルゴリズムが「量子アニーリング」である。「量子」という言葉がついていることから分かるように特に量子力学で記述された焼きなましを参考にしたものである。それに対し、古典力学で記述された焼きなましを参考にした「シミュレーテッド・アニーリング」というアルゴリズムがある。これは昔から古典計算機で用いられてきたアルゴリズムである。量子アニーリングは少しややこしいのでまずはシミュレーテッド・アニーリングについて解説したい。

さて、焼きなましとは金属を高温に熱し、それをゆっくり冷やすことだがこの時の物体を構成している原子・分子に着目すると、高温時には激しく振動し、低温時にはある状態に落ち着いている。この現象がアルゴリズムとしては、高温時が様々な解を探索している時、低温時がある一つの解に決定したときに対応している。今度は山や谷がある地形の

上を運動する球体をイメージして欲しい。高温時に球体は激しく動き山があっても越えて自由自在に運動するだろう。しかし低温時には動きが小さくなり山を越えることができずある谷の底におちつく。冷やす過程をゆっくりであるほどいくつかある谷の中でも一番低い谷に最終的に落ち着くだろう。具体例を挙げよう。例えば、スマホのマップで目的地までの最短ルートを検索する時を考える。この時、いくつものルートを想定し、ルートの移動距離を「標高」、それぞれのルートを「位置」に対応させればある「地形」ができる。その地形上で焼きなましをすれば最終的に一番標高が低い位置、つまり最短距離のルートに落ち着くのである。

「量子アニーリング」は上記で述べた「シミュレーテッド・アニーリング」の量子力学版であるが、どのような違いがあるのだろうか。アルゴリズムとしての違いは「シミュレーテッド・アニーリング」は高温状態を激しく状態が変化することで表現していたが、「量子アニーリング」ではいくつもの状態の重ね合わせとして表現する。これはとても大きな違いである。というのも、シミュレーテッド・アニーリングでは山を越えようとすればその分だけのエネルギーを持った状態でなければならないが、量子アニーリングでは山を越えた先の谷にあるという状態を同時に持つことができる。このことによってある程度低温状態であっても山を越えた先の谷に移動することができるのである。このような効果を「トンネル効果」という。このトンネル効果によって効率的に問題を解くことができるのだ。

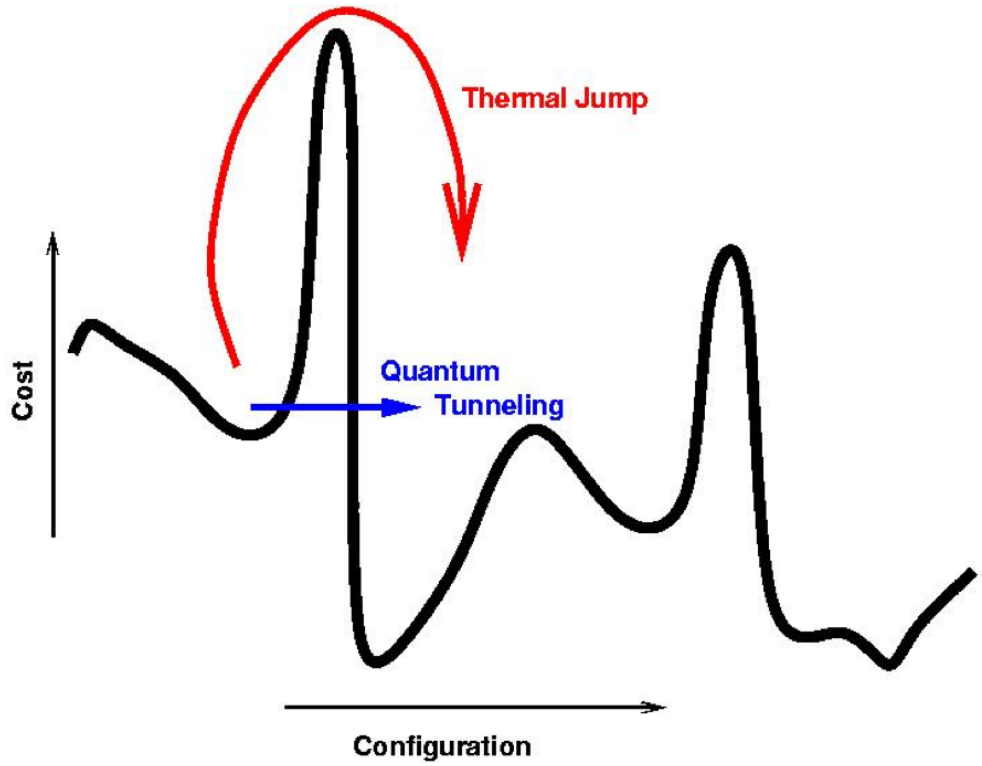


図 5.3　引用元 (https://en.wikipedia.org/wiki/Quantum_annealing)

さて、大まかな仕組みは説明したがハードでどのように量子アニーリングをするのだろうか。まず、量子ビットは小さな回路の流れる電流の向きで表現する。ニオブ製の小さなループは絶対零度付近まで冷やすと超伝導体状態になり、右回りの電流と左回りの電流が同時に存在するようになる。これが 0 と 1 の重ね合わせ状態になる。特に横磁場をかけることで二つの状態が同時に存在しやすくなる。また、上のたとえでの「地形」は量子ビット同士の相互作用で再現する。ここの相互作用というのは統計力学のイジングモデルのような相互作用である。イジングモデルは磁石のなかの原子の状態などを表す模型だが、このモデルは隣り合う原子同士の状態によってエネルギー状態がきまるという特色を

持っている。ある i 番目の量子ビットが 1 ならば 1、0 ならば -1 となる σ_i を定義すると、エネルギーは

$$E = \sum_{i,j} s_{ij}\sigma_i\sigma_j + \sum_i h_i\sigma_i$$

となる。s_{ij} は i 番目と j 番目の量子ビット同士の相互作用を表し、正ならば互いに異なる向き、負ならば同じ向きを向いたときにエネルギーが小さくなる。h_i は量子ビットの状態そのものによるエネルギー項である。系はエネルギーの低い状態をとりたがる。ここでのエネルギーが「地形」の標高に対応する。相互作用を巧みに構成することで、任意の「地形」を構成することができるのだ。相互作用ゼロ、横磁場ありの状態から始めて、横磁場をゆっくり弱めながら相互作用を強くしてやるとエネルギーが低い状態が支配的となり最終的な解が出てくる。横磁場がどうして量子揺らぎを引き起こすのかを説明するには紙幅が足りないので省略する。

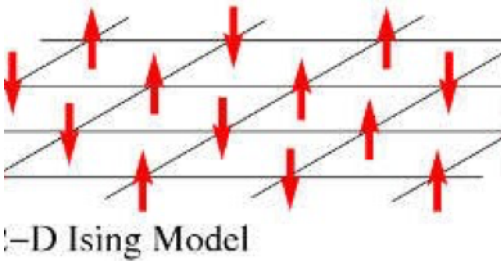


図 5.4　引用元 (https://dropseaofulaula.blogspot.com/2017/01/le-simmetrie-di-una-catena-di-ising.html)

5.5　量子アニーリングと機械学習

上記の説明だけでは量子アニーリングの仕組みがわかりにくかったと思う。ここでは具体的な問題に対してどのようにして量子アニーリングが解を求めるのかを説明する。

5.5.1　巡回セールスマン問題

まず説明するのは巡回セールスマン問題というものである。有名なので知っている人も多いと思う。これはセールスマンが複数の家を訪ねなければならない時、移動距離が最短となるような家を訪ねる順番を求めるという問題である。たとえば 5 か所の家を訪ねなければならない場合を考えよう。この時同じ家には二度訪ねないとすると、5!= 25 通りのルートがある。N か所の家の場合、$N!$ 通りなので N を大きくすると指数関数的にルートの数は増加する。この問題を愚直に解こうとすれば全ルートの移動距離を計算し、一番距離が小さいルートが解となる。しかしこの方法では N が大きくなった時に解くのに莫大な時間がかかってしまう。古典計算機でこの問題を解くとき厳密解を求めるのではなく、近似解を求めることが多い。そのアルゴリズムの一つが「シミュレーテッド・アニーリング」である。しかし、量子アニーリングでは効率的に最短経路を見つけることができる。

量子アニーリングは様々な最適化問題に利用されるが、問題ごとに量子ビットの数や相互作用の設定を変える。今回の例では 5 行 5 列に並べた 25 個の量子ビットを使う（ここでの 5 行 5 列の配置は考えやすくするためであって実際に量子ビットがそういった配置をしているわけではない）。量子ビットの列は A, B, C, D, E の 5 つの家を表しており、行は訪れたときそれが何番目に訪れた家なのかを表す。例えば、$A \to D \to C \to B \to E$ という順番で訪れたとする

と量子ビットは次の表のようになる

	A	B	C	D	E
1 番目	1	0	0	0	0
2 番目	0	0	0	1	0
3 番目	0	0	1	0	0
4 番目	0	1	0	0	0
5 番目	0	0	0	0	1

表 5.1 量子ビットの状態

さて、問題を解くための相互作用を決めよう。まず必ずどの家も一度訪ねなければならないので各列量子ビットが 1 となるものがちょうど 1 つずつなければならない。同様に各行も 1 の量子ビットがちょうど 1 つずつなければならない。詳しくは説明しないが各行の量子ビットどうしの相互作用、または各列の量子ビット同士の相互作用でこの条件を課すことができる。また「1 番目」の行の量子ビットと「2 番目」の行の量子ビット同士の相互作用の強さを家どうしの距離に応じて設定する。同様にほかの行についても相互作用を設定する。この設定の下で操作をしてやればエネルギーが一番小さい状態、つまり最短距離の状態がでる。

5.5.2 クラスタリング

量子アニーリングで巡回セールスマン問題を解く方法を述べた。同じように量子アニーリングでクラスタリングをすることができる。クラスタリングというのはデータ点の集合を二つ以上のグループに分ける教師なし学習の一種である。例えば、小学校にいる人物全員の身長と体重を測定し、グラフに結果を打ったとしよう。すると、グラフ上でデータ点の二つの塊が見えるだろう。グラフだけではわからないが一方が小学生、他方が先生のグループを表す。点の情報だけで、それぞれの点がどちらのグループに属しているのかラベリングするのがクラスタリングというアルゴリズムである。データ点を A と B という二つのグループに分ける時を考えよう。ある点 x, y のグラフ上の距離を $d(x, y)$ とし、次の関数を考える。

$$E = \sum_{x,y \in A} d(x, y) + \sum_{x,y \in B} d(x, y)$$

クラスタリングは数学的にはこの E を最小とするグループ A, B の状態を求めることと定義される。量子アニーリングでは各データ点を量子ビットで A に属するときは 0、B に属するときは 1 とする。相互作用を調整することで関数 E を構成することができ、同じように問題を解く。

5.5.3 ニューラルネットワークとボルツマンマシン

ここに来てやっと人工知能研究会らしいタイトルが出てきた（実はクラスタリングも機械学習という人工知能の一種だが)。ここではニューラルネットワークとボルツマンマシンでの量子アニーリングの活用方法について述べたい。さて、ニューラルネットワークについては本会誌のはじめのほうで解説があったと思うが、大雑把に言えばノードが層を構成し層と層の間で全結合されているモデルである。このモデル何かに似ていないだろうか。これはちょうど量子アニーリングの量子ビットの関係に似ている。相互作用がノード間の結合の重みに対応する。これが示唆するのはニューラルネットワークで学習したものを量子アニーリングに用いることの有用性である。一度学習すれば高速な動作が可能

になるだろう。

また量子アニーリングをボルツマン機械学習に用いる研究がされている。ボルツマン機械学習もニューラルネットワークを利用する人工知能であるが、入力と出力に確率的な変化を与えている。学習時には今の状態でうまくいっているかどうか調べる必要があるが、ボルツマン機械学習の場合、「サンプリング」といってとりあえずデータを大量に出してみて学習してほしいデータと比べることで評価することができる。このサンプリングというのが長時間かかり今までのネックとなっていた。量子アニーリング式の量子コンピュータである D-wave マシンがこれを解決する可能性が指摘された [Amin, 2016]。量子アニーリング自体厳密解を求めるために開発されたものであるが実際にはどうしてもノイズが発生することにより厳密解でなく近似解が導かれることがある。代わりに D-wave マシンは計算が速いという強みがある。これによりサンプリングで長時間かかってしまう心配がないのだ。必ずしも厳密解を出せなくてもサンプリング用マシンとして利用できるのである。

5.6　終わりに

以上が量子アニーリングについての解説である。量子コンピュータはフロンティアであり、何ができて何ができないかわからないことも多いので人工知能への応用の可能性もまだあまりわかっていない。しかし、少しずつ応用例が提案され、量子コンピュータの期待は高まっている。

また、量子コンピュータの副次的な効果も重要である。古典計算機の分野でも触発されて新しいアルゴリズムが開発されることもあった。また、量子コンピュータの研究による成果が理論物理学に与える影響は大きい。

量子コンピュータは物理、数学、情報が入り混じる分野である。複雑であるがそれ故に可能性は大きい。量子コンピュータという一つのアイデアによって結ばれた分野間の繋がりが未来にどんな影響を与えるか誰も計算できない。

参考文献

[1] 斉藤 康毅，ゼロから作る Deep Learning, オライリー・ジャパン

[2] 斉藤 康毅，ゼロから作る Deep Learning2, オライリー・ジャパン

[3] http://brohrer.github.io/how_convolutional_neural_networks_work.html

[4] https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[5] https://www.deeplearningbook.org/contents/convnets.html

[6] https://www.kaggle.com/cdeotte/how-to-choose-cnn-architecture-mnist

[7] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

[8] AminH.Mohammad. (2016). Quantum Boltzmann Machine. arXiv:1601.02036.

[9] 西森秀稔、大関真之. (2017). 量子コンピュータが人工知能を加速する. 港区: 日経 BP 社.

[10] 赤間世紀. (2010). 量子コンピュータがわかる本. 新宿区: 工学社.

[11] 田中宗. (2017 年 11 月 30 日). 量子アニーリングとは何か？機械学習を飛躍させる D-Wave 実装の原理　日本生まれの最先端技術がもたらす未来とは. 参照先: ビジネス＋ IT:

[12] https://www.sbbit.jp/article/cont1/34242

第 6 章

展示物紹介

6.1　歌詞生成 AI

製作者：
綿岡晃輝

Twitter：
@WataokaKoki

京都大学人工知能研究会
KaiRA
Kyoto univ. AI Research Association

歌詞生成 AI

AIに歌詞を作らせたい！その一心でこのプロジェクトは動き出しました。制作期間だいたい3ヶ月。本や記事や論文を読み漁り、開発していく日々はまさに学びの喜びそのものでした。皆さんもぜひ歌詞生成AIを動かして遊ぶだけでなく、内部の理論を理解し、その美しさに感動して見て下さい。至らぬ点も多々あると思いますが頑張って作りましたので、是非この解説記事を楽しんでお読み下さい。

※ この記事は一般向けに作成しました。機械学習を専門とする方には少し満足感の欠けるものかもしれません。

手法

1. データセットを準備する。
2. 歌詞を単語ごとに切り分ける。
3. 単語に番号を割り振る。
4. 番号を用いてAIに学習させる。
5. 学習済みAIに例文を渡し、次の単語を予想させる。
6. 5を繰り返す。

さてさて、いきなり全体像をお見せしました。機械学習に精通していらっしゃる方ならすぐに理解していただけるかと思いますが、専門外の方はどうでしょうか。早くも置いて行かれそうな気がしたでしょうか。ご心配なく、頑張って説明します。。。

1. データセットを準備する。

データセットの準備には割と時間がかかりました。何しろ欲しいものは歌詞です。著作物です。どのサイトもタダで提供してくれるはずがなく、ずいぶん四苦八苦しました。ウェブサイトからスクレイピングしました。スクレイピングというものはウェブサイトから情報を抽出する技術のことです。要するにプログラミングの力で歌詞の収集を自動化させたということです。今回は下記のサイトを参考にして、歌詞TIMEというサイトから6万曲ほどの歌詞を収集しました。

・Rubyを使って歌詞サイトから歌詞を取得する方法 (http://geektrainee.hatenablog.jp/entry/2014/07/15/105134)

2. 歌詞を単語ごとに切り分ける

英語の文章では単語ごとにスペースで区切られているのですが、日本語では違います。うまく単語ごとに切り分けることを分かち書きと言います。日本語で分かち書きをするにはMeCabというツールを使います。MeCabは京都大学情報学研究科のプロジェクトを通じて開発された無料で使えるエンジンです。これを使うことで、ただ単語を分かち書きするだけでなく、それぞれの品詞や活用形までも解析することができます。

3. 単語に番号を割り振る

歌詞をコンピュータに学習させる訳ですから、当然数字に変換する必要があります。割り振り方は至って単純。順番に読み取った単語に1, 2, 3, …と数字を割り振っていきます。この時、プログラム内できっちりと数字と単語の対応関係を記録していきます。例えば、1ならば「私」。42ならば「山」。24552ならば「走る」。というように大量に対応関係をとっていきます。下に実際のコードを記載しておきます。

ここで定義されたword2indexに単語を渡すと数字を返し、index2wordに数字を渡すと単語を返します。このようにして単語と数字の対応関係を保存しておくのです。

※ word_setという変数には読み込んだ大量の単語が重複なく格納されています。

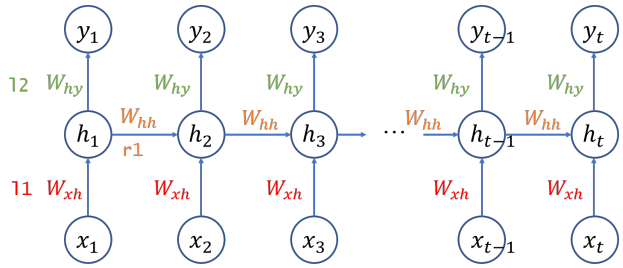
```
word2index = dict((c, i) for i, c in enumerate(word_set))

index2word = dict((i, c) for i, c in enumerate(word_set))
```

4. 番号を用いてAIに学習させる

さて、この解説記事の肝です。大量の歌詞を単語に切り分け、それらを数字にしましたが、どのようにして学習するの

でしょうか。ここではLSTM(Long Short Term Memory)というシステムを使用しました。LSTMとは下の図のような形をしています。



$$h_t = \tanh(\, l1(x_t) + r1(h_{t-1})\,)$$
$$y_t = l2(h_t)$$

難しそうですね…。ざっくりと説明しますと、このシステムは時系列データを学習するように設計されたものです。例えば、1, 2, 3, 4, 5と数字が続いた後に、どんな数字がくるでしょうか。皆さんは6だろうと簡単に正解にたどり着くことができるでしょう。しかしコンピュータにはそれは難問です。ですので、似たような問題と答えを大量に用意してあげることで、予想の精度を高めていくのです。

5. 学習済みAIに例文を渡し、次の単語を予想させる。

皆さんは気づいたでしょうか。先ほどのシステムを用いることで歌詞生成を実現させることができることに。私たちは先ほど歌詞の単語を数字に変え、それを順番に保存していきました。なので、その順番を学習させることで「君」「が」の次の単語は「好き」じゃないか。というようなそれっぽい予想ができるようになるのです。

より詳しくLSTMの理論を知りたい方は下のサイトがオススメです。

・LSTMネットワークの概要 (https://qiita.com/KojiOhki/items/89cd7b69a8a6239d67ca)

・ 今更聞けないLSTMの基本 (https://www.hellocybernetics.tech/entry/2017/05/06/182757)

・わかるLSTM ～ 最近の動向と共に (https://qiita.com/t_Signull/items/21b82be280b46f467d1b)

6. 5を繰り返す。

予想した単語を文章の続きとして見なし、またその続きを予想させることで歌詞生成を実現することができるのです。今回は全ての歌詞を学習させず、特定の歌手を学習させることにしました。(その方が世界観が統一され、歌詞っぽくなったからです。**筆者が様々な歌手を試し、出力された歌手を見ていきましょう。

結果

ひとつを今夜に川の

コップに笑い負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負け

負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた

負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた

負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた

負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた

負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた

自分た負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた

負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた

負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた

負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた

負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負けた負け

選択歌手：長渕剛

学習1回目

学習3回目

ままから路地雨の中も
俺は俺の鹿児島は
今としたものを
人で行くのにとしたよ
ものは僕は数えそう
俺の名は今お前を
俺は俺はもこんなととしたよ
人でもないで俺とも
人をしてた俺らも
私は俺の街を
朝にくると人に
負けたよか人でも
俺は俺は今と人は
今お前をしてた俺は
今あなたの前によ
俺は俺が今俺らは
今今俺が今だと人を
俺は俺は今お前と今たよ
人が俺がある俺は

学習9回目

少女たくし
お前はこと人ためよ
好きな好きもね
好きてくれてもない俺らの
好き俺は最後だよ
あなたよ機嫌をまでまで
終業はあんたの
お前は抱きしめて行くわけを
好きだと思ったね
好き好き厚化粧だだ「するするに
たくたく好きにだだから
TouchTakeyourBACKで言わonyour海が抱き海が向こうか
俺はあんただとですを
風に何をてて
何何何に何も
度出がみよ「、優しでに
許してくれくれたお前は
好きに好きだだから
好きてきだよぜ

学習を重ねることで少しずつ文章っぽくなっていきます。

選択歌手：福山雅治

学習回数10回目

風になるの日曜日君は
ポケットのもの君のものだよ
素直になれない…そう
君と見たいいつもずっとさ
君のすべてみつめていつも愛に
希望のこの胸の中できっときっと夢は
前になれ言葉の明日の
僕は見ているの
ものを君のように
俺道を太陽たらふたりの

これが最も出来が良かった歌詞です。

選択歌手：MR.CHILDREN

学習回数10回目

光まで愛してる僕は
君の僕はどう君の
たい君に会いたい愛して
君にどうたい君の
たい愛していた君に
たい君に会いたい愛して
君に会いたい愛して
君にどう君に会いたい愛して
君に会いたい君に
たい愛しています君は
君に会いたい愛して
君に会いたい愛して
君にどうたい君の
たい君に会いたい愛して
君にどう君に会いたい君に
たい愛しています君がどう
君に会いたい君に
たい愛しています君がどう
君に会いたい君に
たい愛しています君に

君会いたいしか言わなくなったのでミスチルは諦めました。

選択歌手：福山雅治+長渕剛

学習回数10回目

俺を俺は俺は

今までも行くそうそう

青春ね欲しい明日が笑ってねえ俺は

俺らの中になり生きて

二を待っになれの

お前がこないでを

事にけれどと破片で俺らが

お前はだけの中だけ

Tomorrow明日明日と信じと信じて

お前が時と思えん

愛を待ってたん

心をずっとその笑っを

涙の心になりて

頭ねお前のんだ笑って笑ってNeverそう

今は今あたいのいつもそう

真っ直ぐ真っ直ぐ真っ直ぐ真っ直ぐ真っ直ぐ生きて

お前が好きだ嫌いと抱きしめそう

君ととひとり手の

女の俺は今でしょでしょ愛の

!!KNOCK!KNOCK!

このように複数歌手の足し算できます。どことなく福山雅治の甘い歌詞と長渕剛の男らしい歌詞が混在している気がしませんか？

終わりに

AIを応用させることでこんなにも面白い仕組みを作ることができます。そして、AIの面白さはまだまだあります。この解説記事をお読みになられた方が少しでもAIに興味を持っていただけたなら幸いです。

6.2 似ている芸能人判定 AI

VGGFace2 というデータセットで訓練したニューラルネットを使っています．VGGFace2 は 9131 人の 331 万枚の顔写真が含まれています．このデータセットを使い，ニューラルネットに顔写真を 9131 クラスに分類する問題を解かせます．今回使っているニューラルネットは ResNet50 というアーキテクチャにサイズ 128 と サイズ 9131 の全結合層をつけたものです．顔写真で訓練されたこのニューラルネットのサイズ 128 の全結合層の出力は，人間の顔の特徴をうまく表現できる 128 次元のベクトルだと考えることができます．

この自分の顔に似ている有名人を教えてくれるプログラムは，予め用意されている様々な有名人の顔写真と自分の顔写真を入力とし，VGGFace2 データセットで訓練したニューラルネットで 128 次元の特徴ベクトルを計算します．それから，自分の顔の特徴ベクトルと各有名人の顔の特徴ベクトルとのなす角の余弦 (cos) を計算します．2 つの顔の特徴ベクトルのなす角が小さいほど顔が似ていることになります．数式で書くと，求めたい有名人 i は

$$\arg\max_i \frac{f(\boldsymbol{x}_{self}) \cdot f(\boldsymbol{x}_i)}{|f(\boldsymbol{x}_{self})|\ |f(\boldsymbol{x}_i)|}$$

$\boldsymbol{x}_{self}$ は自分の顔写真，$\boldsymbol{x}_i$ は i 番目の有名人の顔写真，$f(\boldsymbol{x})$ は顔写真 $\boldsymbol{x}$ を受け取ったときのニューラルネットの出力を表します．

サークル情報

京都大学人工知能研究会 KaiRA

代表　金子 英樹 (農学部 4 回)
活動日　毎週木曜日 18:30〜
場所　京大病院内
会費　無料
入会資格　誰でも
Mail　kyoto.kaira@gmail.com
Website　https://kyoto-kaira.github.io/contact.html
Twitter　@kyoto_kaira

活動内容

1. 本の輪読
   深層学習や機械学習に関する専門書籍の輪読を行います。各章に担当者を設定し、プレゼンやハンズオンなど様々な形態で相互の理解を深めます。
2. ソフトウェア開発
   学んだことを活かして各自が興味のあるプロジェクトに取り組みます。成果は勉強会や学園祭で発表します。

文責

はじめに　金子 英樹
1 章 Deep Learning 入門　大山 百々勢，新里 顕大，橋本 竜馬，廣瀬 雄一
2 章 コンピュータビジョン　大山 百々勢
3 章 自然言語処理　廣瀬 雄一
4 章 深層学習と自動会話文生成　金子 英樹
5 章 量子コンピュータ　田中 大登
6 章 展示物紹介　綿岡 晃輝，DU YICHENG
冊子編集　新里 顕大