パーセプトロンの原理

すうがくぶんか

　　　　　　　　　　　　　　　【目次】

[１『Python機械学習プログラミング』学び方ガイド 3](#_Toc466794812)

[『Python機械学習プログラミング』とは 3](#_Toc466794813)

[本書の前提知識 3](#_Toc466794814)

[Pythonの基礎的な文法 3](#_Toc466794815)

[データ解析ライブラリの基本的な使用方法 3](#_Toc466794816)

[数学の基礎 4](#_Toc466794817)

[本書の構成 5](#_Toc466794818)

[本書の読み方のプラン 8](#_Toc466794819)

[●プラン1. 速習コース 8](#_Toc466794820)

[●プラン2. 特訓コース 9](#_Toc466794821)

[本書を読むための環境構築 9](#_Toc466794822)

[本書を読了した後の参考文献 10](#_Toc466794823)

[２　パーセプトロンを読み解く　その１ 11](#_Toc466794824)

[第2章の内容 11](#_Toc466794825)

[パーセプトロンの定式化 11](#_Toc466794826)

[行列による総入力の表現 14](#_Toc466794827)

[重みを推定するための計算方法 15](#_Toc466794828)

[３　パーセプトロンを読み解く　その２ 15](#_Toc466794829)

[具体例で数式を読み解く 16](#_Toc466794830)

[４　パーセプトロンを読み解く　その３ 22](#_Toc466794831)

[５　単純な例でADALINEのイメージをつかむ　その１ 25](#_Toc466794832)

[ADALINE 25](#_Toc466794833)

[誤差平方和：真の値と予測値の差を2乗して足し合わせる 26](#_Toc466794834)

[６　単純な例でADALINEのイメージをつかむ　その２ 28](#_Toc466794835)

[勾配降下法: 誤差平方和の勾配と逆方向に重みをずらしていく 28](#_Toc466794836)

[７　単純な例でADALINEのイメージをつかむ　その３ 31](#_Toc466794837)

[ADALINEを実装する 31](#_Toc466794838)

# １『Python機械学習プログラミング』学び方ガイド

『Python Machine Learning』が2015年9月に米国などで発売。「機械学習の考え方」と「Pythonプログラミングによる実践」をバランスよく解説していると評価され、米国Amazon.comでベストセラー。その日本語訳はコラムや脚注、付録が追加され、2016年6月に発売されました。ここでは、本書のより効果的な活用法について監訳者が解説します。

『Python機械学習プログラミング』とは

書籍[『Python機械学習プログラミング』](http://book.impress.co.jp/books/1115101122)は、Sebastian Raschka氏によるPythonを用いた機械学習の入門書"Python Machine Learning"の翻訳書です。著者はミシガン州立大学の博士課程で生物統計学(計算生物学)の研究を行っており、scikit-learnなど多数の機械学習やデータサイエンスのライブラリの開発にも携わっています。

本書の特徴は、Pythonによる実装、ライブラリの使用方法、機械学習の考え方の説明をバランスよく織り交ぜ、機械学習の登山口から実務で使用できるレベルまで連れて行ってくれる点にあります。以降では、本書を通してより効果的に機械学習を学べるように、必要な前提知識、本書の構成と読み方のプラン等について説明します。

なお、著者はデータ解析に用いる言語についてエッセイを執筆しており、言語間に優劣はないとしながらも著者にとってはPythonがベストだったと説明しています。読み物として面白いので、お時間のあるときに一読されるとよいのではないかと思います。

■著者によるエッセイ  
◎Python, Machine Learning, and Language Wars. A Highly Subjective Point of View  
<http://sebastianraschka.com/blog/2015/why-python.html>

◎和訳版: Pythonや機械学習、そして言語の競争について – 極めて主観的な見地から  
<http://postd.cc/python-machine-learning-and-language-wars-a-highly-subjective-point-of-view/>

本書の前提知識

本書を読むのに必要な基礎知識は、Pythonの基礎的な文法、データ解析ライブラリの基本的な使用方法、数学の基礎（微積分、線形代数）の3つです。

Pythonの基礎的な文法

Pythonの文法については、リスト、タプル、ディクショナリなどの基本的なデータ構造、forループ、print関数、zip関数、enumerate関数、関数やクラスの作成方法などが理解できていれば十分です。例えば以下の書籍の該当箇所を読んでおくとよいでしょう。

■『[みんなのPython 第3版](http://www.amazon.co.jp/gp/product/4797371595/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4797371595&linkCode=as2&tag=ipc-thinkit-22)』(ソフトバンククリエイティブ、2012年)

データ解析ライブラリの基本的な使用方法

本書の複数箇所で使用されているPythonのデータ解析ライブラリは、主にNumPy/SciPy、pandas、matplotlib、seaborn、scikit-learnです。これらをタスクで大別すると表1のようになります。

表1：本書で使用するライブラリとそのタスク  
[](https://thinkit.co.jp/sites/default/files/article_node/PML_Hyo01.png)

これらのライブラリのうち、SciPy、pandas、seabornは使用する機能が多くなく、該当箇所で丁寧に説明されています。そのため、前提知識はほとんど必要ないと思います。また、scikit-learnも丁寧な説明があるので、予備知識は特に不要でしょう。

残りのNumPy、matplotlibについては多少の知識が必要です。matplotlibの基本的な使用方法は付録Bに追加していますので、必要があれば参照してください。NumPyについては、知識があまりなくても通読できるようになっていますが、以下の資料等を読んであらかじめ知識をつけておくと理解が深まるでしょう。

◎朱鷺の杜Wiki「Numpyの概要」  
<http://ibisforest.org/index.php?python%2Fnumpy#q38168ab>

◎機械学習のPythonとの出会い  
<http://www.kamishima.net/mlmpyja/>

なお、NumPy/SciPy、matplotlib、scikit-learnの導入的な説明は以下の書籍がまとまっています。一読しておくと本書の理解が深まると思います。

■『[データサイエンティスト養成読本 機械学習入門編 (Software Design plus)](http://www.amazon.co.jp/gp/product/4774176311/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4774176311&linkCode=as2&tag=ipc-thinkit-22)』(技術評論社、2015年)  
第II部 特集2 Pythonによる機械学習入門

この書籍は機械学習の入門に適しているので、他の章も合わせてお読みください。  
以下の書籍は、NumPy/SciPy、pandas、matplotlibについて詳しく説明しています。

■『[Pythonによるデータ分析入門 ―NumPy、pandasを使ったデータ処理](http://www.amazon.co.jp/gp/product/4873116554/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4873116554&linkCode=as2&tag=ipc-thinkit-22)』(オライリー・ジャパン、2013年)

数学の基礎

数学の基礎知識として、微積分と線形代数が必要になります。

微積分については偏微分や連鎖律、ヤコビ行列等の知識が必要になります。以下の書籍等で学習するとよいと思います。

■『[キーポイント微分積分 (理工系数学のキーポイント 1))](http://www.amazon.co.jp/gp/product/4000078615/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4000078615&linkCode=as2&tag=ipc-thinkit-22)』(岩波書店、1996年)

■『[キーポイント多変数の微分積分 (理工系数学のキーポイント (7))](http://www.amazon.co.jp/gp/product/4000078674/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4000078674&linkCode=as2&tag=ipc-thinkit-22)』(岩波書店、1996年)

線形代数については、行列の積、逆行列、固有値と固有ベクトル(固有分解または固有値分解)などの知識が必要になります。本書内でも挙げましたが、以下の書籍がまとまっていてわかりやすいので一読をお勧めします。

■『[プログラミングのための線形代数](http://www.amazon.co.jp/gp/product/4274065782/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4274065782&linkCode=as2&tag=ipc-thinkit-22)』（オーム社、2004年）

本書を読み進めるうえでは、以下の箇所を読めば十分です。

第0章 動機  
第1章 ベクトル・行列・行列式  
第2章 ランク・逆行列・一次方程式  
第4章 固有値・対角化・Jordan標準形  
「4.5 固有値・固有ベクトル」まで

また、行列の積、固有値と固有ベクトル(固有分解)については導入的な内容になりますが、それぞれ本書の2章、付録Cに説明があるので適宜参照して下さい。

## 本書の構成

[本書](http://book.impress.co.jp/books/1115101122)の章立ては以下のようになっています。

* 第1章 「データから学習する能力」をコンピュータに与える
* 第2章 分類問題 -- 機械学習アルゴリズムのトレーニング
* 第3章 分類問題 -- 機械学習ライブラリscikit-learnの活用
* 第4章 データ前処理 -- よりよいトレーニングセットの構築
* 第5章 次元削減でデータを圧縮する
* 第6章 モデル評価とハイパーパラメータのチューニングのベストプラクティス
* 第7章 アンサンブル学習 -- 異なるモデルの組み合わせ
* 第8章 機械学習の適用1 -- 感情分析
* 第9章 機械学習の適用2 -- Webアプリケーション
* 第10章 回帰分析 -- 連続値をとる目的変数の予測
* 第11章 クラスタ分析 -- ラベルなしデータの分析
* 第12章 ニューラルネットワーク -- 画像認識トレーニング
* 第13章 ニューラルネットワーク -- 数値計算ライブラリTheanoによるトレーニングの並列化

第1章は導入にあたり、前半では機械学習が「教師あり学習」「教師なし学習」「強化学習」の3種類に大別されること、そしてそれぞれの概要が説明されています。本書では、「教師あり学習」と「教師なし学習」について扱われます。また、本章の後半では使用する数学的な記法、PythonやAnacondaの導入等について説明されています。

2章以降の章をカテゴライズしてそれぞれの章の依存関係を図1に示します。

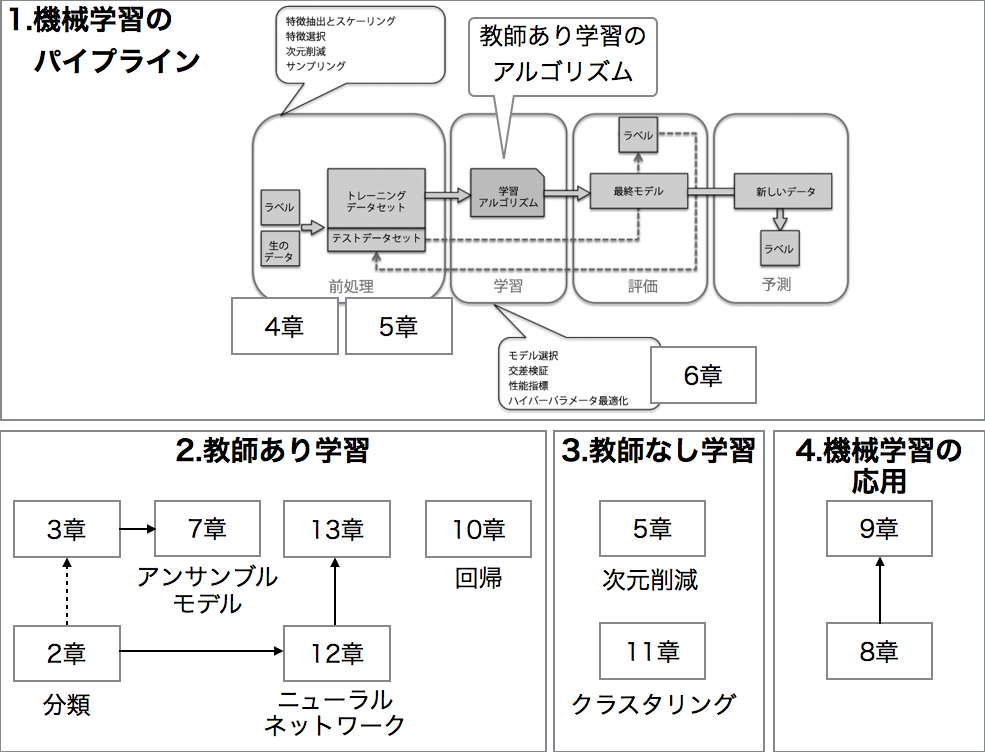
図1：本書の各章の関係  
[](https://thinkit.co.jp/sites/default/files/article_node/PML_structure_v2.png)

図1では、本書の内容を以下の4つの観点でカテゴライズしています。なお、5章については、機械学習のパイプラインと教師なし学習のアルゴリズムの双方に分類しています。

1. 機械学習のパイプライン(4章、5章、6章、および「2. 教師なし学習のアルゴリズム」の該当章)
2. 教師あり学習のアルゴリズム(2章、3章、7章、10章、12章、13章)
3. 教師なし学習のアルゴリズム(5章、11章)
4. 機械学習の応用(8章、9章)

以上では、機械学習のアルゴリズムを「教師あり学習」と「教師なし学習」で分類しました。本書では、「教師あり学習」として「分類」と「回帰」の2つのタスク、「教師なし学習」として「次元削減」と「クラスタリング」の2つのタスクを扱っています。

1. 機械学習のパイプラインに該当する章では、分類や回帰により教師あり学習のモデルを構築し評価するときに必要な処理を説明しています。これは、実務で機械学習を適用する際に必要となる実践的な内容です。4章ではデータの前処理、5章では次元削減、6章ではハイパーパラメータのチューニングや予測の評価等について説明されています。これらの章は順番に読んでいくとよいでしょう。また、機械学習のパイプラインで現れる「学習アルゴリズム」を知るには、「2. 教師あり学習のアルゴリズム」の該当する章を参照する必要があります。なお、5章の次元削減で扱われる主成分分析や判別分析の式の詳細な導出を知りたい場合は、以下の書籍等を参照されることをお勧めします。

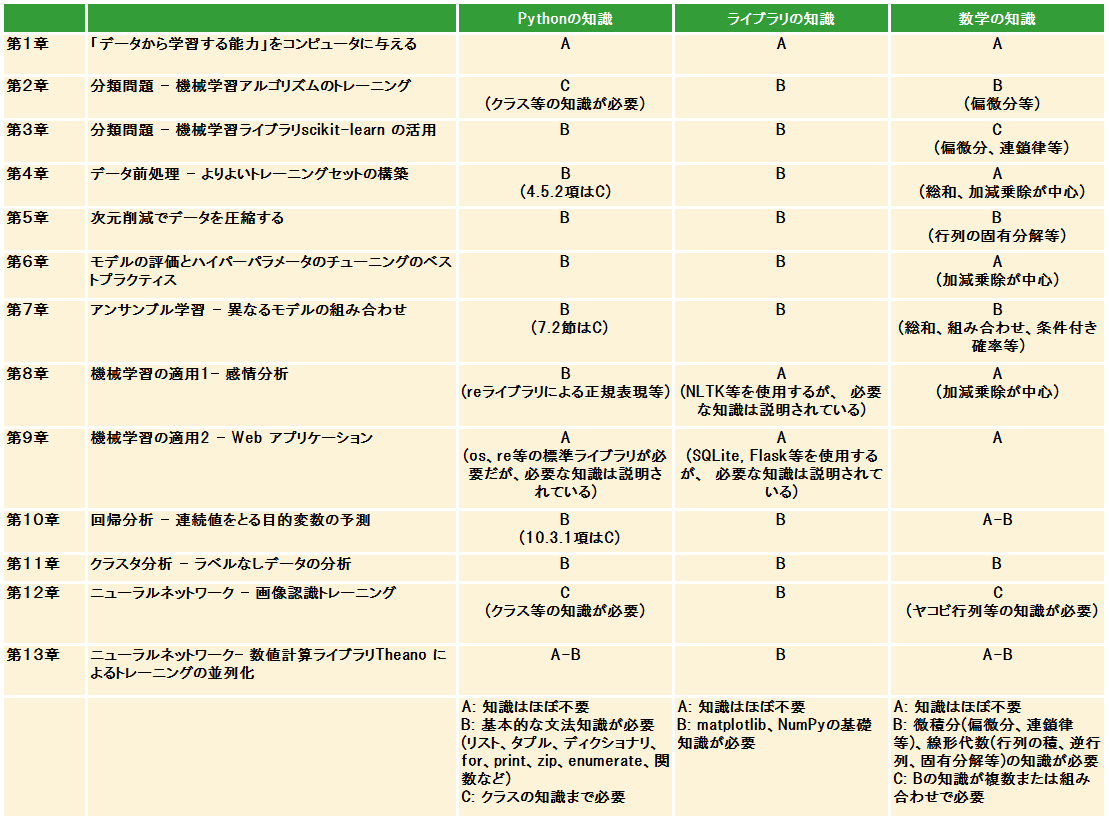
■『[わかりやすいパターン認識](http://www.amazon.co.jp/gp/product/4274131491/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4274131491&linkCode=as2&tag=ipc-thinkit-22)』（オーム社、1998年）

2. 教師あり学習のアルゴリズムに該当する章では、分類と回帰のアルゴリズムが説明されています。分類は2章、3章、7章、12章、13章、回帰は10章で説明されています。分類については、7章を読むには3章、12章を読むには2章、13章を読むには12章をそれぞれ読んでおくと理解が深まると思います。

3. 教師なし学習のアルゴリズムは、5章で次元削減、11章でクラスタリングが説明されています。これら2つの章はそれぞれ独立に読むことが可能です。

4. 機械学習の応用は、8章で映画のレビュー文に対して肯定意見か否定意見かを分類するモデルを構築し、9章でWebアプリケーションを作成しています。これら2つの章はセットのため、順番に読む必要があります。なお、8章はテキストデータの解析(自然言語処理、テキストマイニング)の一例となっているため、この分野に関心のある場合は目を通したほうがよいでしょう。

各章で必要とされる「Python」「ライブラリ」「数学」のそれぞれの知識を評価すると表2のようになります。

表2：各章のおおまかな難易度  
[](https://thinkit.co.jp/sites/default/files/article_node/PML_chap_difficulty02.png)

特に注意が必要なのは、2章は比較的早いうちにクラスを用いてパーセプトロンやADALINEというアルゴリズムの実装が行われることです。そのため、Pythonの知識に不安がある場合は2章を後回しにして3章でscikit-learnライブラリを使用しながら、分類のアルゴリズムに慣れていくのも一つの手です。この後、「本書の読み方のプラン」で詳しく説明します。なお、5月下旬に刊行された以下の書籍には、2章で扱われる題材が歴史的な経緯も含めて説明されています。読み物として面白いので、興味のある方は副読本として一読されることをお勧めします。

■『[脳・心・人工知能 数理で脳を解き明かす (ブルーバックス)](http://www.amazon.co.jp/gp/product/4062579685/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4062579685&linkCode=as2&tag=ipc-thinkit-22)』(講談社、2016年)

## 本書の読み方のプラン

以上で説明した[本書](http://book.impress.co.jp/books/1115101122)の構成や章ごとの難易度をもとに、本書の読み方についてここでは2つのプランを提示します。

●プラン1. 速習コース：最初は関数やクラスを定義してアルゴリズムを記述したり、数式を丁寧に追ったりすることよりも、ライブラリの使用方法を覚え、一通り機械学習を使いこなすことを優先します。Pythonのクラスや数学の知識があまりない場合は、このプランに従って本書を読み進めるとよいと思います。

●プラン2. 特訓コース：関数やクラスを用いたアルゴリズムの記述、数式に抵抗感がない場合はこのプランに従って読み進めるとよいと思います。

以下ではそれぞれのプランについて説明します。

### ●プラン1. 速習コース

このプランでは、図2に示す順に読み進めていきます。

図2：速習コースのフロー  
[](https://thinkit.co.jp/sites/default/files/article_node/PML_plan1_flow.png)

ここでは、まずはscikit-learnを用いて教師あり学習の分類モデル、その構築、検証方法について一通り学習していきます。3章から7章で、Pythonのクラスや微積分、線形代数の基礎知識が要求される以下の箇所は、初読の際は飛ばしてもよいかもしれません。

◆Pythonのクラスなどにあまりなじみがない場合

4.5.2項「逐次特徴選択アルゴリズム」

5.3.2項「Pythonでカーネル主成分分析を実装する」、5.3.3項「新しいデータ点を射影する」

7.2節「単純な多数決分類器の実装」、7.3節「アンサンブル分類器の評価とチューニング」

◆数学にあまりなじみがない場合

3章の数式による説明

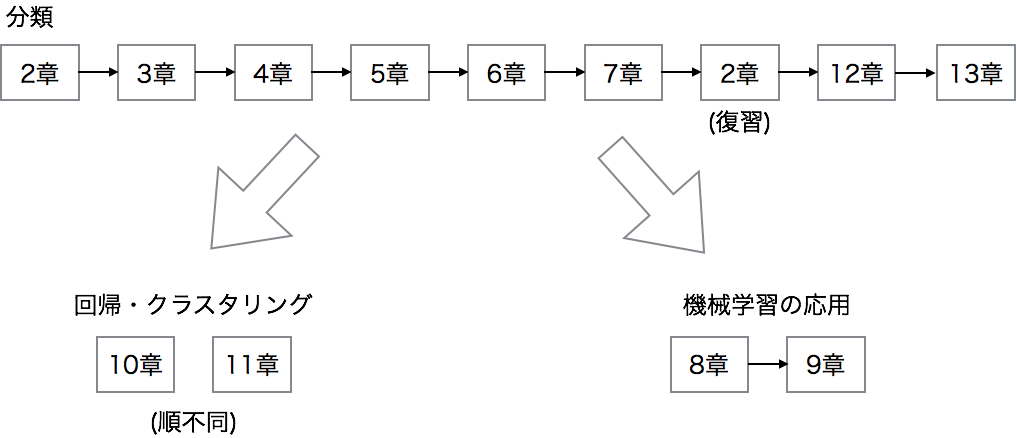
5章の数式による説明

まず3章でscikit-learnを用いて分類の主要なアルゴリズムの概要を理解し、続いて4章でデータの前処理、5章で次元削減、6章でハイパーパラメータのチューニング、モデルの検証等の機械学習のパイプラインについて学習していきます。7章のアンサンブル学習では3.6節で学習する決定木の知識が必要になります。

その後、12章、13章のニューラルネットワーク(多層パーセプトロン)に移行します。これらの章は、最近流行している深層学習(ディープラーニング)を他書で本格的に学ぶための導入的な役割を果たしています。12章を読むにはクラスなどのPythonの基礎知識、数学(微積分、線形代数)を避けては通れません。そのため、先に挙げた参考文献で補強しながら12章の基礎となっている2章を読み、12章に挑むのがよいと思います。

### ●プラン2. 特訓コース

このプランでは、図3に示す順番で読み進めていきます。

図3：特訓コースのフロー  
[](https://thinkit.co.jp/sites/default/files/article_node/PML_plan2_flow.png)

このプランでは、まずは分類について一通り学習していきます。プラン1の速習コースでは2章を飛ばしましたが、ここでは2章から学んでいきます。

分類をひととおり学んだ後は、回帰、クラスタリング、機械学習の応用を興味に合わせて読み進めるとよいでしょう。

## 本書を読むための環境構築

本書は、実際にコードを実行して結果を確かめながら読み進めていくタイプの書籍です。使用する主要なライブラリを一括で揃えた環境を構築するためには、1.9.1項「Pythonパッケージのインストール」で説明されているようにAnacondaを使用するとよいでしょう。

Windowsを使用している場合は、以下のページからインストーラをダウンロードしインストールします。Python 2系と3系の両方を提供していますが、本書はPython 3系の使用を想定しているので、特に理由がなければ3系を選択するとよいでしょう。

◎Anacondaインストーラのダウンロード先  
<https://www.continuum.io/downloads>

Mac OSやLinuxを使用している場合はpyenvを経由してAnacondaをインストールするとよいでしょう。以下のQiitaの記事に情報がまとまっています。

◎データサイエンティストを目指す人のpython環境構築 2016  
<http://qiita.com/y__sama/items/5b62d31cb7e6ed50f02c>

また、著者はGitHubでソースコードや数式の補足等の資料を提供していますので、必要に応じて使用しましょう。

◎著者による本書のGitHubサイト  
<https://github.com/rasbt/python-machine-learning-book>

これらのソースコードはipynb形式で提供されており、Jupyter Notebookを使用することにより実行できます。Jupyter Notebookを使用すると、ブラウザ上でソースコードを記述、実行し結果を確認する試行錯誤の過程を記録することができます。また、ドキュメント、数式、画像なども含めることができます。なお、Anacondaは、Jupyter Notebookも提供しているのでインストールの作業は不要です。

Jupyter Notebookについては、本書の付録Aに、上記のソースコードを読み込む方法も含めて基本的な使用方法を説明しました。なお、監訳にあたってはJupyter Notebook上でソースコードを打ち込みながら確認を行いました。

## 本書を読了した後の参考文献

最後に、本書を読了した後の参考文献をいくつか挙げます。

■『[実践 機械学習システム](http://www.amazon.co.jp/gp/product/4873116988/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4873116988&linkCode=as2&tag=ipc-thinkit-22)』（オライリー・ジャパン、2014年）：scikit-learnを用いた豊富な実践例が紹介されており、本書で機械学習の基礎を身につけた後に読むのに適しています。

■『[Data Science from Scratch](http://www.amazon.co.jp/gp/product/149190142X/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=149190142X&linkCode=as2&tag=ipc-thinkit-22)』（Oreilly & Associates Inc, 2015年）：本書でもscikit-learnのクラス、関数を使うのではなく、自力でクラスや関数を実装しているケースがいくつかあります。この書籍では、scikit-learnなどを使用せずに、自力で機械学習のアルゴリズムを実装しています。使用されているソースコードがGitHubで公開されているので、参照するのもよいでしょう

（<https://github.com/joelgrus/data-science-from-scratch>）。

■『[言語処理のための機械学習入門 (自然言語処理シリーズ)](http://www.amazon.co.jp/gp/product/4339027510/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4339027510&linkCode=as2&tag=ipc-thinkit-22)』（コロナ社、2010年）：「言語処理のための」と書名にはありますが、数学の基礎知識から始まり、数式を交えて機械学習の基礎を学ぶのに適しています。

■『[パターン認識と機械学習 上](http://www.amazon.co.jp/gp/product/4621061224/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4621061224&linkCode=as2&tag=ipc-thinkit-22)』（丸善出版、2012年）：機械学習の理論面を勉強する場合に是非読みたい一冊です。

以下の3冊は、本書の12章、13章の次に深層学習の理解を深めるために読みたい書籍です。ツール、ライブラリの使用方法も交えて学習したい場合は最後の「イラストで学ぶディープラーニング」を読むとよいでしょう。

■『[深層学習 (機械学習プロフェッショナルシリーズ)](http://www.amazon.co.jp/gp/product/4061529021/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=4061529021&linkCode=as2&tag=ipc-thinkit-22)』（講談社、2015年）

■『[深層学習 Deep Learning (監修:人工知能学会)](http://www.amazon.co.jp/gp/product/476490487X/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=476490487X&linkCode=as2&tag=ipc-thinkit-22)』（近代科学社、2015年）

■『[イラストで学ぶ ディープラーニング (KS情報科学専門書)](http://www.amazon.co.jp/gp/product/406153825X/ref=as_li_ss_tl?ie=UTF8&camp=247&creative=7399&creativeASIN=406153825X&linkCode=as2&tag=ipc-thinkit-22)』（講談社、2016年）

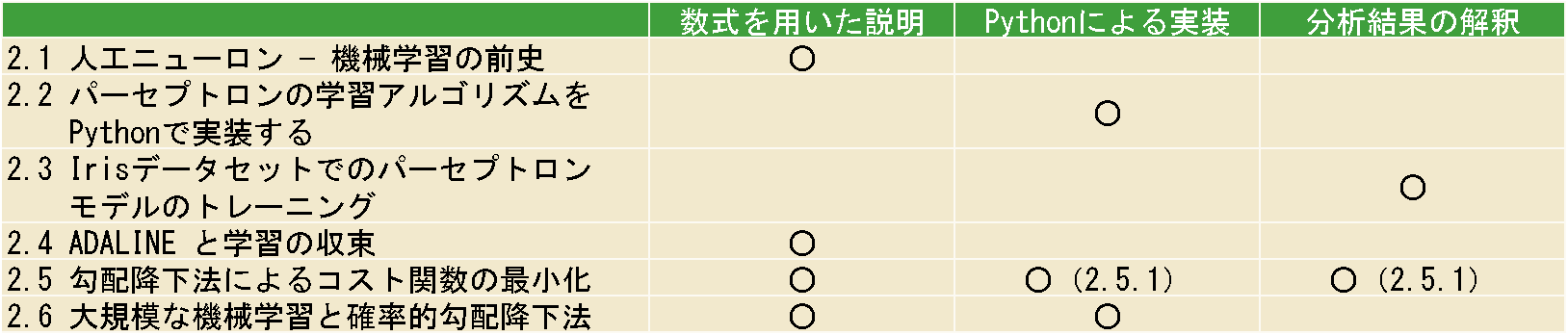
# ２　パーセプトロンを読み解く　その１

原著の『Python Machine Learning』は2015年9月に米国などで発売。「機械学習の考え方」と「Pythonプログラミングによる実践」をバランスよく解説していると評価され、米国Amazon.comでベストセラー。その日本語訳はコラムや脚注、付録が追加され、2016年6月に発売されました。ここでは、本書のより効果的な活用法について監訳者が解説します。

第2章の内容

『Python機械学習プログラミング』の第2章では、パーセプトロン、ADALINE という2 つのアルゴリズムについて説明されます。本章の内容を理解することにより、機械学習の分類タスクの考え方、Python による機械学習アルゴリズム実装への習熟、最適化の考え方等について理解を深めることができます。

各節で扱われる内容を「数式を用いた説明」「Pythonによる実装」「分析結果の解釈」の3つによって大まかに分類すると、表1のようになります。

表1：各節で扱われる内容について特性で分類  
[](https://thinkit.co.jp/sites/default/files/article_node/Hyo01.png)

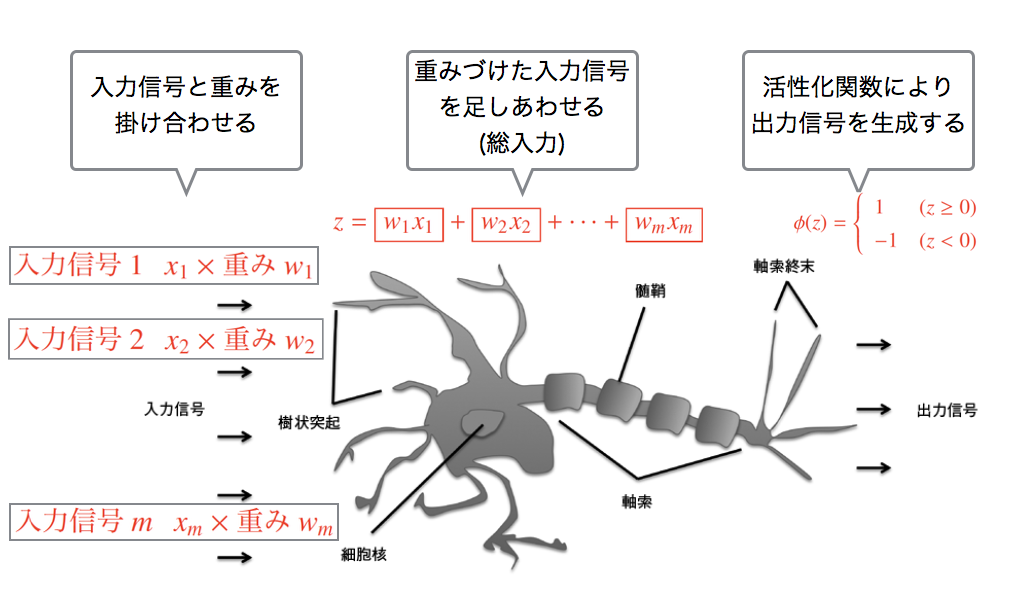
「Pythonによる実装」は、第3章以降でscikit-learnを用いて機械学習のアルゴリズムを実行するときの基礎になります。「数式による説明」は、第3章でロジスティック回帰やサポートベクトルマシンなどのアルゴリズム、また第12章でニューラルネットワークを理解するのに重要な事項です。

以上のように、第2章は本書を読み進めていく上で重要な位置付けになっています。一方で、前回記述したように、数式が最初から登場したり、クラスを用いてパーセプトロンやADALINEのアルゴリズムを実装したりと、少し取っ付きづらい面もあるかもしれません。本記事では、そうした点を補っていければと思います。

パーセプトロンの定式化

2.1節「人工ニューロン--機械学習の前史」から2.3節「Irisデータセットでのパーセプトロンモデルのトレーニング」では、数式を交えたパーセプトロンの概要、Pythonによる実装、そしてその分析結果が説明されます。

p.18の図において、入力信号が*m*個あるとします。以下のように図の左側には5個の矢印が描かれていますが、この1つ1つが入力信号だと考えてください。

図1：神経細胞の模式図  
[](https://thinkit.co.jp/sites/default/files/10342/neuron_with_eq.png)

入力信号1の信号値がx1、入力信号2の信号値がx2、・・・、入力信号*m*の信号値がxmであるとします。これらの入力信号は重みをw1、・・・、wmとして、それぞれを足し合わせたものが細胞核に取り込まれます。これを総入力(net input)と呼んでいます。

総入力z = (入力信号1 の重み付きの信号値) + (入力信号2 の重み付きの信号値) + … + (入力信号m の重み付きの信号値)

　　　 = w1x1 + w2x2 + … + wmxm

例えば、3 つの入力信号があるとして、1 番目の入力信号の信号値をx1 = 1、2 番目の入力信号の信号値をx2 = 2、3 番目の入力信号の信号値をx3 = 3 とします。これらの入力信号の重みをw1 = 0.2、w2 = 0.4、w3 = 0.5 とします。このとき、

総入力z = (入力信号1 の重み付きの信号値) + (入力信号2 の重み付きの信号値) + … + (入力信号m の重み付きの信号値)

　　　 = w1x1 + w2x2 + w3x3

　　　 = 0.2 × 1 + 0.4 × 2 + 0.5 × 3 = 0.2 + 0.8 + 1.5 = 2.5

となります。例えば、すべての入力信号の重みが1 のときの総入力は、

z = 1・x1 + 1・x2 +・・・+ 1・xm

= x1 + x2 +・・・+ xm

となり、単純にすべての入力信号の信号値を足し合わせたもの(和) になります。さて、本書内でも説明されているように、Rosenblatt によるパーセプトロンは以上の神経細胞を模したアルゴリズムで、入力信号から出力信号を判定するものです。そのアイディアは、

**入力信号の重み付きの和が一定値(しきい値) 以上であれば出力信号が生成される**

というものです。これは、総入力z がしきい値θ を超えているかどうかを判定することにより行います。数式で表すと、次の判定条件になります。

総入力z ≥ しきい値θ

総入力z の式を具体的に代入すると、以下のようになります。

z = w1x1 + w2x2 + … + wmxm ≥ θ

先ほどの具体例では、以下のようになります。

w1x1 + w2x2 + w3x3 = 2.5 ≥ θ

例えば、θ = 2 とすると左辺はθ より大きくなります。θ = 3 とすると左辺はθ よりも小さくなります。

上記の総入力z がしきい値θ を超えるかどうかを判定する式において、右辺のθ を左辺に移行すると、以下の式を得ます。

-θ + w1x1 + w2x2 + ・・・ + wmxm ≥ 0

この式で、

-θ = w0x0

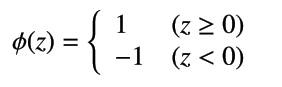
とおくと、

w0x0 + w1x1 + w2x2 + w3x3 + ・・・ + wmxm ≥ 0

となり、統一的な形で表すことができます。w0とx0の決め方は様々なものが考えられます。例えば、w0 = -1、x0 = θ としても良いですし、w0 = θ、x0 = -1 としても良さそうです。本書では、重みをw0 = -θ、x0 = 1 としています。このとき、総入力z は次式

z = w0x0 + w1x1 + ・・・ + wmxm

で表せます。総入力z が0 以上ならば出力信号が生成され、0 未満ならば出力信号が生成されません。このように、総入力z の値から出力の値を判断する活性化関数φ = φ(z) は次式で定義されます。

[](https://thinkit.co.jp/sites/default/files/article_node/p03_001.jpg%20target=)

例えば、入力信号の重みをw0 = 0.2、w1 = 0.4、w2 = 0.5 とすると、総入力が0 以上となるのは、

z = w0x0 + w1x1 + w2x2

= 0.2・1 + 0.4・x1 + 0.5・x2 ≥ 0

となります。この式を整理すると、

x2 ≥ -0.4/0.5x1 - 0.2/0.5

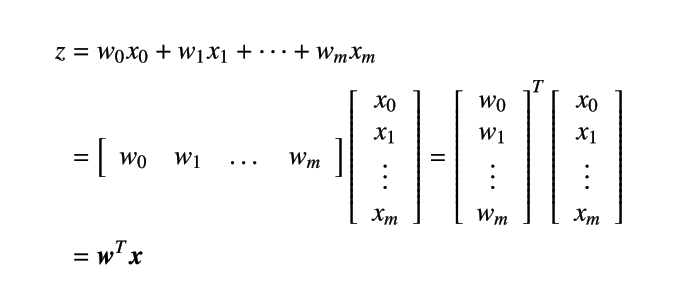
となります。分数を簡潔にすると、

x2 ≥ -0.8x1 - 0.4

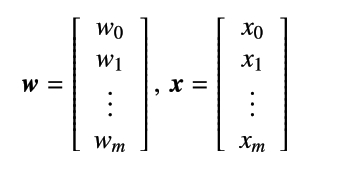
となります。

行列による総入力の表現

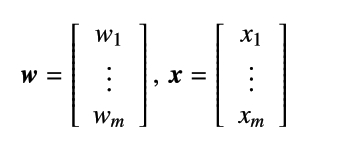
なお、総入力z の式は、行列を用いて書き表すことができます。

[](https://thinkit.co.jp/sites/default/files/article_node/p03_04.jpg%20target=)

ここでは、以下のようにしています。

[](https://thinkit.co.jp/sites/default/files/article_node/p04_01.jpg%20target=)

なお、p.19の式2.1.1 ではベクトル**w**と**x**は以下のように定義されています。

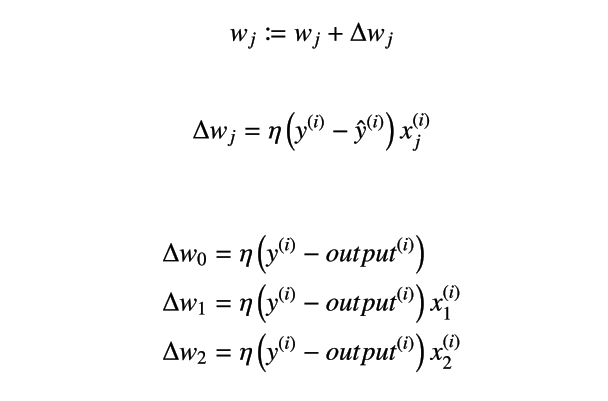
[](https://thinkit.co.jp/sites/default/files/article_node/p04_02.jpg%20target=)

同じページの式 2.1.3 以降では**w**、**x** にそれぞれw0、x0が追加されていることに注意してください。

重みを推定するための計算方法

さて、以上では入力信号の重みw0、・・・、wm は与えられるものとして扱ってきました。しかし、これらの重みは、出力信号が生成される場合と生成されない場合をうまく説明できるように調整する必要があります。Rosenblatt によるパーセプトロンは、次の手順で重みを調整していきます。

* 1. 重みを0 または値の小さい乱数で初期化する。
* 2. トレーニングサンプルx0(i) ごとに以下の手順を実行する。
* （a）出力値ŷを計算する。
* （b）重みを更新する。

[](https://thinkit.co.jp/sites/default/files/article_node/p04_03.jpg%20target=)

# ３　パーセプトロンを読み解く　その２

前回は、書籍『Python機械学習プログラミング』を読むために必要な知識、読み方等について説明しました。今回は、2章の内容についてより分かりやすく理解するためのヒントを解説していきます。

具体例で数式を読み解く

ここからは具体例を考えてみましょう。ここでは、入力信号は2 つあるとします。すなわち、入力信号x1、x2 に対して総入力z を

総入力z = w0x0 + w1x1 + w2x2

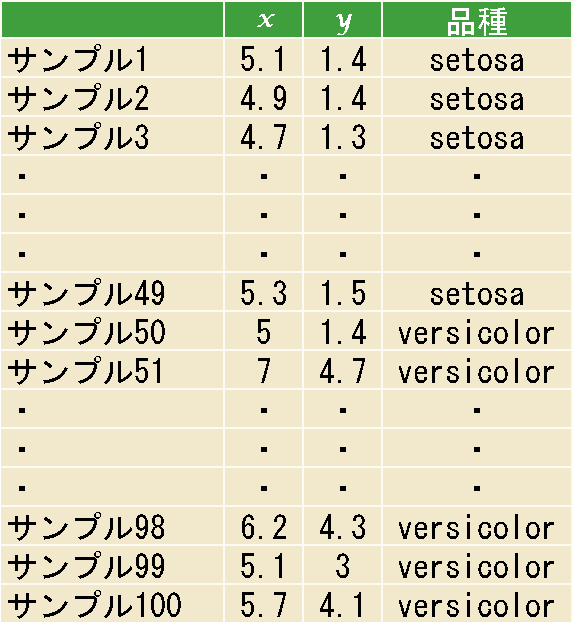
とします。ここで、重みw0、w1、w2 をそれぞれ0 で初期化します。すなわち、以下のように設定します。

w0 = 0, w1 = 0, w2 = 0

ここでは、2.3 節「Iris データセットでのパーセプトロンモデルのトレーニング」で扱われるIris データセットの品種がsetosa, versicolor のデータを抽出してみます。

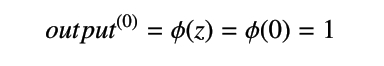
1 番目から50 番目までのサンプルの品種はsetosa、51 番目から100 番目までのサンプルの品種がversicolorです。

1 番目のサンプルは、x1(1) = 5.1、x2(1) = 1.4、y(1) = -1 です。したがって、総入力z は以下のようになります。

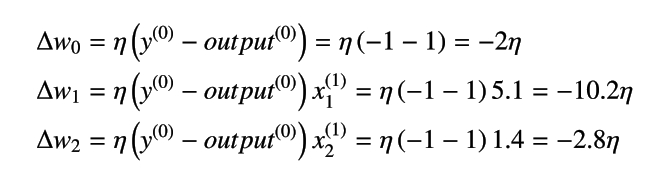
[](https://thinkit.co.jp/sites/default/files/article_node/Hyo02.png)

z = w0x0 + w1x1(1) + w2x2(1) = 0・1 + 0・5.1 + 0・1.4 = 0

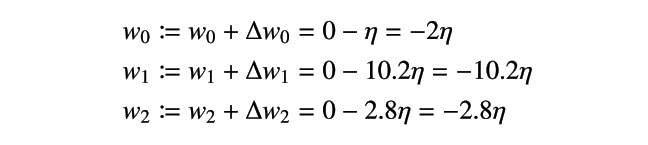
活性化関数の出力は、次のようになります。

[](https://thinkit.co.jp/sites/default/files/article_node/p05_01.jpg%20target=)

以上により、1 番目のサンプルの品種は setosa であり y(0) = -1 であるにもかかわらず、予測結果はŷ(0) = 1 となったことを確認できます。このとき、重みの更新量は、以下のようになります。

[](https://thinkit.co.jp/sites/default/files/article_node/p05_02.jpg%20target=)

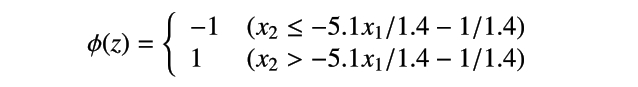
そのため、重み w0、w1、w2 は次のように更新されます。

[](https://thinkit.co.jp/sites/default/files/article_node/p05_03.jpg%20target=)

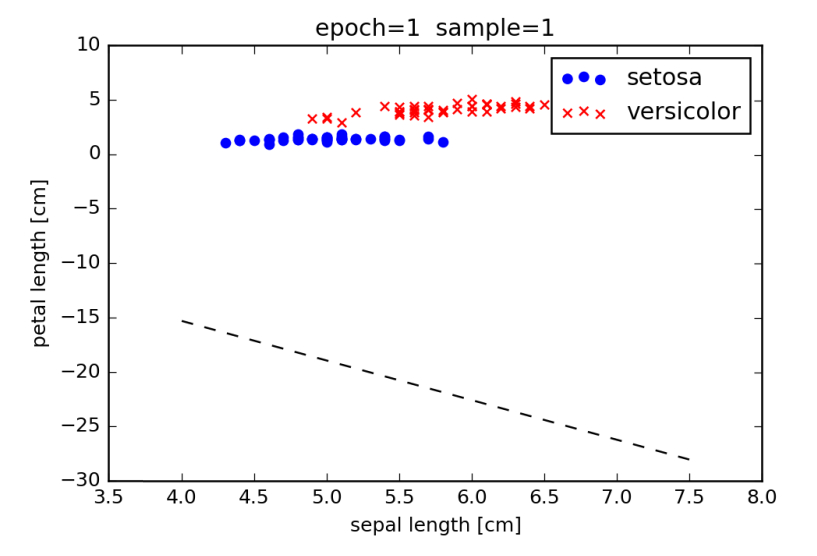
以上により、総入力 z が

[https://thinkit.co.jp/sites/default/files/article_node/p05_04.jpg](https://thinkit.co.jp/sites/default/files/article_node/p05_04.jpg%20target=)

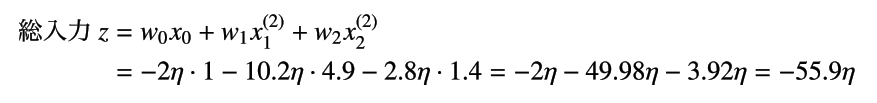
ならば setosa 、そうでないならば versicolorと判定します。したがって、

[](https://thinkit.co.jp/sites/default/files/article_node/p05_05.jpg%20target=)

以上の結果得られる決定境界 (この場合は品種を識別する境界線) を図2に示します。

図2： 1 番目のサンプルを用いて重みを更新した後の決定境界  
[](https://thinkit.co.jp/sites/default/files/article_node/p06_01.jpg)

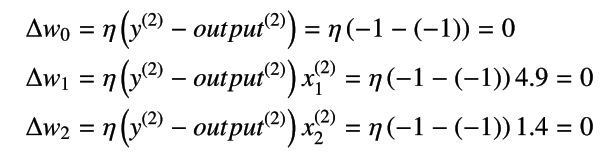
2 番目のサンプルは、x1(2) = 4.9、x2(2) = 1.4、y(2) = -1 です。したがって、総入力は以下のようになります。

[](https://thinkit.co.jp/sites/default/files/article_node/p06_02.jpg%20target=)

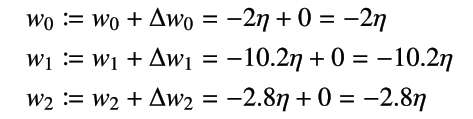
となります。したがって、出力信号は

[https://thinkit.co.jp/sites/default/files/article_node/p06_03.jpg](https://thinkit.co.jp/sites/default/files/article_node/p06_03.jpg%20target=)

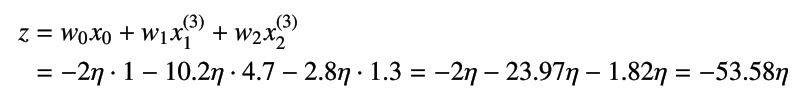
となります。よって、重みの更新量は以下のようになります。

[](https://thinkit.co.jp/sites/default/files/article_node/p06_04.jpg%20target=)

したがって、重みは以下のように更新されます。

[](https://thinkit.co.jp/sites/default/files/article_node/p06_05.jpg%20target=)

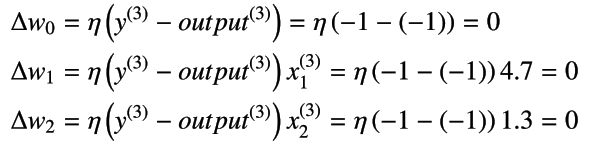
2 番目のサンプルを用いると重みの値は更新されないことを確認できます。3 番目のサンプルはx1(3) = 4.7、x2(3) = 1.3、y(3) = -1 です。したがって、

[](https://thinkit.co.jp/sites/default/files/article_node/p06_06.jpg%20target=)

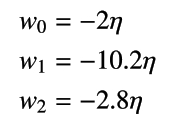
となります。したがって、出力信号は

[https://thinkit.co.jp/sites/default/files/article_node/p06_07.jpg](https://thinkit.co.jp/sites/default/files/article_node/p06_07.jpg%20target=)

となります。重みの更新量は、以下のようになります。

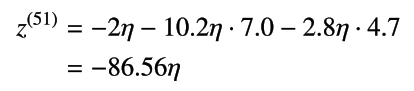
[](https://thinkit.co.jp/sites/default/files/article_node/p07_01.jpg%20target=)

以上のようにして、2 番目から 50 番目までのサンプルでは重みは更新されません。したがって、50 番目のサンプルが終了した時点での重みは、

[](https://thinkit.co.jp/sites/default/files/article_node/p07_02.jpg%20target=)

となります。

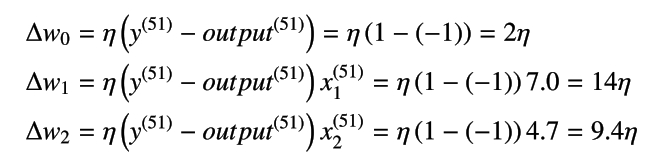
続いて、51 番目のサンプルはx1(51) = 7.0、x2(51) = 4.7、y(51) = 1 となります。 総入力 z(51)は、以下のようになります。

[](https://thinkit.co.jp/sites/default/files/article_node/p07_03.jpg%20target=)

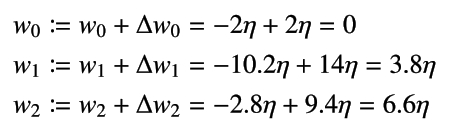
したがって、

[https://thinkit.co.jp/sites/default/files/article_node/p07_04.jpg](https://thinkit.co.jp/sites/default/files/article_node/p07_04.jpg%20target=)

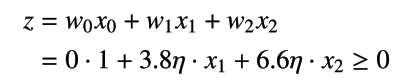
となります。重みの更新量は以下のようになります。

[](https://thinkit.co.jp/sites/default/files/article_node/p07_05.jpg%20target=)

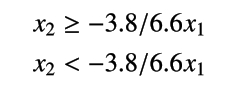
重みは以下のように更新されます。

[](https://thinkit.co.jp/sites/default/files/article_node/p07_06.jpg%20target=)

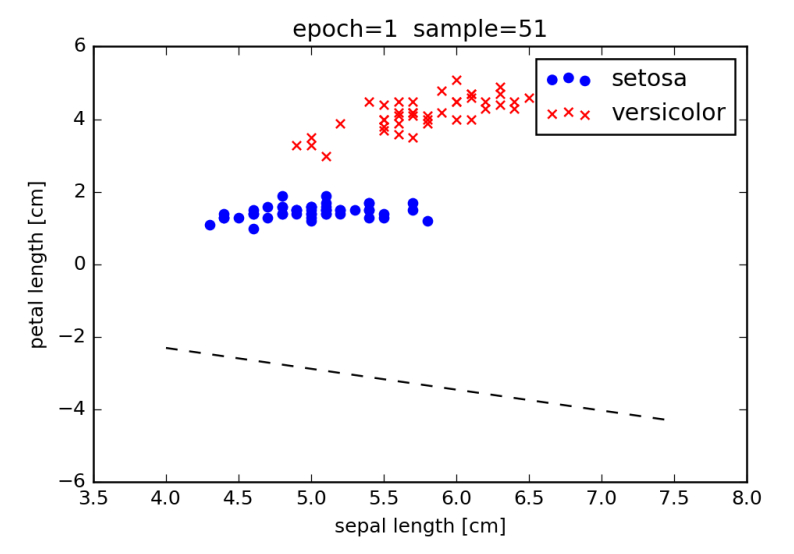
以上により、

[](https://thinkit.co.jp/sites/default/files/article_node/p07_07.jpg%20target=)

このときは品種が versicolor、そうではないときは品種が setosa と判定されることを確認できます。以上を整理すると、

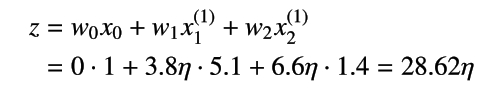
[](https://thinkit.co.jp/sites/default/files/article_node/p07_08.jpg%20target=)

以上の状況を可視化すると、図3を得ます。

図3： 51 番目のサンプルを用いて重みを更新した後の判別境界  
[](https://thinkit.co.jp/sites/default/files/article_node/p08_G01.jpg)

52 番目から 100 番目のサンプルでは重みの更新が行われません。

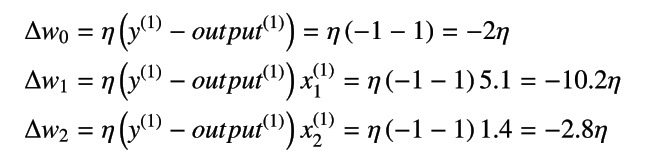
続いて、2 巡目に入っていきます。1 番目のサンプルは、x1(1) = 5.1、x2(1) = 1.4、y(1) = -1 です。したがって、総入力zは以下のようになります。

[](https://thinkit.co.jp/sites/default/files/article_node/p08_01.jpg%20target=)

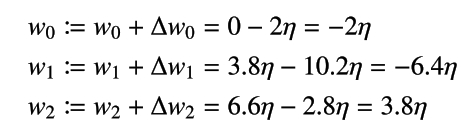
となります。よって、

[https://thinkit.co.jp/sites/default/files/article_node/p08_02.jpg](https://thinkit.co.jp/sites/default/files/article_node/p08_02.jpg%20target=)

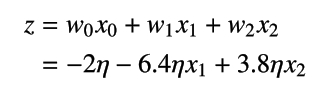
となります。重みの更新量は、

[](https://thinkit.co.jp/sites/default/files/article_node/p08_03.jpg%20target=)

となります。したがって、

[](https://thinkit.co.jp/sites/default/files/article_node/p08_04.jpg%20target=)

以上の結果、総入力 z は、以下のように表されます。

[](https://thinkit.co.jp/sites/default/files/article_node/p08_05.jpg%20target=)

したがって、活性化関数が

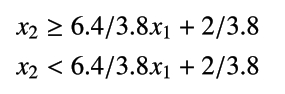
[https://thinkit.co.jp/sites/default/files/article_node/p09_01.jpg](https://thinkit.co.jp/sites/default/files/article_node/p09_01.jpg%20target=)

のときは 1 を、

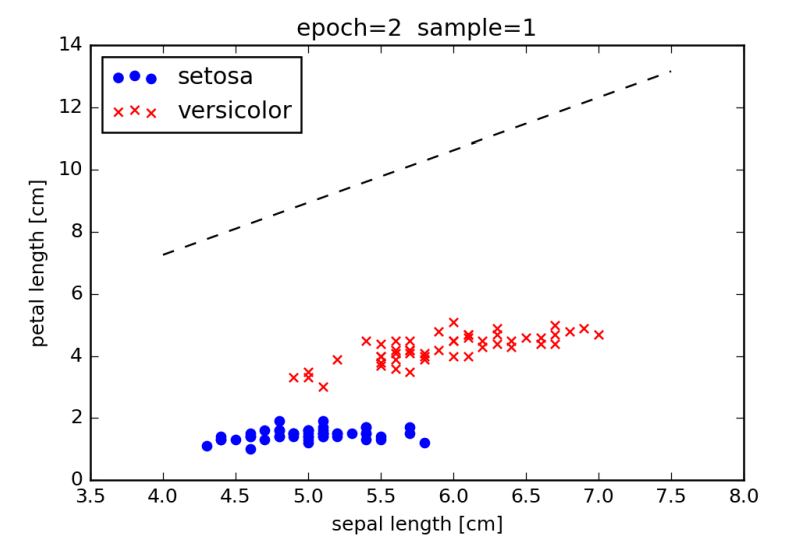
[https://thinkit.co.jp/sites/default/files/article_node/p09_02.jpg](https://thinkit.co.jp/sites/default/files/article_node/p09_02.jpg%20target=)

のときは -1 となります。

以上を整理すると、

[](https://thinkit.co.jp/sites/default/files/article_node/p09_03.jpg%20target=)

となります。この状況を可視化すると、図4のようになります。判別境界の直線が右肩上がりになり、さらにすべての点はこの直線よりも下側にあること (つまり、この判別境界を用いるとすべてが setosa と判定される) を確認できます。

図4：1 番目のサンプルを用いて重みを更新した後の決定境界  
[](https://thinkit.co.jp/sites/default/files/article_node/p09_G01.jpg)

# ４　パーセプトロンを読み解く　その３

さて、以上の数式をプログラムで記述してみましょう。2.2 節「パーセプトロンの学習アルゴリズムをPython で実装する」では、クラスを用いてパーセプトロンを実装しています。ここでは、より簡単に実装してみましょう。

まずは、本書と同様の方法により Iris データセットを読み込みます。

In [1]: %matplotlib inline

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

plt.rcParams['savefig.dpi'] = 180

In [2]: df = pd.read\_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/iris/' \

'iris.data', header=None)

df.tail()

Out[2]: 0 1 2 3 4

145 6.7 3.0 5.2 2.3 Iris-virginica

146 6.3 2.5 5.0 1.9 Iris-virginica

147 6.5 3.0 5.2 2.0 Iris-virginica

148 6.2 3.4 5.4 2.3 Iris-virginica

149 5.9 3.0 5.1 1.8 Iris-virginica

In [3]: y = df.iloc[0:100, 4].values

y = np.where(y == 'Iris-setosa', -1, 1)

X = df.iloc[0:100, [0, 2]].values

散布図をプロットすると次のようになります。

In [4]: # 品種setosaのプロット(赤の○)

plt.scatter(X[:50,0], X[:50,1], color='red', marker='o', label='setosa')

# 品種versicolorのプロット(青の×)

plt.scatter(X[50:100,0], X[50:100,1], color='blue', marker='x', label='versicolor')

# 軸のラベルの設定

plt.xlabel('sepal length [cm]')

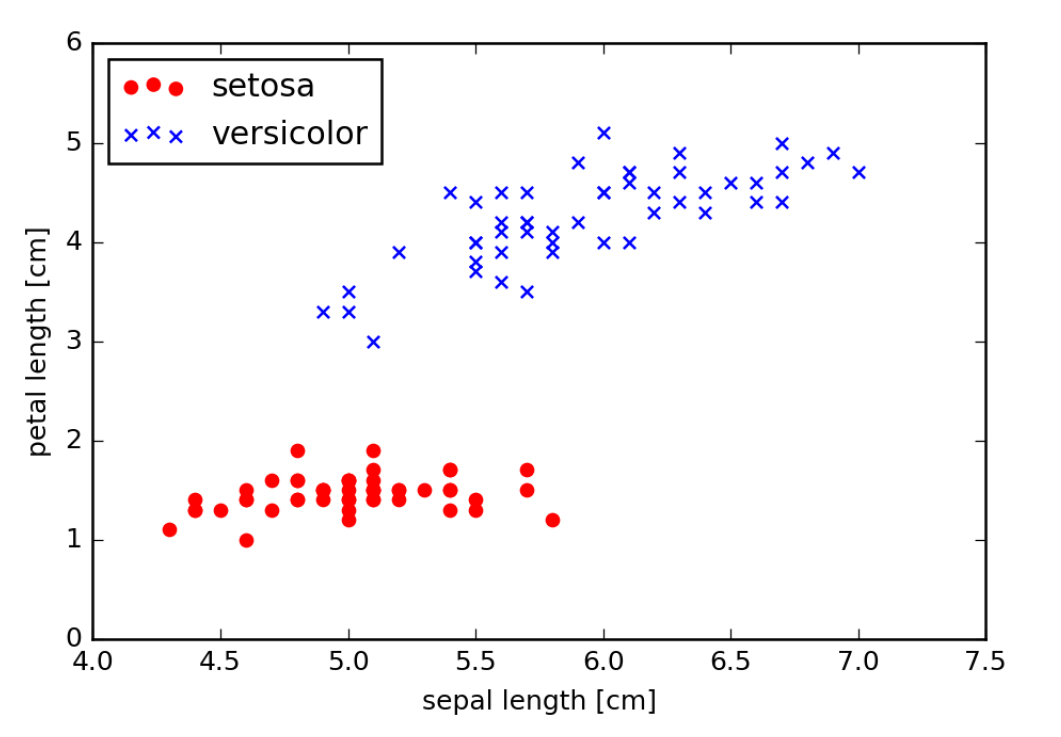
plt.ylabel('petal length [cm]')

# 凡例の設定(左上に配置)

plt.legend(loc='upper left')

# 図の表示

plt.show()

[](https://thinkit.co.jp/sites/default/files/article_node/p10_G01.jpg)

パーセプトロンを以下の方針で実装してみましょう。

* **学習率 η**：ここでは、 η = 0.01 と設定します。変数名は eta とします。
* **反復回数 N**：ここでは、N = 10 と設定します。変数名は n\_iter とします。
* **重み w0、w1、w2**：本書内では NumPy 配列を使用していますが、ここでは変数 w0、w1、w2 を定義して、それぞれを 0 で初期化することにします。

In [5]: import numpy as np

eta = 0.01 # 学習率

n\_iter = 10 # トレーニングデータのトレーニング回数

w0 = w1 = w2 = 0 # 重みを初期化

w0\_, w1\_, w2\_ = [], [] ,[] # 重みを格納するリスト

errors\_ = [] # 誤差を格納するリスト

for epoch in range(n\_iter): # トレーニング回数分トレーニングデータを反復

errors = 0

for xi, target in zip(X, y): # 各サンプルで重みを更新

# 総入力の計算

net\_input = w0 + w1 \* xi[0] + w2 \* xi[1]

# 総入力が0以上の場合は1、0未満の場合は-1と予測

pred = np.where(net\_input >= 0.0, 1, -1)

# 重みを更新

update = eta \* (target - pred)

w1 += update \* xi[0]

w2 += update \* xi[1]

w0 += update

# 誤差を算出

errors += int(update != 0.0)

# 重みを格納

w0\_.append(w0)

w1\_.append(w1)

w2\_.append(w2)

# 誤差を格納

errors\_.append(errors)

上記のソースコードは、2.3 節でクラスを用いて実装されているパーセプトロンのエッセンスのみを実装しています。Python のクラスにあまりなじみのない方は上記のコードで処理の流れを把握して、2.3 節の実装を確認すると良いでしょう。

以上のコードを実行した結果、各反復の完了時点での誤差をプロットすると以下のようになります。6 回目の反復で収束していることを確認できます。

In [6]: # 各反復での誤差をプロット

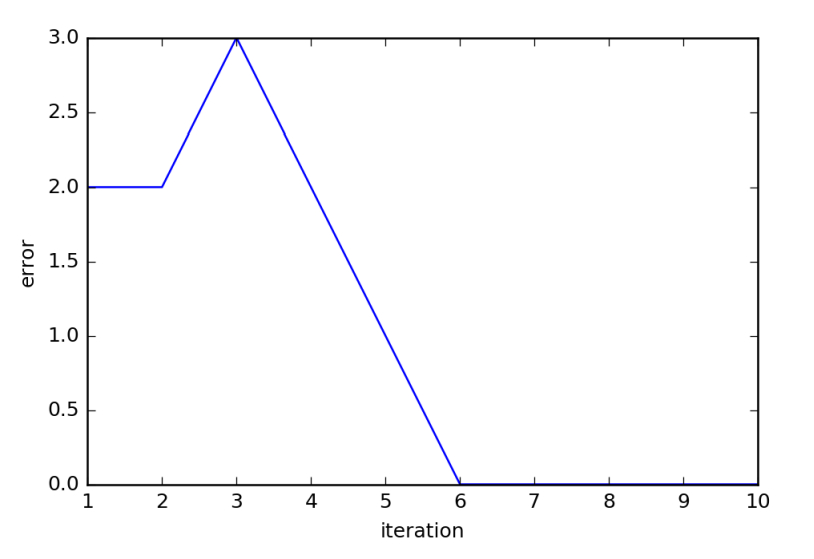
iters = range(1, n\_iter+1)

plt.plot(iters, errors\_)

plt.xlabel('iteration')

plt.ylabel('error')

plt.show()

[](https://thinkit.co.jp/sites/default/files/article_node/p12_G01.jpg)

重み w0、w1、w2の変化をプロットすると次のようになります。

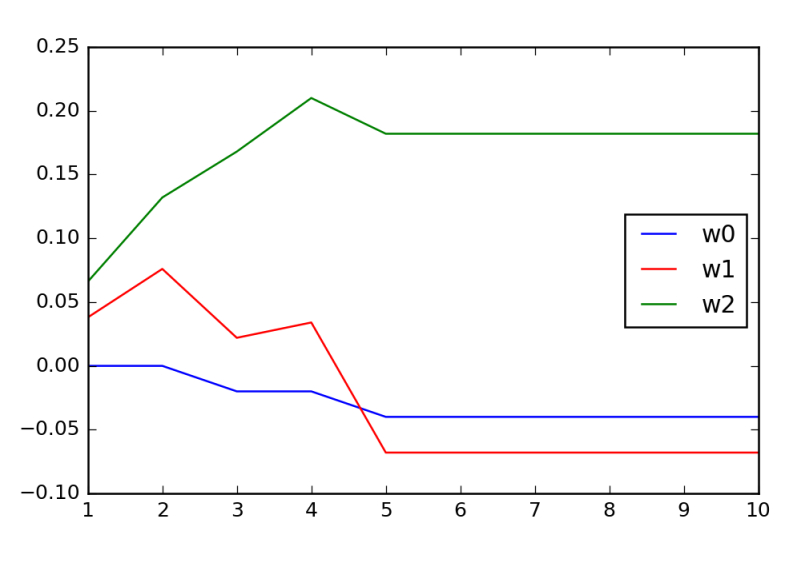
In [7]: plt.plot(iters, w0\_, c='blue', label='w0')

plt.plot(iters, w1\_, c='red', label='w1')

plt.plot(iters, w2\_, c='green', label='w2')

plt.legend(loc='best')

plt.show()

[](https://thinkit.co.jp/sites/default/files/article_node/p13_G01.jpg)

それぞれの重みは、5回目の反復で概ね収束していることを確認できます。

# ５　単純な例でADALINEのイメージをつかむ　その１

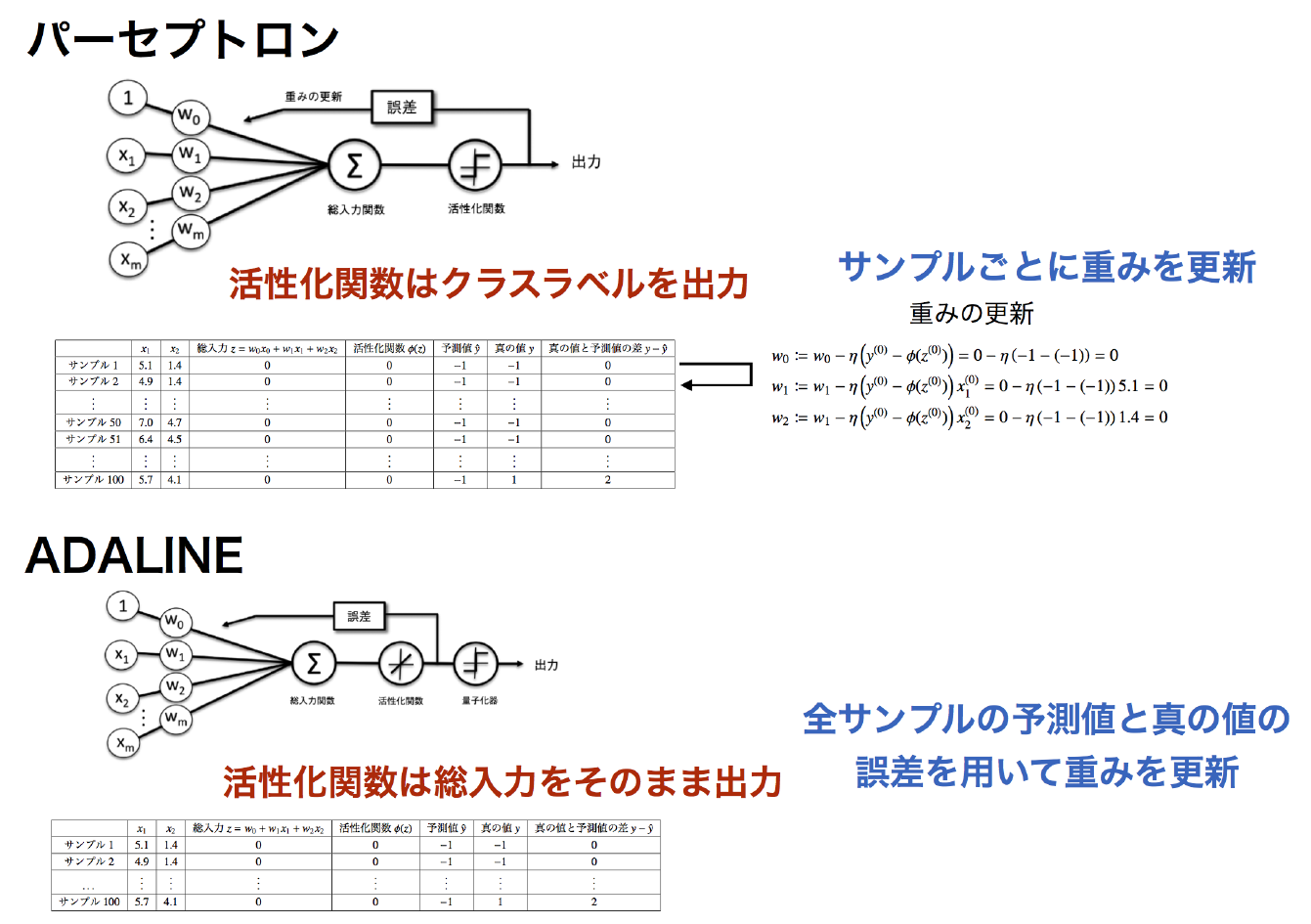
今回も前回に引き続き、書籍『Python機械学習プログラミング』の2章の内容についてより分かりやすく理解するためのヒントを解説していきます。今回は、ADALINEのイメージがつかめるように、ADALINEのシンプルな例を取り上げます。

原著の『Python Machine Learning』は2015年9月に米国などで発売。「機械学習の考え方」と「Pythonプログラミングによる実践」をバランスよく解説していると評価され、米国Amazon.comでベストセラー。その日本語訳はコラムや脚注、付録が追加され、2016年6月に発売されました。ここでは、本書のより効果的な活用法について監訳者が解説します。

ADALINE

本書の2章では、「ADALINEと学習の収束」から「大規模な機械学習と確率的勾配降下法」（2.4～2.6節）の中で、ADALINE（ADAptive LInear NEuron）というアルゴリズムが扱われます。前回説明したパーセプトロンとADALINEの主な違いは、図2-1のようにまとめられます。

図2-1：パーセプトロンと ADALINE の相違点

[](https://thinkit.co.jp/sites/default/files/article_node/Zu2-1.png)

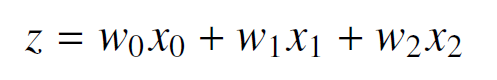
**■活性化関数の出力**  
パーセプトロンの活性化関数は、総入力をクラスを表す数値に変換しますが、ADALINEの活性化関数の出力は総入力そのものになります。

**■重みの更新**  
パーセプトロンは重みをサンプルごとに更新しますが、ADALINEは全サンプルに対して更新します。

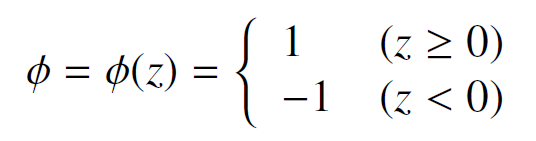
ここでは、ADALINEについてより詳しく補足的な解説を行っていきます。

誤差平方和：真の値と予測値の差を2乗して足し合わせる

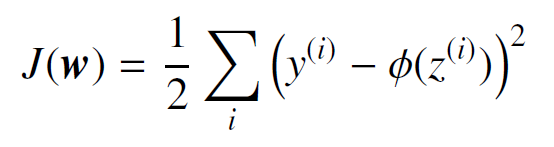
パーセプトロンでは、総入力*z*を各入力信号に重みをかけて足し合わせたものとして計算していました。例えば、入力信号が2つの場合（*x*1と*x*2）、総入力*z*を次式で定めていました（*w*0は定数項のため、*x*0 = 1）。

[](https://thinkit.co.jp/sites/default/files/article_node/3-1.png%20target=)

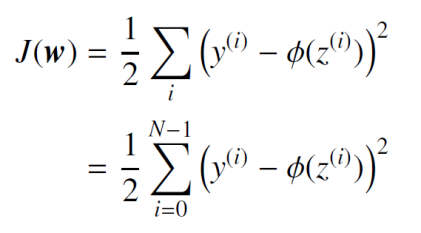
また、出力については、総入力が0以上かどうかを判定して、総入力が0以上の場合はクラス1を、総入力が0未満の場合はクラス-1を割り当てていました。すなわち、次式の活性化関数を定義していました。

[](https://thinkit.co.jp/sites/default/files/article_node/3-2.png%20target=)

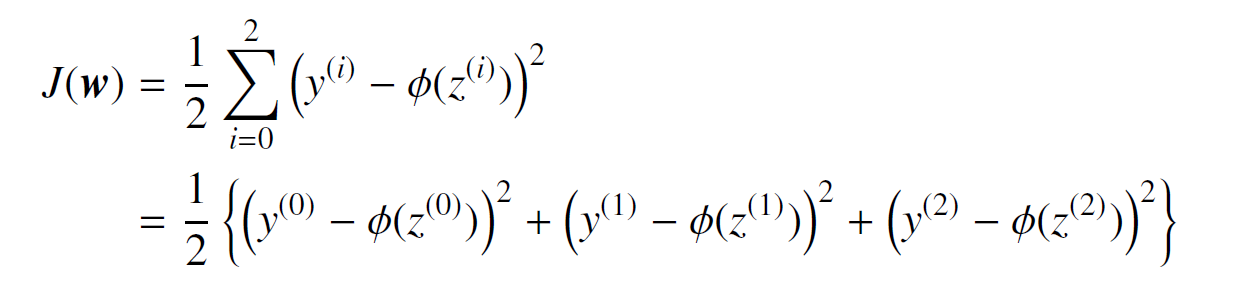
ADALINEで使用される誤差は、誤差平方和（Sum of Squared Error : SSE）です。誤差平方和は、「**各サンプルの真の値と予測値の差（残差）を2乗したものをすべてのサンプルに対して足し合わせたもの**」です。つまり、誤差平方和が大きいほどサンプル全体として真の値と予測値の差が平均的に大きいことになります。i 番目のサンプルの真の値を *y(i)*、予測値を φ(*z(i)*) とすると、誤差平方和は次式で表されます。

[](https://thinkit.co.jp/sites/default/files/article_node/3-3.png%20target=)

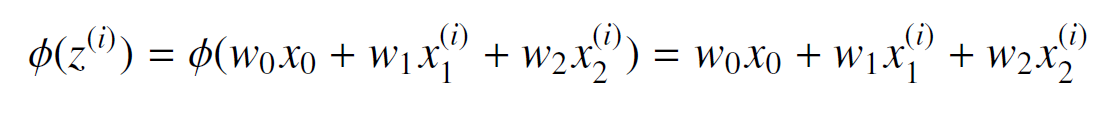
*N*個のサンプルのインデックスを *i* = 0 から *i* = *N* − 1 までとすると、上式は以下のように表すことができます。

[](https://thinkit.co.jp/sites/default/files/article_node/3-4.png%20target=)

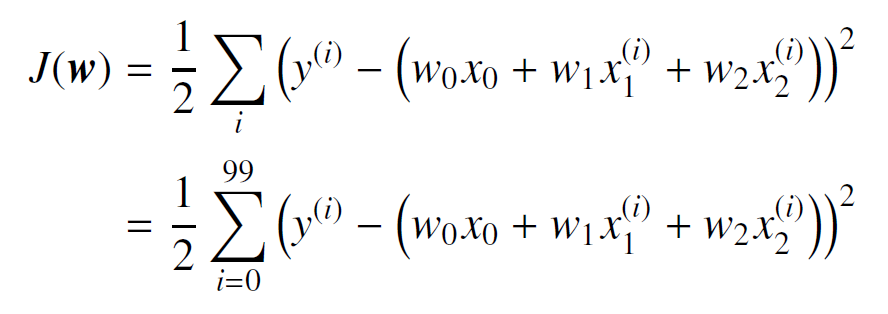
例えば、*N* = 3 の場合（サンプルが3個）では、以下のようになります。

[](https://thinkit.co.jp/sites/default/files/article_node/3-5.png%20target=)

パーセプトロンで扱った例の場合、*i* 番目のサンプルに対して活性化関数の出力は、次のとおりです。

[](https://thinkit.co.jp/sites/default/files/article_node/3-6.png%20target=)

したがって、誤差平方和は 1 番目から 100 番目までのサンプルの真の値と予測値の差 （残差）を 2 乗して足し合わせたものになります。したがって、誤差平方和は次のとおりです。

[](https://thinkit.co.jp/sites/default/files/article_node/3-7.png%20target=)

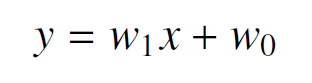
この誤差平方和が小さくなれば、真の値と予測値は平均的に近くなります。このような重み *w*0、*w*1、*w*2を探す必要があります。

# ６　単純な例でADALINEのイメージをつかむ　その２

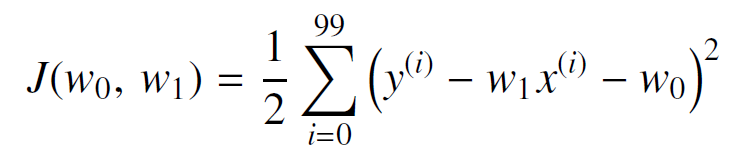
今回も前回に引き続き、書籍『Python機械学習プログラミング』の2章の内容についてより分かりやすく理解するためのヒントを解説していきます。今回は、ADALINEのイメージがつかめるように、ADALINEのシンプルな例を取り上げます。

勾配降下法: 誤差平方和の勾配と逆方向に重みをずらしていく

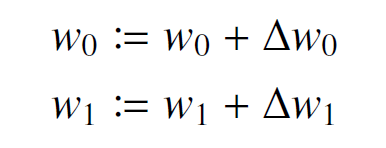
ここでは、勾配降下法を用いて重みの更新を行いながら、最適な重みの値に近づけていきます。本書では、一般的な記法で記述されていますが、ここでは簡単な例で説明します。Irisデータセットの品種がsetosaまたはversicolorの100サンプルについて、2番目の特徴量（Petal-Length）と品種（setosaまたはversicolor）の関係を説明します。*x*をPetal-Length、*y*を品種（setosaのとき-1、versicolorのとき1）として、これらの関係を次の式で表します。この式の*w*1は傾き、*w*0は切片です。

[](https://thinkit.co.jp/sites/default/files/article_node/3-8.png%20target=)

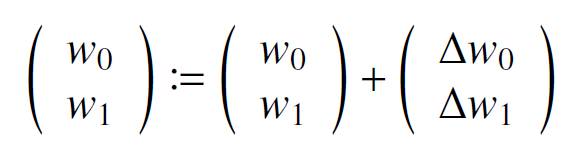
*w*0、*w*1の最適値を勾配降下法を用いて求めてみましょう。この場合、以下の式が最小となる*w*1、*w*0を推定します。

[](https://thinkit.co.jp/sites/default/files/article_node/3-9.png%20target=)

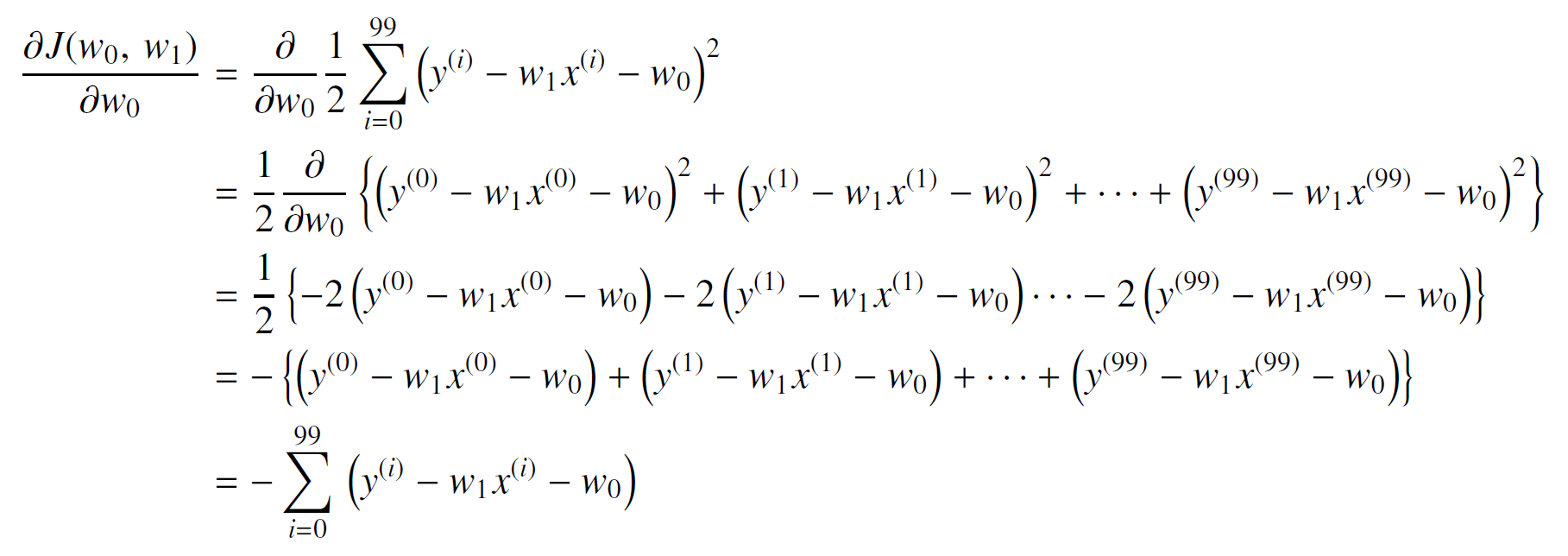
この例の場合は、最小二乗法などを用いて、解析的に *J*(*w*0, *w*1 ) を最小化する*w*0、*w*1を求めることができます。しかし、ここではそうしたアプローチを行うのではなく、*w*0、*w*1を少しずつ変化させながら収束させていく方法をとってみましょう。そのために、以下の式により重み*w*0、*w*1の更新処理を繰り返していきます。

[](https://thinkit.co.jp/sites/default/files/article_node/3-10.png%20target=)

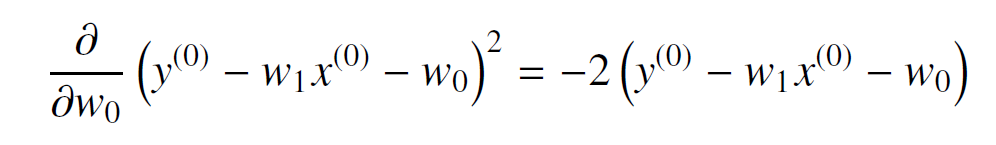
Δ*w*0、Δ*w*1は重みの更新の大きさを表しています。これらの式をまとめると、次のように行列を用いて表すこともできます。

[](https://thinkit.co.jp/sites/default/files/article_node/3-11.png%20target=)

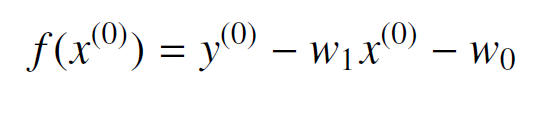
更新の大きさΔ*w*0、Δ*w*1はどのように決めればよいのでしょうか。**勾配降下法では「誤差平方和の勾配方向」**にとります。誤差平方和の勾配とは、「誤差平方和 *J*(*w*0, *w*1 ) の *w*0、*w*1に関する偏導関数」となります。*w*0に関する偏導関数は、次式のようになります。

[](https://thinkit.co.jp/sites/default/files/article_node/3-12.png%20target=)

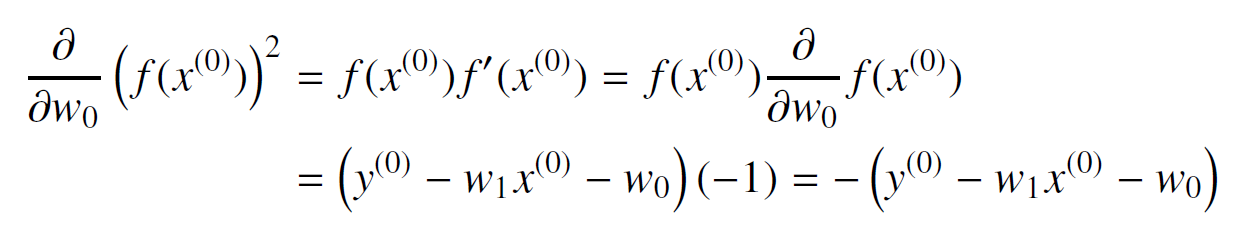
2行目から3行目の変形では、例えば、次の関係を用いています。

[](https://thinkit.co.jp/sites/default/files/article_node/3-13.png%20target=)

わかりにくい場合は、まず次の式をおきます。

[](https://thinkit.co.jp/sites/default/files/article_node/3-14.png%20target=)

この式に対して、連鎖律を使って次のように変形するとよいかもしれません。

[](https://thinkit.co.jp/sites/default/files/article_node/3-15.png%20target=)

同様にして、*w*1に関する偏導関数は、次のようになります。

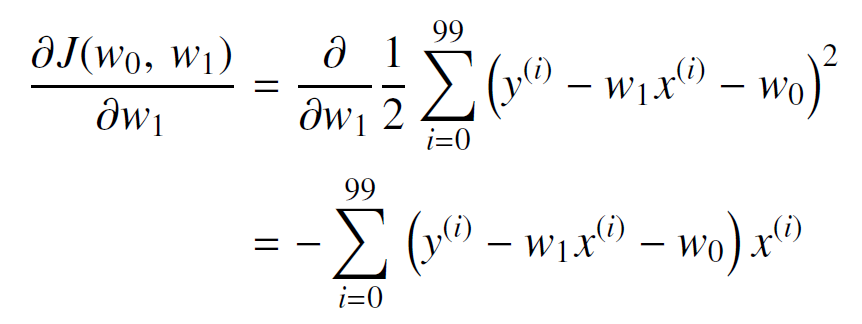
[](https://thinkit.co.jp/sites/default/files/article_node/3-16.png%20target=)

図2-2 は、*w*0、*w*1ともに − 5 から 5 までで 0.01 刻みで点を取ったときに誤差の値をプロットしています。また、各点における勾配の向きと大きさをベクトルで表しています。

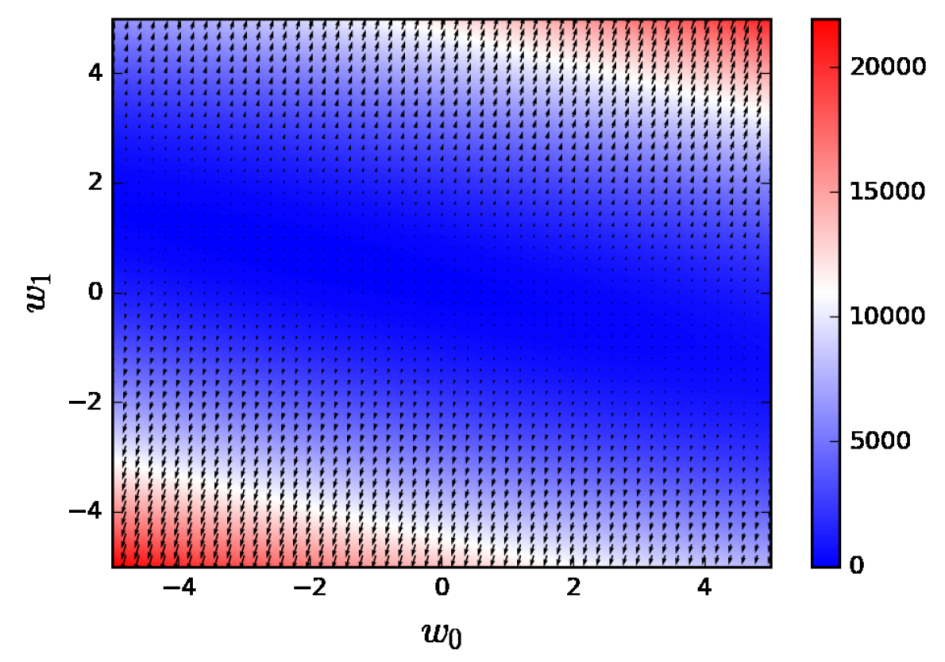
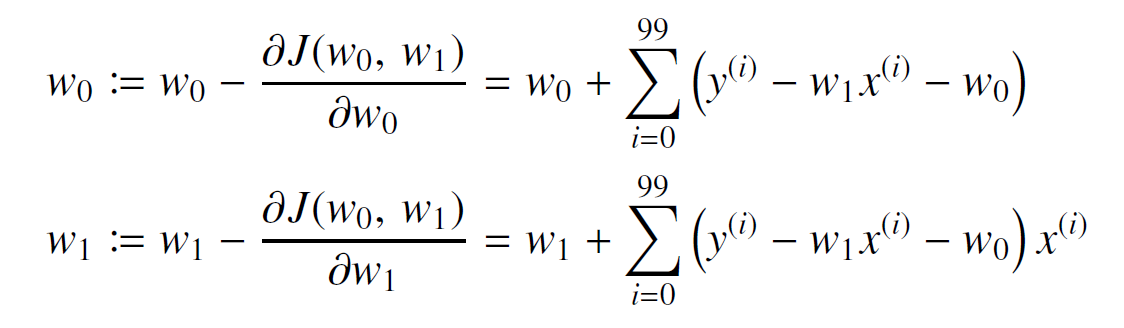
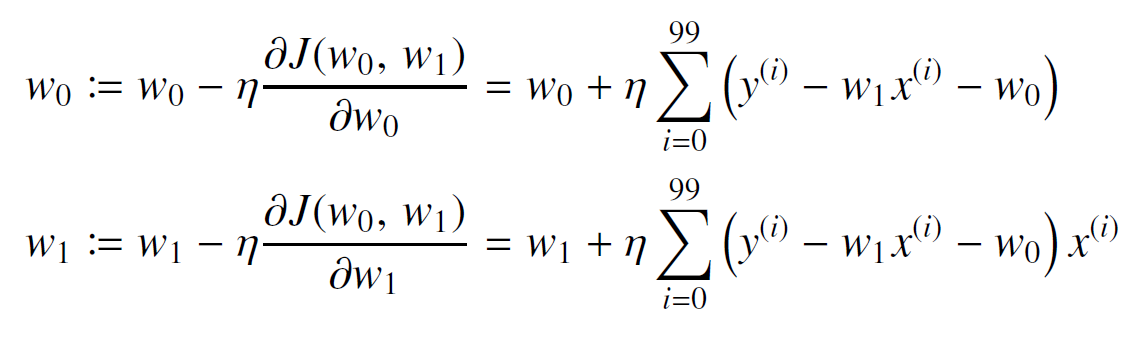
図2-2：誤差とその勾配。横軸：*w*0、縦軸：*w*1  
[](https://thinkit.co.jp/sites/default/files/article_node/Zu2-2.png)

図2-2 を見ると、図の左側のやや上側付近から右側のやや下側付近にかけて青色が濃い領域が存在していることを確認できます。この領域では、誤差および勾配が小さくなっていることを確認できます。

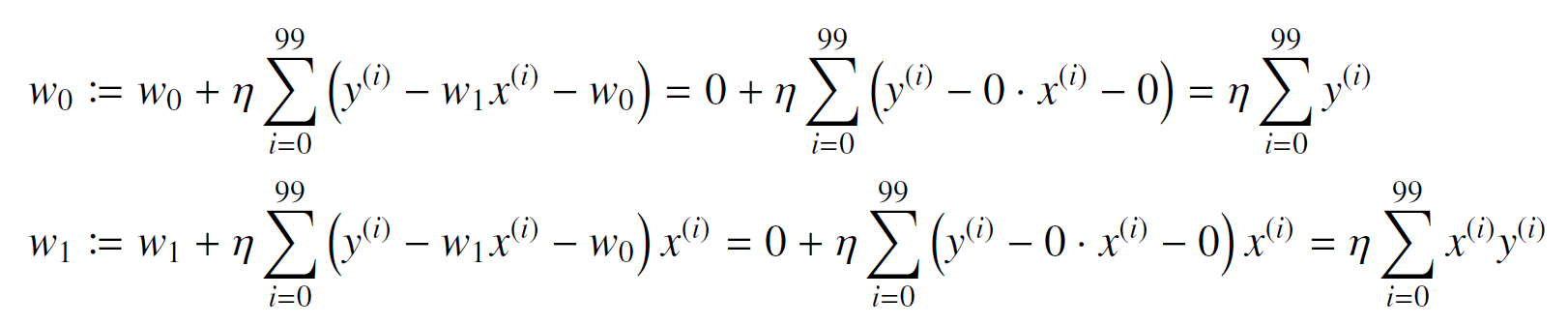
勾配降下法では、誤差を最小化するために、それぞれの重みにおいて誤差が最も小さくなる方向に重みをずらしていく処理を繰り返していきます。誤差が最も小さくなる方向とは、勾配の逆方向です。したがって、先ほど求めた∂*J*(*w*0, *w*1 ) ⁄ ∂*w*0、∂*J*(*w*0, *w*1 ) ⁄ ∂*w*1にマイナスをつけた−∂*J*(*w*0, *w*1 ) ⁄ ∂*w*0、−∂*J*(*w*0, *w*1 ) ⁄ ∂*w*1によって、*w*0、*w*1を更新します。よって、次の式で更新することになります。

[](https://thinkit.co.jp/sites/default/files/article_node/3-17.png%20target=)

さらに、学習率というパラメータを導入します。学習率は、「どの程度勾配の逆方向にずらすか」を表すパラメータです。学習率を *η* とすると、*w*0、*w*1の更新ルールは次式のように表されます。

[](https://thinkit.co.jp/sites/default/files/article_node/3-18.png%20target=)

例えば、重み*w*0、*w*1の初期値を *w*0 = 0、*w*1 = 0 と設定します。このとき、1 回目の重みの更新は、次式により行われます。

[](https://thinkit.co.jp/sites/default/files/article_node/3-19.png%20target=)

# ７　単純な例でADALINEのイメージをつかむ　その３

今回も前回に引き続き、書籍『Python機械学習プログラミング』の2章の内容についてより分かりやすく理解するためのヒントを解説していきます。今回は、ADALINEのイメージがつかめるように、ADALINEのシンプルな例を取り上げます。

ADALINEを実装する

勾配降下法の理解が進んだところで、ADALINEを実装してみましょう。ここでも、本書とは異なり、クラスを用いずに簡易的に実装してみます。

まず、データを作成します。

# 1-100 行目の目的変数の抽出

y = df.iloc[0:100, 4].values

# Iris-setosa を-1、Iris-virginica を 1 に変換

y = np.where(y == ' Iris-setosa ' , -1, 1)

# 1-100 行目の 3 列目の抽出

X = df.iloc[:100, [2]].values

続いて、ADALINEを実装します。ここでは、学習率 *η* = 0.01、トレーニング回数を20として、*w*0、*w*1をそれぞれ 0 で初期化して計算します。

eta = 0.01 # 学習率

n\_iter = 20 # トレーニングデータのトレーニング回数

w0, w1 = 0.0, 0.0 # 重みを 0 で初期化

w0\_, w1\_ = [], [] # 重みを格納するリスト

cost\_ = [] # 誤差を格納するリスト

for epoch in range(n\_iter): # トレーニング回数分トレーニングデータを反復

# 総入力

output = w1 \* X + w0

# 誤差

error = (y - output.ravel())

# 重みの更新

w0 += eta \* error.sum()

w1 += eta \* (error \* X.ravel()).sum()

# 誤差の格納

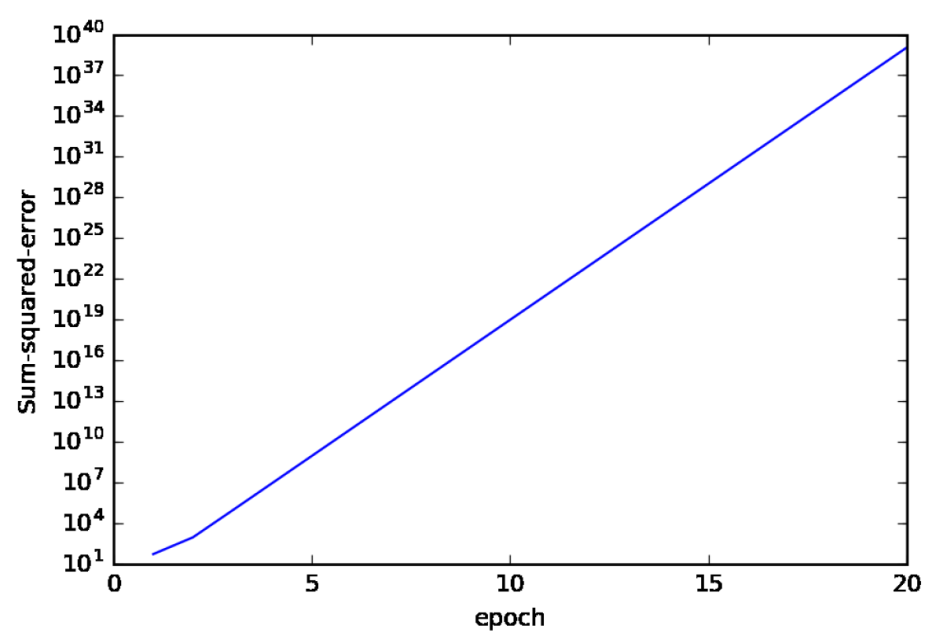
cost\_.append(0.5 \* np.sum(error\*\*2))

# 重みの格納

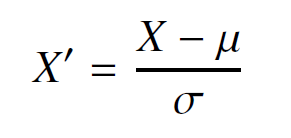
w0\_.append(w0)

w1\_.append(w1)

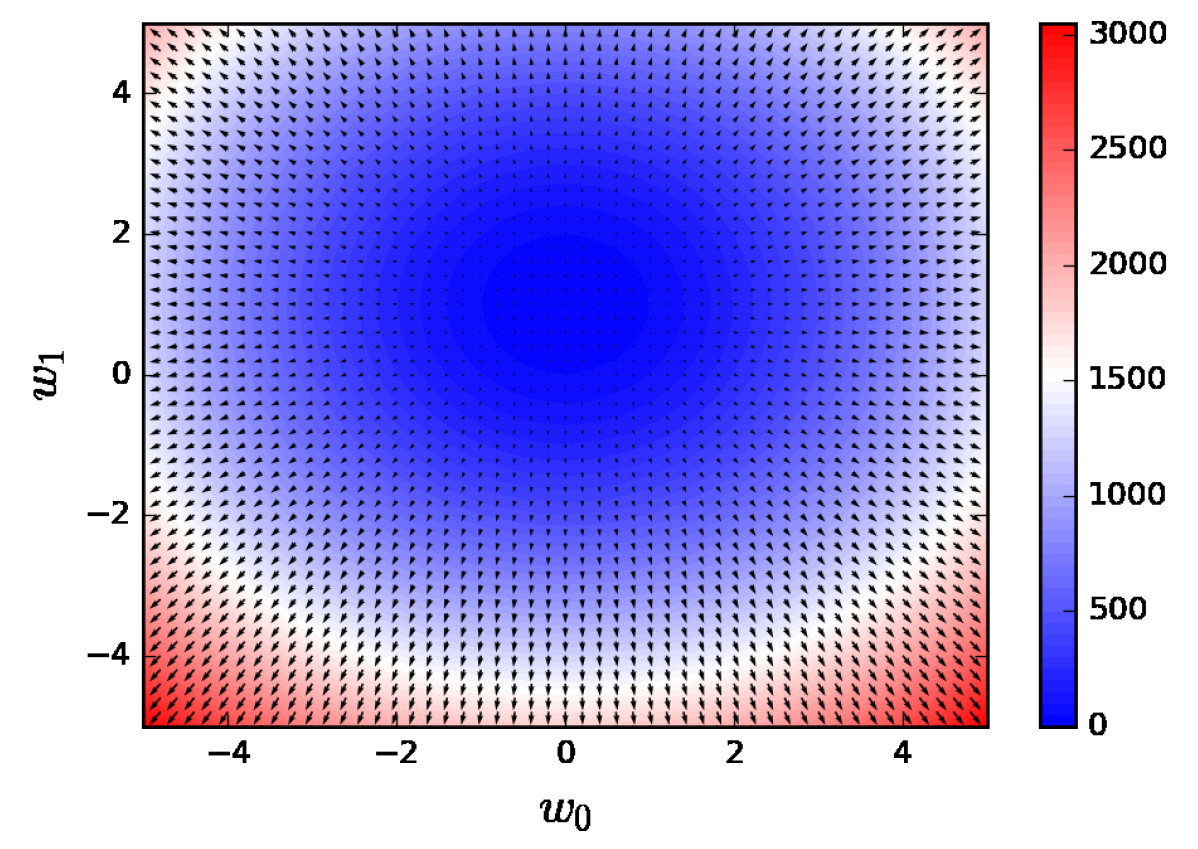
さて、学習率 *η* = 0.01 と設定してプログラムを実行し、横軸に反復回数、縦軸に誤差平方和をプロットすると図2-3が得られます。誤差平方和は反復の度に指数関数的に増加してしまい、収束しないことを図から確認できます。

図2-3: 反復回数と誤差平方和の関係。横軸: 反復回数、縦軸: 誤差平方和  
[](https://thinkit.co.jp/sites/default/files/article_node/Zu2-3.png)

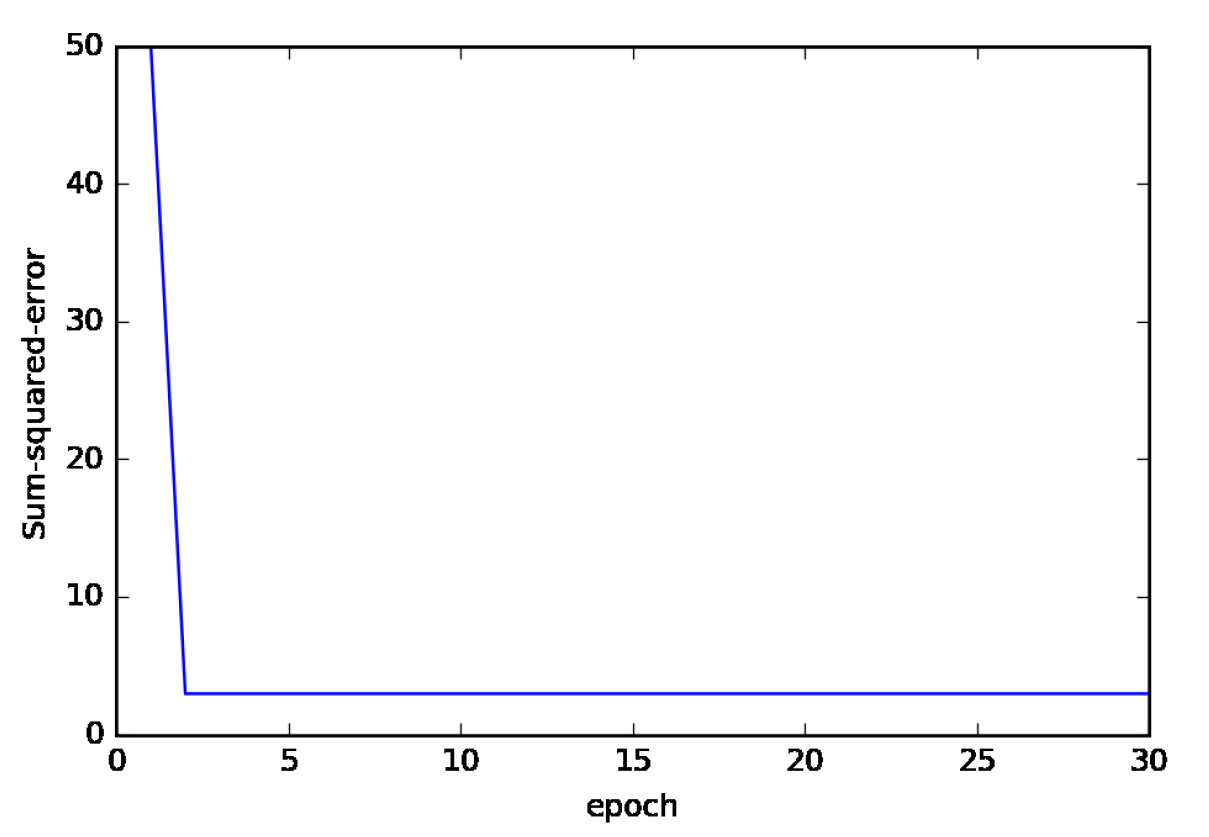
ここでは、本書の2.5.1項「ADALINE を Python で実装する」で行われているように、各変数を標準化してみます。ここでの標準化とは、それぞれの変数をその平均値を引いた後にその分散で割る（「分散正規化」とも呼ばれます）処理です。つまり、次式による標準化です。

[](https://thinkit.co.jp/sites/default/files/article_node/3-20.png%20target=)

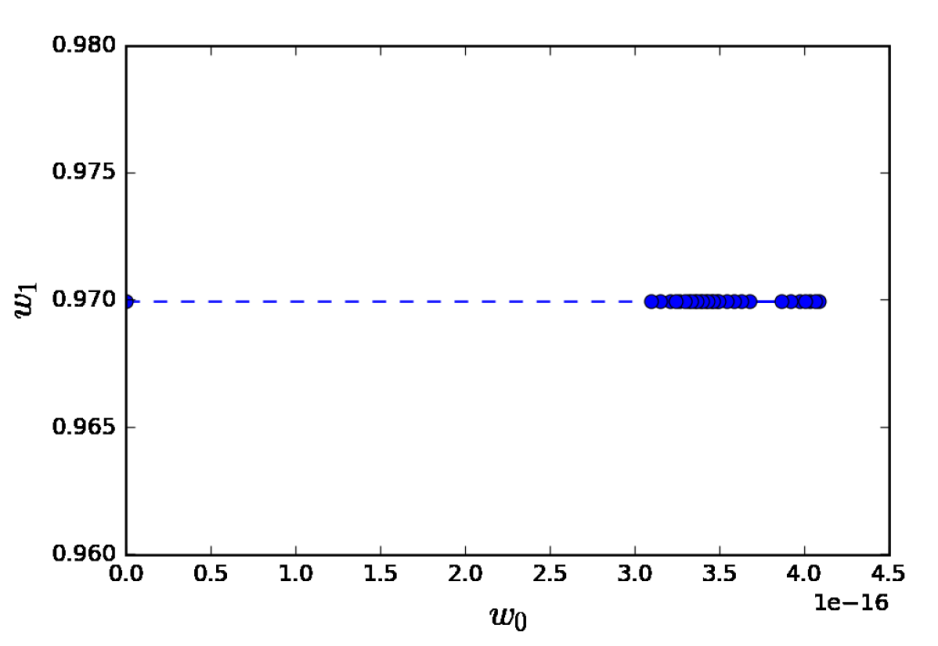
標準化を行った後に、重み *w*0、*w*1の組み合わせに対する誤差とその勾配は、図2-4のようになります。

図2-4: 標準化後の誤差とその勾配  
[](https://thinkit.co.jp/sites/default/files/article_node/Zu2-4.png)

こうして正規化を行った後に ADALINE を実行すると、今度は無事に収束していることを確認できます。勾配降下法により重み*w*0、*w*1が収束する様子を可視化すると、図2-5のようになります。

図2-5：勾配降下法による重み*w*0、*w*1の収束  
[](https://thinkit.co.jp/sites/default/files/article_node/Zu2-5.png)

横軸に*w*0、縦軸に*w*1をとり、収束していく様子をプロットすると図2-6のようになります。

図2-6：勾配降下法により収束していく様子  
[](https://thinkit.co.jp/sites/default/files/article_node/Zu2-6.png)

以上では、説明をわかりやすくするために特徴量が 1 つの場合（重みが*w*0、*w*1の2つ）で説明してきました。本書の2.5節では特徴量が2つの場合（重みが*w*0、*w*1、*w*2の3つ）で説明されています。基本的な考え方は同じなので、本書を読み進めるうえでの参考になれば幸いです。