

仕事ではじめる 機械学習

有賀 康頭
中山 心太 著
西林 孝

Copyright © 2017 Michiaki Ariga, Shinta Nakayama, Takashi Nishibayashi. All rights reserved.

本書で使用するシステム名、製品名は、いずれも各社の商標、または登録商標です。なお、本文中では™、®、©マークは省略している場合もあります。

本書の内容について、株式会社オライリー・ジャパンは最大限の努力をもって正確を期していますが、本書の内容に基づく運用結果については責任を負いかねますので、ご了承ください。

まえがき

機械学習という言葉は、ソフトウェアエンジニアの間でも、最近では聞かない日がないほど一般的な言葉になりました。より世の中を広く見渡してみると、「人工知能が職を奪う」とか、「ディープラーニングを知らないと駄目なのではないか」という声も少なからず聞こえるようになりました。これらのことには、コンピュータが圍碁のプロ棋士を破り、人工知能に対して大きな期待が寄せられたという背景があるでしょう。また、機械学習の面からは、多くのデータが手に入り、それを処理をするハードウェアの進化も進み、更にはオープンソースで最新のアルゴリズムを利用できる、便利なフレームワークやライブラリが広く普及したことも強く影響しているでしょう。

こうした機械学習に対する期待の高まりとともに、私たちのところへ「機械学習について教えてください」と聞きにくる人が増えてきました。幸いにも、多くの研究者によってアルゴリズムや理論に関する素晴らしい本が書かれており、機械学習フレームワークの使い方や実装方法についての書籍や雑誌記事も多く世に広まるようになりました。これにより、機械学習を知らなかったソフトウェアエンジニアでも、機械学習に取り組むハードルは以前に比べ劇的に低くなっているといえるでしょう。

最近では、情報系の学生は大学の講義や研究で機械学習の理論を学んだり研究対象としており、そうした学生が卒業しソフトウェアエンジニアとして働くことも増えてきました。彼らは理論的なバックグラウンドを活かしながら、機械学習エンジニアなどと呼ばれ研究開発を推し進めています。

しかし、Couseraなどのオンラインコースや書籍・大学の研究だけでは、機械学習の基礎や理論的背景を学ぶことはできても、実際のビジネスにどのように活かせば良いのか、ビジネスにおける機械学習やデータ分析を活かすにはどうしたら良いのかは自明ではありません。問題設計をどのようにすれば、システムはどのように設計すれば良いのかを座学で学ぶことはなかなか難しいと言わざるを得ません。

本書の扱っている内容

本書では、以下のような読者を対象とし、機械学習やデータ分析の道具をどのようにビジネスに生

かしていいのか、また不確実性が高いと言われている機械学習プロジェクトの進め方について整理しています。

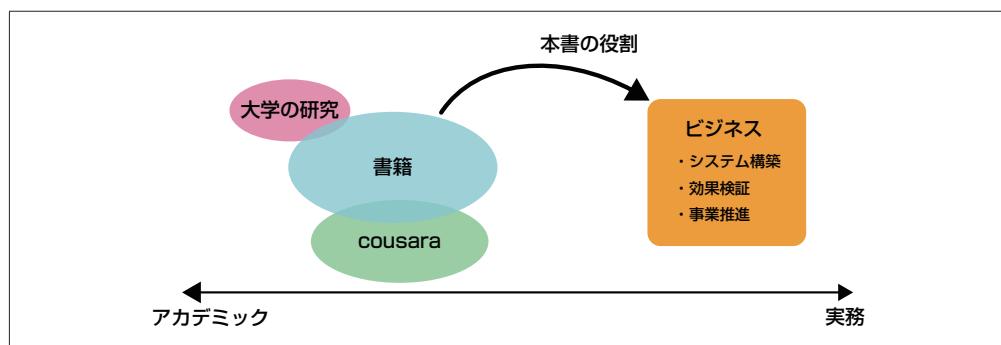
- 機械学習の入門教材は終えて、実務に活かしたいエンジニア
- 大学の講義などで機械学習を学んだ経験を、プロダクトに活かしたい若手エンジニア

より具体的には、以下のような内容を取り上げています。

- 機械学習のプロジェクトをどのようにはじめるか
- 機械学習と既存システムをどう連携するのか
- 機械学習のデータをどのように集めるのか
- 仮説をどのように立てて分析を進めるのか

本書はもともと、機械学習の初学者向けに書いた文章からはじめました。入門者のために書きはじめたのですが、実際には理論を軽めにしたソフトウェアエンジニア向けの実践的なカタログのような形になっています。

アルゴリズムの話などは他の書籍でも数多く取り上げられているので、本書ではプロジェクトのはじめ方や、システム構成、学習のためのリソースの収集方法など、読者が「実際どうするの？」と気になるであろう点を中心しています。



本書で扱っていない内容

一方で、本書のような方は本書の対象読者として想定していません。

- 機械学習の研究者
- 機械学習の理論が学びたい人
- scikit-learn や TensorFlow といった機械学習のフレームワークの使い方を学びたい人
- 機械学習のフレームワーク自体を実装したい人

そのため、以下のような内容は扱いません。

- 機械学習の理論やアルゴリズム、特にディープラーニングについて
- プログラムの基礎
- 微分積分、行列計算、確率計算など高校卒業程度の数学

本書には、「人工知能でいい感じの成果を出してくれ」と言う上司をコントロールする方法が直接的に書かれているわけではありません。しかし本書を読むことで、どういう道筋でこのような要求に対応すれば良いのかについて、一歩引いた考え方を獲得していただけたら嬉しいです。

本書が前提としている内容

本書では、一部の章を除いてあまり数式を使わないようにしていますが、最低限の数学と機械学習の基本を知っていることが望ましいです。CourseraのMachine Learningのオンラインコース^{†1}を受講したり、オライリー・ジャパンの書籍『ゼロから作るディープラーニング』[dlscratch]を読んだ次の二冊目として活用いただければと思います。

また、本書ではPythonとscikit-learnを主に使ったコードを使って説明します。本文中ではPythonやscikit-learn、Jupyter Notebookの基礎的な使い方は説明しません。詳細が知りたい場合は、scikit-learnのドキュメント^{†2}を参照いただくか、オライリー・ジャパン『Pythonではじめる機械学習』[sklearn_ml_book]、インプレス『Python機械学習プログラミング』[python_ml]、技術評論社『PythonユーザのためのJupyter[実践]入門』[jupyter_book]などの書籍を手に取っていただくと良いでしょう。

本書の構成

本書では、機械学習プロジェクトを進めていくために必要な知識をまとめた第I部と、実際に手を動かすケーススタディの第II部の2部構成となっています。

第I部では実務で機械学習を使う上で必要な基本的な知識として、まず第1章では機械学習プロジェクトをどのような流れで進めていくかについて整理します。機械学習の初歩のおさらいと、機械学習を含むコンピュータシステム特有の難しさについて説明します。

第2章では、機械学習でできることと各種アルゴリズムについて紹介します。機械学習のどういったアルゴリズムがどういう特徴を持っているかをカタログ的に並べています。馴染みの薄い方は、どのアルゴリズムを使えば良いのかというフローチャートと、どのように分類されるのかという決定境界をカタログ的に見ると良いでしょう。

^{†1} <https://www.coursera.org/learn/machine-learning>

^{†2} <http://scikit-learn.org/stable/documentation.html>

第3章は、学習を行ったときにオンラインで予測モデルをどのように評価するかという方法について、スパム判定を例に説明しています。

第4章は、コンピュータシステムに機械学習の仕組みをどのように組み込むのかのパターンを整理します。合わせて、学習の元手となるログ設計について説明します。

第5章は、機械学習の分類タスクにおける正解データの収集の仕方について説明をします。

第6章は、西林さんに寄稿してもらい、導入した施策が本当に効果があったのかを検証するために、統計的検定、因果効果推論、そしてA/Bテストについて紹介をしています。第3章では予測モデルのオンラインでの検証でしたが、本章では実際に導入しながらどのように検証するかを解説します。とても大切な内容ですが、第I部の他の章とくらべて必要な数学や統計の前提知識が多いため、難しいを感じたらいったん飛ばしてしまって、後から読むのが良いでしょう。

より実践的な話題の第II部では、第7章で映画の推薦を例に推薦の予測システムの作り方を学びます。

続く8章と9章は中山さんに寄稿してもらった章で、まず第8章では探索的な分析の過程とそれを元にしたレポートをお届けします。第1章の機械学習の流れで出てきた「機械学習をしない例」の1つとして、また実際に分析を行った結果をどのようにまとめるのかについての知見を得られるでしょう。

第9章では、Uplift Modelingと呼ばれる方法を用いて、より効果的にマーケティングを行う方法を学びます。

また、本書のコードのJupyter Notebookは、サポートレポジトリ <https://github.com/oreilly-japan/ml-at-work>にて参照できます。

機械学習もデータを活用するアプローチの1つです。闇雲に機械学習を行うのではなく、問題を解決するための道具として、地に足の着いた考え方方が進められる方法がこの本を通じて得られれば何よりです。

意見と質問

本書の内容については、最大限の努力をもって検証、確認していますが、誤りや不正確な点、誤解や混乱を招くような表現、単純な誤植などに気がつかれることもあるかもしれません。そうした場合、今後の版で改善できるようお知らせいただければ幸いです。将来の改訂に関する提案なども歓迎いたします。連絡先は次の通りです。

株式会社オライリー・ジャパン

電子メール japan@oreilly.co.jp

本書のWebページには次のアドレスでアクセスできます。

<https://www.oreilly.co.jp/books/9784873118215/>

オライリーに関するそのほかの情報については、次のオライリーのWebサイトを参照してください。

<https://www.oreilly.co.jp/>

<https://www.oreilly.com/> (英語)

表記上のルール

本書では、次に示す表記上のルールに従います。

太字 (Bold)

新しい用語、強調やキーワードフレーズを表します。

等幅 (Constant Width)

プログラムのコード、コマンド、配列、要素、文、オプション、スイッチ、変数、属性、キー、関数、型、クラス、名前空間、メソッド、モジュール、プロパティ、パラメーター、値、オブジェクト、イベント、イベントハンドラ、XMLタグ、HTMLタグ、マクロ、ファイルの内容、コマンドからの出力を表します。その断片（変数、関数、キーワードなど）を本文中から参照する場合にも使われます。

等幅太字 (Constant Width Bold)

ユーザーが入力するコマンドやテキストを表します。コードを強調する場合にも使われます。



ヒントや示唆、興味深い事柄に関する補足を表します。



ライブラリのバグやしばしば発生する問題などのような、注意あるいは警告を表します。

サンプルコードの使用について

本書の目的は、読者の仕事を助けることです。一般に、本書に掲載しているコードは読者のプログラムやドキュメントに使用してかまいません。コードの大部分を転載する場合を除き、我々に許可を求める必要はありません。例えば、本書のコードの一部を使用するプログラムを作成するために、許可を求める必要はありません。なお、オライリー・ジャパンから出版されている書籍のサンプルコードをCD-ROMとして販売したり配布したりする場合には、そのための許可が必要です。本書や本書のサンプルコードを引用して質問などに答える場合、許可を求める必要はありません。ただし、本書のサンプルコードのかなりの部分を製品マニュアルに転載するような場合には、そのための許可が必要です。

出典を明記する必要はありませんが、そうしていただければ感謝します。有賀康顕、中山心太、西林孝 著『仕事ではじめる機械学習』(オライリー・ジャパン発行、ISBN978-4-87311-821-5) のように、タイトル、著者、出版社、ISBNなどを記載してください。

サンプルコードの使用について、公正な使用の範囲を超えると思われる場合、または上記で許可している範囲を超えると感じる場合は、japan@oreilly.co.jpまでご連絡ください。

目次

まえがき	iii
------------	-----

第1部 1

1章 機械学習プロジェクトのはじめ方	3
1.1 機械学習はどのように使われるのか	3
1.2 機械学習プロジェクトの流れ	4
1.2.1 問題を定式化する	5
1.2.2 機械学習をしなくて良い方法を考える	6
1.2.3 システム設計を考える	8
1.2.4 アルゴリズムを選定する	9
1.2.5 特微量、教師データとログの設計をする	10
1.2.6 前処理をする	11
1.2.7 学習・パラメータチューニング	12
1.2.8 システムに組み込む	14
1.3 実システムにおける機械学習の問題点への対処方法	15
1.3.1 人手でゴールドスタンダードを用意して、予測性能のモニタリングをする	15
1.3.2 予測モデルをモジュール化をしてアルゴリズムのA/Bテストができるようにする	16
1.3.3 モデルのバージョン管理をして、いつでも切り戻し可能にする	16
1.3.4 データ処理のパイプラインごと保存する	17
1.3.5 開発/本番環境の言語/フレームワークは揃える	17
1.4 機械学習を含めたシステムを成功させるには	19
1.5 この章のまとめ	20

2章 機械学習で何ができる？	21
2.1 どのアルゴリズムを選ぶべきか？	21
2.2 分類	23
2.2.1 パーセプトロン	24
2.2.2 ロジスティック回帰	31
2.2.3 SVM	37
2.2.4 ニューラルネットワーク	41
2.2.5 k-NN	44
2.2.6 決定木、ランダムフォレスト、GBDT	46
2.3 回帰	50
2.3.1 線形回帰の仕組み	50
2.4 クラスタリング・次元削減	52
2.4.1 クラスタリング	52
2.4.2 次元削減	53
2.5 その他	53
2.5.1 推薦	53
2.5.2 異常検知	53
2.5.3 頻出パターンマイニング	54
2.5.4 強化学習	54
2.6 この章のまとめ	55
3章 学習結果を評価しよう	57
3.1 分類の評価	57
3.1.1 正解率を使えば良いのか？	57
3.1.2 データ数の偏りを考慮する適合率と再現率	58
3.1.3 F値でバランスの良い性能を見る	59
3.1.4 混同行列を知る	59
3.1.5 多クラス分類の平均のとり方：マイクロ平均、マクロ平均	63
3.1.6 分類モデルを比較する	63
3.2 回帰の評価	64
3.2.1 平均二乗誤差	64
3.2.2 決定係数	65
3.3 機械学習を組み込んだシステムのA/Bテスト	66
3.4 この章のまとめ	67
4章 システムに機械学習を組み込む	69
4.1 システムに機械学習を含める流れ	69

4.2	システム設計.....	69
4.2.1	混乱しやすい「バッチ処理」と「バッチ学習」.....	70
4.2.2	バッチ処理で学習+予測結果をWebアプリケーションで直接算出する (リアルタイム処理で予測).....	71
4.2.3	バッチ処理で学習+予測結果をAPI経由で利用する(リアルタイム処理で予測).....	74
4.2.4	バッチ処理で学習+予測結果をDB経由で利用する(バッチ処理で予測).....	75
4.2.5	リアルタイム処理で学習をする.....	78
4.2.6	各パターンのまとめ.....	78
4.3	ログ設計	80
4.3.1	特微量や教師データに使いうる情報	80
4.3.2	ログを保持する場所.....	81
4.3.3	ログを設計する上での注意点.....	82
4.4	この章のまとめ	84

5章 学習のためのリソースを収集しよう 85

5.1	学習のためのリソースの取得方法	85
5.2	公開されたデータセットやモデルを活用する	86
5.3	開発者自身が教師データを作る	88
5.4	同僚や友人などにデータ入力してもらう	89
5.5	クラウドソーシングを活用する	90
5.6	サービスに組み込み、ユーザに入力してもらう	91
5.7	この章のまとめ	91

6章 効果検証 93

6.1	効果検証の概要	93
6.1.1	効果検証までの道程	93
6.1.2	オンラインで検証しにくいポイント	94
6.2	仮説検定の枠組み	96
6.2.1	コインは歪んでいるか	96
6.2.2	二群の母比率の差の検定	98
6.2.3	偽陽性と偽陰性	100
6.3	仮説検定の注意点	101
6.3.1	繰り返し検定をしてしまう	101
6.3.2	有意差とビジネスインパクト	102
6.3.3	複数の検定を同時に使う	104
6.4	因果効果の推定	105
6.4.1	ルービンの因果モデル	105

6.4.2	セレクションバイアス	107
6.4.3	ランダム化比較試験	107
6.4.4	過去との比較は難しい	108
6.5	A/Bテスト	109
6.5.1	2群の抽出と標本サイズ	110
6.5.2	A/Aテストによる均質さの確認	110
6.5.3	A/Bテストの仕組み作り	110
6.5.4	テストの終了	111
6.6	この章のまとめ	111

第II部**113****7章 映画の推薦システムをつくる 115**

7.1	シナリオ	115
7.1.1	推薦システムとは	115
7.1.2	応用シーン	116
7.2	推薦システムをもっと知ろう	117
7.2.1	データの設計と取得	117
7.2.2	明示的データと暗黙的データ	118
7.2.3	推薦システムのアルゴリズム	119
7.2.4	ユーザー間型協調フィルタリング	119
7.2.5	アイテム間型協調フィルタリング	121
7.2.6	モデルベース協調フィルタリング	122
7.2.7	内容ベースフィルタリング	123
7.2.8	協調フィルタリングと内容ベースフィルタリングの得手・不得手	123
7.2.9	評価尺度	123
7.3	MovieLensのデータの傾向を見る	124
7.4	推薦システムの実装	129
7.4.1	Factorization Machineを使った推薦	129
7.4.2	いよいよ Factorizatoin Machineで学習する	132
7.4.3	ユーザーと映画以外のコンテキストも加える	135
7.5	この章のまとめ	138

8章 Kickstarterの分析、機械学習を使わないという選択肢 139

8.1	KickstarterのAPIを調査する	139
8.2	Kickstarterのクローラを作成する	140
8.3	JSONデータをCSVに変換する	141

8.4	Excelで軽く眺めてみる.....	142
8.5	ピボットテーブルでいろいろと眺めてみる.....	145
8.6	達成したのにキャンセルされたプロジェクトを見てみる.....	151
8.7	国別に見てみる.....	153
8.8	レポートを作る.....	154
8.9	今後行いたいこと.....	164
8.10	おわりに.....	166
9章	Uplift Modelingによるマーケティング資源の効率化.....	167
9.1	Uplift Modelingの四象限のセグメント	168
9.2	A/Bテストの拡張を通じたUplift Modelingの概要	169
9.3	Uplift Modelingのためのデータセット生成	170
9.4	2つの予測モデルを利用したUplift Modeling	172
9.5	Uplift Modellingの評価方法、AUUC	176
9.6	実践的な問題での活用	181
9.7	Uplift Modelingを本番投入するには.....	187
9.8	この章のまとめ	189
	参考文献	191
	あとがき	195

第I部

1章から6章までは、機械学習の初学者向けの解説です。入門のために書きはじめたのですが、実際には理論を軽めにしたソフトウェアエンジニア向けの実践的なカタログのような形になりました。

アルゴリズムの話などは他の書籍でも数多く取り上げられているので、本書ではプロジェクトのはじめ方や、システム構成、学習のためのリソースの収集方法など、読者が「実際どうするの？」と気になるであろう点を中心に書いています。

数式はできるだけ書かないようにしたつもりですが、2章など、はじめて機械学習に触れる人には少し難しい内容かもしれません。そういう時には、最初のうちはカタログのようにざっと眺めて、章冒頭のアルゴリズムの使い所のフローチャートを見ていただくのが良いと思います。CourseraのMachine Learning^{†1}のコースを受けたり、『ゼロから作るDeep Learning』[dlscratch]のような書籍を読んだりすると、更に理解しやすくなると思います。

また、本書ではPythonの機械学習ライブラリscikit-learn^{†2}を使うことを前提に話を進めます。

^{†1} <https://www.coursera.org/learn/machine-learning>

^{†2} <http://scikit-learn.org/stable/>

1章

機械学習プロジェクトのはじめ方

本章では、機械学習プロジェクトのはじめ方についてまとめます。

機械学習のプロジェクトは、普通のコンピュータシステムの開発に比べて予測精度を求めるなど試行錯誤をすることが多く、手戻りが発生しやすいため、ポイントを押さえて進めることが重要です。まずは機械学習の概要から、プロジェクトの流れ、機械学習特有の問題、成功させるためのチームづくりについて紹介します。

1.1 機械学習はどのように使われるのか

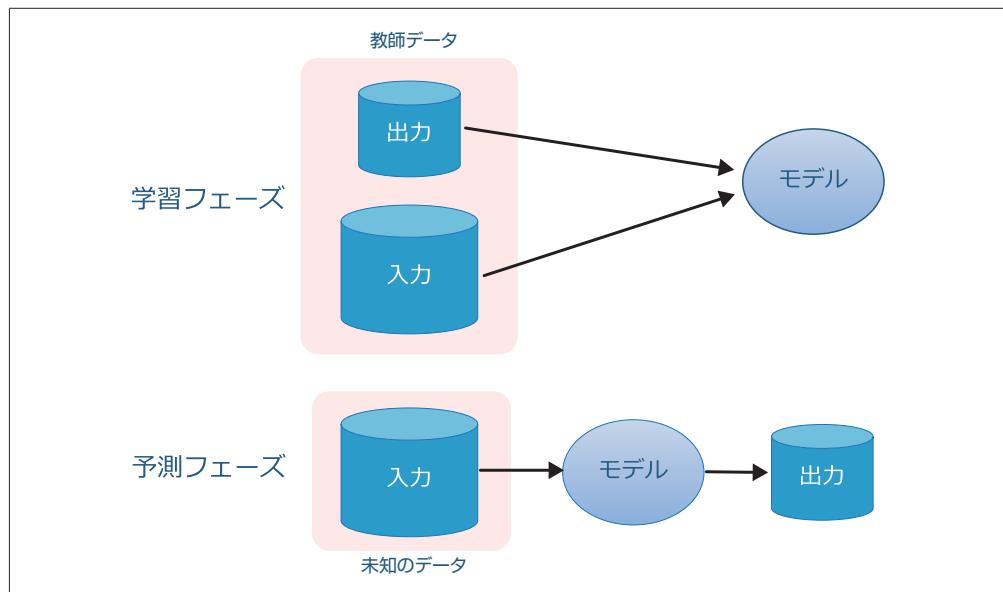


図1-1 機械学習(教師あり学習)の概要

はじめに、機械学習とはどのように使われるものかについて簡単におさらいしておきましょう。

機械学習が良く使われる用途として、未知のデータに対して過去の経験（＝過去のデータ）を元に機械が予測をする、というものがあります。例えばGmailの「迷惑メール」の判定やAmazonの「この商品を買った人はこんな商品も買っています」の推薦など、過去の膨大なデータの傾向を元に、未知のデータに対して予測をします。

このように未知のデータに対して予測を行う場合、ビジネスで頻繁に使われるのが**教師あり学習 (Supervised Learning)** というアプローチです。図1-1に、教師あり学習の処理の概要をあらわしています。教師あり学習は、大雑把に言うと既知のデータと何かしらのアルゴリズムを用いて、入力データ（画像のRGBや日時、気温などの数値をベクトル化したもの）と出力データ（「犬」「猫」などのカテゴリや降水量などの数値）の関係性を獲得し（これを**モデル**と呼びます）、獲得したモデルによって未知のデータに対する予測ができるようになるプログラムを実現します。



モデルとは入力データと出力データの見えない関係性を、数式やルールなどの簡単な仕組みで近似したものです。学習されたモデルは、どのアルゴリズムを使うかという情報と、データから獲得されたパラメータとで構成されています。

教師あり学習には、既知のデータの入力と出力の関係性を得る**学習フェーズ**、未知の入力データからモデルを通じて予測した出力を得る**予測フェーズ**の2つのフェーズがあります。モデルを獲得する、すなわち入出力の関係性を表現するパラメータを獲得することで、未知のデータに対しても一定の基準で予測を行えます。人間は、例えば徹夜で連続して単純作業をしていると、その判断基準がぶれてくることが往々にしてあります。それに対し、機械学習の予測基準は学習フェーズからブレることはないので、大量のデータに対しても人より安定した予測を行えるのです。

ちなみに、教師あり学習の他には入力データからデータの構造を獲得する**教師なし学習 (Unsupervised Learning)** や、囲碁や将棋などでどのように行動をしていくかという戦略を獲得する**強化学習 (Reinforcement Learning)** など他の学習のアプローチもあります（詳しくは2章で紹介します）。

1.2 機械学習プロジェクトの流れ

実際の機械学習を含めたプロジェクトを開始する際には、以下のような流れで進めていきます。

1. 問題を定式化する
2. 機械学習をしないで良い方法を考える
3. システム設計を考える
4. アルゴリズムを選定する

5. 特微量、教師データとログの設計をする
6. 前処理をする
7. 学習・パラメータチューニング
8. システムに組み込む

この流れをおおまかに言うと、「解きたい課題を機械学習で解ける問題設定に落とし込む」(1,2)、「解くための道具選びと前処理」(3~6)、「モデルの作成」(7)、「サービスへの組込み」(8) という4ステップになっています。特に最初の2つの課題設定と前処理が重要です。いくらデータ量があっても、適切に前処理がなされなければ性能は出ませんし、そもそも解こうと思っている課題が人間にも解けない問題の場合は機械学習で解くのも難しいでしょう。機械学習、特に「教師あり学習」では、何が「正解」なのかを人間が機械に教えないかもしれません。つまり、人が「正解」を決められない問題は機械にも解くことができないのです。

まずは、「機械学習でこういう課題を解決した」という事例を見た時に、「どういったアルゴリズムで解決したか」「どのようなデータを特微量として使っているか」「機械学習部分をどのように組み込んでいるか」ということを意識して調べると良いでしょう（特微量という単語については後ほど詳しく説明しますが、ここでは入力情報という意味でとらえてください）。こうして引き出しを増やすことで、何ができるかが何ができるかを判断できるようになります。

こうした引き出しがないと、「機械学習でなにかすごいことをしたい」という上司が現れて、何をすればいいかというところから頭をひねるようなプロジェクトになってしまい、うまくいかない結果に終わる危険性が高まります。大事なことは、機械学習で解ける範囲と解けない範囲を線引きできること、そしてそれを実現するために（泥臭いことも含めて）手を動かせることです。「データ分析業務は前処理が8割」と言われることもあるくらい、フォーマットが崩れているCSVのパースや、Webのログから必要な情報を抽出するなど、分析可能な状態にするまでの時間が大きな割合を占めます。

機械学習をシステムの一部として含むシステム開発は、実際には試行錯誤を繰り返す作業になります。特に、上記の4から7は試してみては変更して、という作業を何度も繰り返すことになると思います。機械学習の理論的な研究では4と7を重点的に取り扱う一方で、ビジネスでは1から8まで全てを行います。そのため、試行錯誤を効率よく行うことが重要です。

では、それぞれの手順について順番に説明をしていきます。

1.2.1 問題を定式化する

次に一般的な課題を解くのと同じように、どのように問題を解くのか定式化します。その時重要なのが、何を目的にするのか、そして解きたい課題について仮説を立て、何をすればよいのかを明確にすることです。そもそも、何かシステムを作るときには、「売上を改善する」「有料会員を増やす」「製品の生産コストを減らす」など、ビジネス上の目的があるかと思います。こうした大きな目的に対して、

例えば「生産コストを減らす」ためには「歩留まり改善」をしなければならず、そのためには「どこで不良が起こっているかを特定する、そのために機械学習の力を使う」というように、より具体的でアクション可能なレベルまでブレイクダウンしていきましょう。

ブレイクダウンをする課程で、売上目標や日毎の有料会員増加数など何かしらビジネス上の指標、いわゆる **KPI (Key Performance Indicator)** が決まってくるはずです。これは、予測モデルの性能とは別の軸で重要な指標です。KPI自身はフェーズによって変わりうるものではありますが、最初のブレイクダウンをしたときに仮でも良いので1つ決めると良いでしょう。なお、KPIの決め方や目的と課題の考え方や整理の仕方は『Running Lean』[runninglean]や『Lean Analytics』[leananalytics]などの書籍を読むと良いでしょう。

機械学習の問題設定としては、例えば、「ECサイトの売上を上げるために、ユーザー毎におすすめ商品を提示する」とか、「工場の電力消費量を最適化するために、消費電力を予測する」などというように、プロジェクトの目的と解き方をセットで考えます。良くない機械学習の問題設定の例としては、「有料会員を増やしたい」といったアクションがすぐ起こせない曖昧なものや、「深層学習で凄いことをする」といった目的も分からぬようなものです。もしかすると、トップダウンでこういった要求が降ってくるかもしれません、現場で手を動かす人はもっとブレイクダウンをしなければいけません。

仮説の立て方と検証方法については、クックパッド開発者ブログに掲載されている「仮説検証とサンプルサイズの基礎」^{†1}という記事が役に立つでしょう。

1.2.2 機械学習をしなくて良い方法を考える

次に、機械学習を本当に使うべきなのかを考えます。機械学習プロジェクトのはじめ方なのに、なぜ?と思うかもしれません、「機械学習は技術的負債の高利貸しのクレジットカード」というタイトルの論文[dsculley]があるほど、機械学習を含んだシステムは通常のシステム以上に技術的負債^{†2}が蓄積しやすいのです。

機械学習を用いるシステム構築の難しさには以下のようなものがあります。

- 確率的な処理があるため自動テストがしにくい
- 長期運用しているとトレンドの変化などで入力の傾向が変化する
- 処理のパイプラインが複雑になる
- データの依存関係が複雑になる
- 実験コードやパラメータが残りやすい
- 開発と本番の言語/フレームワークがバラバラになりやすい

†1 <http://techlife.cookpad.com/entry/2016/09/26/111601>

†2 技術的負債とは、ドキュメントやテストコードがない、行き当たりばったりの設計が残っている、コンパイラの警告が残っているなど、リリースを優先して問題を先送りにすることです。詳細は<https://ja.wikipedia.org/wiki/%E6%8A%80%E8%A1%93%E7%9A%84%E8%B2%A0%E5%82%B5>を参照ください。

これらの難しさの対処方法は、後ほど「1.3 実システムにおける機械学習の問題点への対処方法」で書きたいと思いますが、特に「入力データの傾向が変化する」という点は大きな問題です。例えば、テキストを扱う問題として文のポジネガ判定を行う際に、文章に含まれる「ヤバい」という単語について、昔の文章ではネガティブな意味合いしかなかったのに対し、比較的新しい文章ではポジティブな意味もありうる、といった具合に、テキストの意味が変化する場合があります。これ以外にも、「スイカ」といえば果物のスイカの意味だったのが、電子マネーを指すようになったりと、用法のトレンドが変化したり、新語が登場したりします。

このように、同じ予測モデルを使い続けていると、入力データの傾向やトレンドが変化してしまった場合、予測精度が下がったり意図しない挙動をする可能性があります。これを防ぐためには、定期的に新しいデータで予測モデルを更新したり、必要に応じて特徴量の再検討をする必要があります。つまり、「予測モデルをメンテナンス」し続ける必要があります。

また、機械学習アルゴリズム内には乱数を用いた確率的な処理が含まれていることも多く、ルールベースで処理をした時のように挙動が決定論的ではありません。さらには、あらゆるデータに対する挙動を予め確認することは不可能です。それができないような大量の自動処理を期待しているので機械学習を利用しているのだと思います。そのため、思いもよらない予測結果が出力されるリスクが常に存在するという認識が必要です。以前、Google フォトが写真に写ったアフリカ系の人をゴリラと認識してしまい、差別的であると問題になったことがありました^{†3}。この時には「ゴリラ」というタグが出力されないような後処理を追加したようですが、このように思わぬ出力がされる可能性があります。意図しない予測結果が出てしまったときに、後から介入できる仕組み（例えば特定のラベルはブラックリストに登録しておく）を用意しておくことが重要です。

では、どのようなビジネス上の課題に対して機械学習を利用すれば良いのでしょうか？私は以下の条件を満たす必要があると考えます。

- 大量のデータに対して、高速に安定して判断を求める必要がある
- 予測結果には、一定数の間違いが含まれることが許容できる

機械学習による予測は、人間が疲れによって判断がぶれてしまうとの異なり、大量のデータに対しても同じ基準で常に判断し続けることができます。また、予測結果が100%ということはまずありません。そのため運用上、誤りをカバーできる仕組みがなければなりません。

条件が整った上で、まずMVP (Minimum Viable Product) を作りましょう。MVPとは、『リーンスタートアップ』[leanstartup]の文脈で良く話題にされるもので、最低限の顧客価値を生み出す最小のプロダクトのことです。では、機械学習でのMVPとは何でしょうか。例えば、男女や年齢などのユーザーの属性によるクロス集計でセグメント分けをして、そのセグメント毎にルールベースでレコメンドをして

^{†3} <https://www.theguardian.com/technology/2015/jul/01/google-sorry-racist-auto-tag-photo-app>

はどうか？Apache SolrやElasticsearchなど、既存のモジュールにあるMore Like Thisなどの機能の組み合わせではダメなのか？といったように、集計ベースや既存のモジュールの機能で簡単に実装できるものがMVPとなりえます。もちろん、MVPの原則に則って人手で決め打ちのコンテンツを用意をして、簡単なルールで振り分けるのでも十分でしょう。筆者も経験がありますが、場合にもよりますがこのMVPでも十分機能するということが往々にして起きるのです。

MVPを検証することで、そもそも自分が立てた仮説の筋が良いか悪いかを判断できます。こういった機械学習の問題設定から仮説検証までのサイクルは、多くの場合で一般的なコンピュータシステムのプロダクトよりも長くなるようです。そもそも狙い自体が間違っている場合には、システム構築と実験まで終えた後で、最初の問題設定まで大きく手戻りすることさえあります。本当に顧客が必要だったものは何か、コンセプトは正しいのかを事前に検証することが、通常のプロダクトより重要になってきます。

プロジェクトのはじめに機械学習をやろう、と始まったプロジェクトでも、必要がないのであれば機械学習を使わない方向にかじを切ることを恐れないでください。

このように、機械学習に向いている課題であることを確認し、MVPを作つて最低限のコンセプトを検証することで、システムの設計へと進みます。

1.2.3 システム設計を考える

問題の定式化とMVPの検証が済んだら、機械学習を含めたシステムを設計しましょう。設計の上で重要なポイントは2つあります。

- 予測結果をどういう形で利用するのか
- 予測誤りをどこで吸収するのか

1つ目は、例えばバッチで予測処理をしてその結果をRDB（Relational Database）で配布するのか、Webサービスやアプリケーションを用いてユーザーのアクション毎に非同期で予測するのかなど、方法の違いがあります。予測結果をどう渡すかについて、詳しくは「4章 システムに機械学習を組み込む」を参照してください。

機械学習に100%正解を出力するアルゴリズムは存在しません。システム全体でどのように誤りをカバーするのか、人手による確認や修正は必要なのか、必要だとしたらどこでカバーするのかを考えることも機械学習のシステム設計に含まれます。それを理解した上で、どのようにシステム全体でリスクをコントロールするかが重要です。例えば予測結果を人手で確認するフェーズを用意したり、予測結果が重大な悪影響を与えないといふかっていたりする場所でアプリケーションに利用するなどの方法を用います。

また、このフェーズを過ぎると実際に手を動かしてデータ収集や予測モデルの作成、システム構築を進めるフェーズに入る所以、ここまでに目標性能と撤退ラインを決めましょう。機械学習の予測モ

ル開発は、往々にして「あと少し性能が良くなったら…」と、予測モデル自身を改善する泥沼にハマリこみます。そこまでに、学習データの収集や正解データの作成などといった作業をしてしまうと、一定のドメイン知識が付いてくるため、根拠のない自信で改善ができると思いこんでしまいます。場合によっては、問題設定の見直しまで戻って繰り返し改善し続ける可能性もあります。サンクコストによるバイアスがかかる前に、「2ヶ月で90%の予測性能を達成する」といった、具体的な目標性能と撤退ラインを決めておきましょう。なお、予測性能の決め方については「3章 学習結果を評価しよう」で解説します。

1.2.4 アルゴリズムを選定する

機械学習を使うときに考えなければならない、採用するアルゴリズムについて考えます。



アルゴリズムの選定の指針は「2章 機械学習で何ができる？」も参考にしてください。

過去に類似の問題がどのように解かれているかを調べると、おおよそのあたりは付けられます。データの特性が分からなければ、クラスタリングなどの教師なし学習（2章で説明します）や散布図行列（図1-2）などを使って事前に可視化してあたりをつけ、どういった方法で解けるのかを考えます。

また、想定されるデータ量を含めてオンライン学習が良いのかバッチ学習（「4.2.1 混乱しやすい「バッチ処理」と「バッチ学習」」で説明します）でも十分なのかを見積もります。

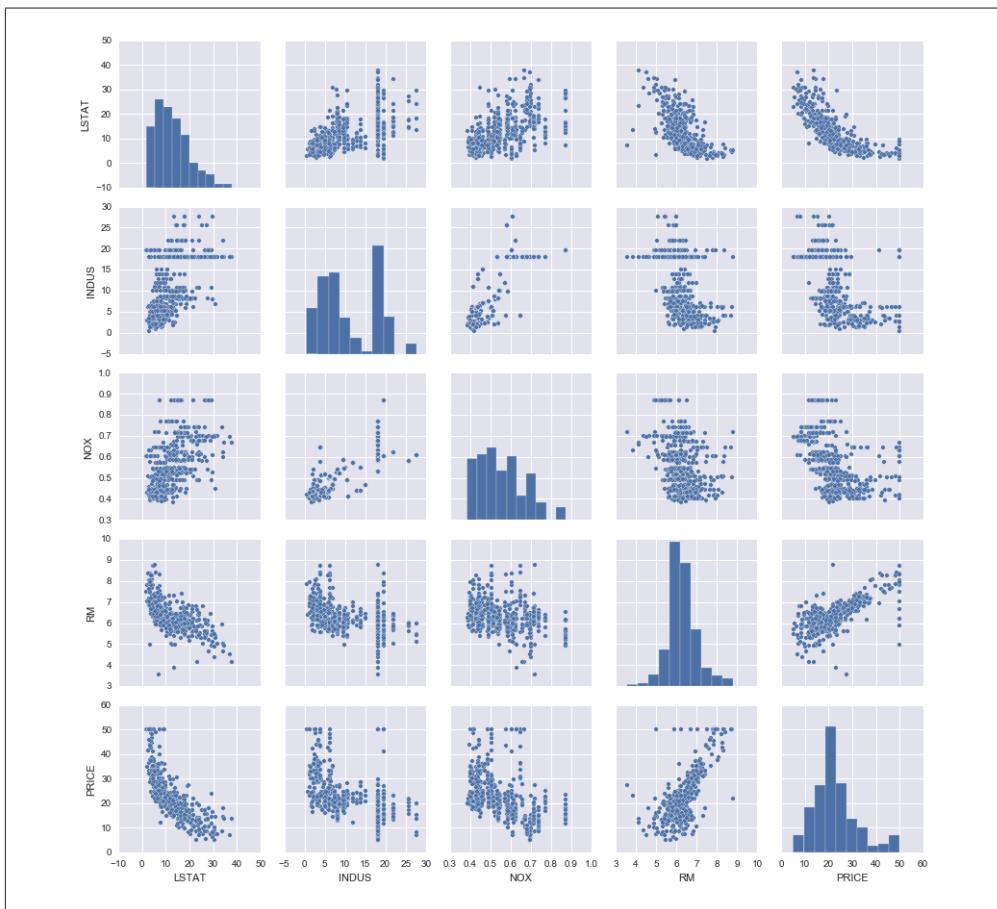


図 1-2 散布図行列の例

1.2.5 特徴量、教師データとログの設計をする

アルゴリズムの候補を見繕ったら、どういった情報が使えるかについて設計をします。

特徴量 (Feature)^{†4}とは、機械学習の予測モデルに入力をする情報のことを指します。機械学習では、入力の情報をまず数値ベクトルにします。例えば、明日の積雪の有無を予測するために、今日の気温(1.0°C)、降水量(0.8mm)、風速(0m)、積雪量(2cm)、天気(曇り)を使うとすると、それぞれを数値化したものリスト(例:[1.0, 0.8, 0.0, 2.0, 1])が特徴ベクトルになります。

ここで、「曇り」のような特徴量を**カテゴリカル変数 (Categorical Variable)**と言い、晴れは0、曇

^{†4} 「feature」という単語の訳について、自然言語処理の分野では「素性(そせい)」という表現が好まれることが多いですが、ここでは特徴量と訳します。

りは1というような数値データに変換して処理をします。このような数値データのことをダミー変数 (Dummy Variable) と呼びます。scikit-learnではLabelEncoderクラスやOneHotEncoderクラスでカテゴリカル変数をダミー変数に変換できます。

古典的な機械学習では特徴量が肝となります^{†5}。ここはビジネスドメインの知識がある人と、ユーザーの行動ログや購買履歴、工場のセンサーデータなど、機械学習の入力となる特徴量が、予測に必要な情報を含んでいることを予め確認しておきましょう。例えば、タービンの不具合を検出するためには、過去の経験を元にハンマーで叩いた音の情報を集めて不良検出をしたという事例もあります。ビジネスのドメイン知識を持った人と協力をして、何がその現象に影響を与えそうかということを確認します。後から不要なデータを削ることはできますが、必要なデータを遡って取得することはできません。

特徴量をいったん決めたら、入力データとなる正解データを用意します。教師データとは、教師あり学習と呼ばれる複数のカテゴリを予測する問題を解く際に必要な、正解カテゴリのラベル (正解ラベル) と元となるデータのセットことです。例えば画像を使った物体認識では、写真に移っている「車」や「犬」といったカテゴリの正解を、あらかじめ人手でつけておく必要があります。なお、こうした分類対象のカテゴリのことを機械学習ではクラス (class) と呼びます。プログラムのクラスとは違うので注意してください。ビジネスの多くでは、教師あり学習を使い何かしらを分類することが多いです。

教師あり学習では、質の良い正解ラベルをどのように取得するかが重要になります。この正解ラベルのクオリティ次第で問題がうまく解決できるかが大きく変わります。教師データの正解ラベル収集については5章を参考にしてください。

また、教師データのもととなるデータは、Webアプリケーションのログなどから取得することも良く行われます。特徴量を抽出するためのログの設計については、「4.3 ログ設計」で詳しく説明します。ログを設計するタイミングに合わせて、思いつく特徴量も列挙すると良いでしょう。というのも、ログの収集はいったん開始してしまうと形式を変えるコストが大きく、変えられたとしても使えないデータが増えてしまうからです。

1.2.6 前処理をする

前処理はタスクに依存するためここでは詳しくは書きませんが、不要な情報を削ぎ落とすなどデータを機械学習に使える形にする重要なプロセスです。機械学習の入力とするデータは、特徴量のところでも説明をしましたが、RDBで表現できるような表形式のデータの形になっています。しかし、実際のWebのログなどの生データはテキスト形式だったり、すぐにそのまま使えるデータではありません。数値データも、センサーデータが一部取得できていないことによる欠損値と呼ばれるデータの処理や、異常な値を除外したり、値の取りうる幅の影響を受けないように正規化したりといった作業が必要で

^{†5} 深層学習 (Deep Learning) では、例えば画像の物体認識ではRGBの値を使うなど、特徴量の設計よりもどういったネットワーク構造を使うかといったことのほうが重要です。

す。テキストデータの場合には、単語に分割して頻度を数えたり、低頻度語を除去したりという調整を行います。先ほど紹介したカテゴリ変数をダミー変数にする操作もここに含まれます。こうしたデータ変換が、前処理の最初に重要なステップとなります。

いくら優れたアルゴリズムを使っても、データを適切に整形・加工することに勝るものはありません。現実の問題では多くの時間をここに取られます。

1.2.7 学習・パラメータチューニング

いよいよ学習を行います。学習のアルゴリズムが決まっているので、機械学習のアルゴリズムを調整するパラメータを試行錯誤しながら変えてみて、より良い結果ができるパラメータを探索します。まず、人力で付与した正解やルールベースで決めた正解など、ベースラインの予測性能を決めてそれを超えることを目指します。

最初のステップとしては、ロジスティック回帰など比較的シンプルなアルゴリズムと既存のライブラリ・フレームワークを用いることで、シンプルな予測モデルを作ることをゴールとしましょう。多くの場合、一部データが正常に取れていないなどデータ自体にバグが潜んでいます。そのため問題を切り分けるために、まずはシンプルな手法で良いのでいったん予測モデルを作るようになります。

はじめての予測モデルでいきなり予測性能99.9%のように高い性能が出た場合には、何かミスがあると疑ってかかりましょう（筆者は、感覚的には初めて書いたコードのテストが全て通ってしまったときに近いもの覚えます）。多くの場合、学習時のデータに対して過剰に適合してしまい、未知のデータを適切に予測できない過学習(Overfitting)や、本来知り得るはずのない正解データの情報が教師データに紛れてモデルの予測性能が不当に高くなってしまうData Leakageが起きています。

過学習、Data Leakage

過学習したモデルは、平たく言うと「学習に使ったデータに対してはきちんと正解できるけど、知らないデータに対しては全然当たらない」というモデルになります。これは、既知のデータに対して過剰に最適化をしてしまい、未知のデータには対応できないという状態になります。昔、私がセンター試験を受けた年に、英語の突如出題傾向が変わりました。塾でバッチリ対策をしていた人が「うわー、今年傾向変わって全然解けなかったー。きっと他の人も解けなかつたよね」という話をしていたのですが、今思うと、これもある意味過学習かもしれません。逆に、未知のデータにも対応できるモデルを汎化(Generalization)性能があるモデルといいます。

Data Leakageの分かりやすい例として、Kaggleの癌予測のためのコンテストのデータに前立

腺手術をしたかどうかというフラグが含まれていたことがありました。^{†6}この情報を使った予測モデルは非常に高い予測性能を達成しましたが、前立腺がんの人がガンとわかった後に手術を受けているというだけの情報で、未知のデータに対しては意味のない予測モデルとなってしまったのです。他にも、よくある例として、時系列データの予測を行う際に学習に使うデータと検証に使うデータをランダムで分割してしまったために、未来のデータに対する予測をするモデルを作りたかったのに、未来のデータも含めて学習してしまったという話があります。

これらのポイントを意識しながら、学習とパラメータチューニングを行います。性能改善を行う場合は、誤判定をした予測結果を実際に見て、何が誤りの原因になっているか、共通項はないかとエラー分析をすることも忘れないようにしましょう。ここでうまく行かなかった場合には4に戻って、アルゴリズムの検討からやりなおします。

過学習を防ぐには

過学習を防ぐにはいくつかの方法があります。アルゴリズムによらない方法としては以下のようない方法があります。

1. 交差検証 (Cross Validation) を使ってパラメータチューニングをする
2. 正則化 (Regularization) を行う
3. 学習曲線 (Learning Curve) を見る

交差検証とは、学習の際に学習用の訓練データ (Training Data) と評価用の検証データ (Validation Data) を分割して性能を計測することで、特定のデータによらない汎化性能のあるモデルを得る方法です。例えば、データを10分割して9割の訓練データで学習、残り1割の検証データ評価を行う、というプロセスを10回繰り返すことで、平均的に性能の良いハイパーパラメータ (Hyper-parameter、ニューラルネットワークの隠れ層の数やロジスティック回帰のしきい値など、モデルの性能を左右するパラメータ) を選びます (scikit-learnにはcross_val_score() という関数やGridSearchCVというクラスなど交差検証を簡単に行う便利な道具があります)。

実際には、最初に例えば1割のデータを抜き取って最後の性能評価にのみ使うことで、ハイパーパラメータのチューニングとは独立した性能評価ができるようにします。この抜き取ったデータをテストデータ (Test Data) と呼びます。なお、本書では訓練データと検証データを合わ

^{†6} <https://www.kaggle.com/wiki/Leakage>

せたデータを開発データ (Development Data) と呼びます。

過剰適応を防いで汎化性能を上げるために正則化というテクニックを用います。簡単に言うと、2つのクラスを分離する境界を、全データ正しく分かれるようにきっちり設定するのではなく、多少違うクラスのデータが混ざって誤ってもいいから未知のデータに対する対応力を持つといふものです。詳細は「[2.2.2 ロジスティック回帰](#)」で説明します。

学習曲線とは、データサイズや学習の繰り返し回数に対して、訓練データと検証データでの損失（あるいは精度）の推移をグラフに書いたものです。損失についての詳細は2章を、学習曲線の詳細は筆者の記事「[そのモデル、過学習しているの？未学習なの？と困ったら](#)」^{†7}を参照ください。

1.2.8 システムに組み込む

おめでとうございます、良い性能の予測モデルを得ることができましたね。それでは機械学習のロジックをシステムに組みましょう。

この時気をつけたいのが、予測性能とそれに伴うビジネスインパクト（例えば商品購入のコンバージョン率など）をモニタリングすることです。システムに組み込むことは、ビジネス的には仮説検証のための1つのステップにすぎません。予測性能のモニタリングには、あらかじめ人手で用意したデータと正解ラベルのセットを使って予測性能を計測します。



このようなデータのセットをゴールドスタンダード (Gold Standard) と呼びます。

予測モデルの開発に注力していると往々にして忘れがちなのですが、本来は予測モデルを作ることで、売上目標や日毎の有料会員増加数など何かしらビジネス上の指標、いわゆるKPIの改善をしたいというような要求があるはずです。そちらの推移を見守り、必要に応じて性能改善を続けます。

機械学習は「[1.2.2 機械学習をしなくて良い方法を考える](#)」でも書いたように、長期運用をしていると入力の傾向が変わります。これにより、予測性能がじわじわと、あるいは急激に劣化する場合が多くあります。KPIが悪化した場合は、5～7に立ち戻って改善をする必要があります。これに立ち向かうためにも「システムに組み込んで終わり」ではなく、継続してビジネスに貢献しているのかを確認し、改善し続けましょう。

また、改善し続けることができる持続的な組織づくりも重要となります。KPIをきちんとトラックで

^{†7} <http://chezou.hatenablog.com/entry/2016/05/29/215739>

きるようダッシュボードを作ったり、異常時にはアラートを飛ばすようにしたり、アクションをいつでも取れるような体制が重要です。

1.3 実システムにおける機械学習の問題点への対処方法

「1.2.2 機械学習をしなくて良い方法を考える」でも書きましたが、実システムにおける機械学習の問題点には以下のようなものがあります。

1. 確率的な処理があるため自動テストがしにくい
2. 長期運用しているとトレンドの変化などで入力の傾向が変化する
3. 処理のパイプラインが複雑になる
4. データの依存関係が複雑になる
5. 実験コードやパラメータが残りやすい
6. 開発と本番の言語/フレームワークがバラバラになりやすい

まとめると、予測性能のみを追い求めてモデルの更新が難しくなり易く、システムの複雑性が上がってメンテナンス性が低下しがちであり、変化に追従しづらいと言えます。

こうした問題に対処するためには、変化を前提とした設計をするために、以下のポイントが重要なとなります（カッコ内の数字は対応している課題です）。

- 人手でゴールドスタンダードを用意して、予測性能のモニタリングをする（1、2、4）
- 予測モデルをモジュール化をしてアルゴリズムのA/Bテストができるようにする（2）
- モデルのバージョン管理をして、いつでも切り戻し可能にする（4、5）
- データ処理のパイプラインごと保存する（3、5）
- 開発/本番環境の言語/フレームワークは揃える（6）

以下では5つのポイントについて順番に説明をしていきます。

1.3.1 人手でゴールドスタンダードを用意して、予測性能のモニタリングをする

ゴールドスタンダードを用意することについては、「1.2.8 システムに組み込む」に書きましたのでそちらを参照してください。機械学習の予測結果は確率的な処理を含むため、個別の予測結果を決定論的に自動テストで検証するのは難しいです。ですので、予め用意しておいたデータと正解に対して予測性能を測定し、その推移を監視します。そうすることで、1の自動テストがしづらい問題を、予測性能のモニタリングでカバーします。また、2の入力の傾向の変化についても、予測性能をダッシュボードでモニタリングし、閾値を設けてアラートを飛ばすことで、長期運用時に傾向の変化に気づきやすく

なります。4については、例えば予測モデルの更新と単語分割用の処理の辞書の更新とを同時に行つたときに、本番環境で予測モデルだけを更新してしまい辞書の更新を忘れてしまうと言ったデータの不整合が起こります。こうした問題も、予測性能のモニタリングによりカバーできます。

1.3.2 予測モデルをモジュール化をしてアルゴリズムのA/Bテストができるようにする

予測モデルのモジュール化について説明をします。性能向上を継続するためには、1つのアルゴリズムだけでは天井が見えててしまうことがあります。そのようなときのために、複数の予測モデルを並列で並べてA/Bテストできるようにモジュール化して交換が容易な設計をしておくことが重要です。モジュール化によってモデルの比較をしやすいシステムにしておくことで、特徴量を変更したりアルゴリズムを変更したモデルを並行させて性能検証ができるのです。そうすることで、2の長期運用した時の入力の傾向の変化に対しても、現行の予測モデルを動かしながら新しい取り組みが容易になるのです。

1.3.3 モデルのバージョン管理をして、いつでも切り戻し可能にする

モデルのバージョン管理については、いつ、何の影響によって本番環境の予測モデルが性能劣化を起こすか分かりません。入力データの形式が変化したかもしれませんし、途中の処理が変わったかもしれません。モデルの更新が原因かどうかを切り分けるためにも、過去のモデルに切り戻せるようにしておくことが大切です。当然、ソースコードはバージョン管理をすると思いますが、可能であれば過去のモデルに切り戻したときにきちんと比較できるよう、データもバージョニングをしておくのが望ましいでしょう。ソースコード、モデル、データの3つの変更が管理されていること理想的な状態です。

モデルのバージョン管理をし、それに加えてどういうデータでモデルを生成したかをドキュメント化することで、4のデータの依存関係の問題は緩和されます。また、5の実験コードとアルゴリズムのパラメータがコード上に散逸する問題も、モデルのバージョン管理とドキュメント化することで取り組みやすくなるでしょう。

1.3.4 データ処理のパイプラインごと保存する

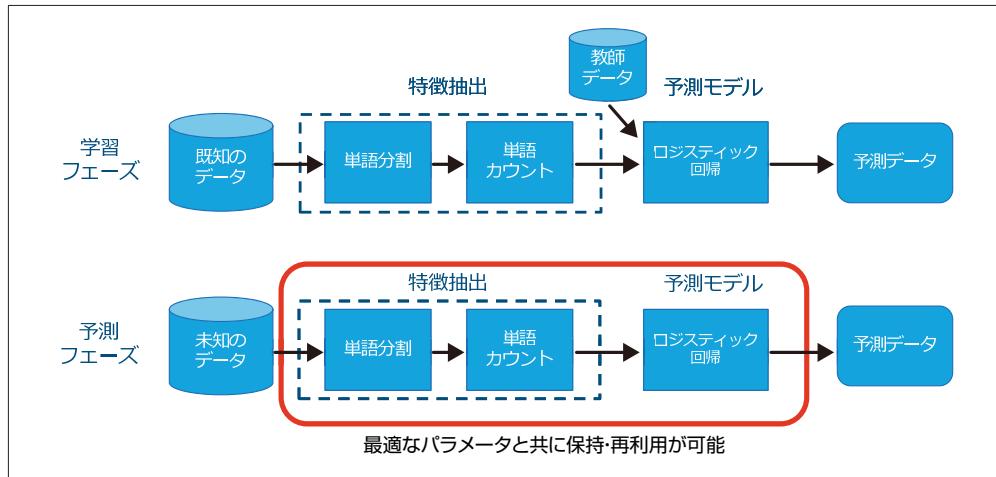


図1-3 データ処理のパイプラインを保持する

続いて、データ処理のパイプラインの保持について説明します（図1-3）。予測モデルを作成する際には、予測モデル自身のハイパーパラメータもチューニングしますが、それ以前の前処理にもパラメータが含まれることが多いです。例えば、テキスト処理では単語に分割をし、その頻度を数えた後に低頻度な単語や高頻度な単語を除外することが多くあります。こうしたときの、低頻度と足切りするしきい値は何が良いのか、といった点もチューニングポイントに含まれます。

だんだんとパラメータの数が増えデータが複雑になると、開発と本番でのパラメータがずれてしまって想定していた性能が出ないという事態が発生します。これを防ぐためにも、前処理から予測モデルの構築までを含めたデータ処理のパイプライン全体を保存することが重要になります。

データ処理をパイプラインとして扱いやすくするという点においては、scikit-learnの抽象化はよくできています。それに影響を受けて、Sparkなどの機械学習ライブラリも、パイプラインで処理の再利用ができるようになっています。

パイプラインごと保存することで対応するコードがまとまるため、3の処理のパイプラインが複雑になったときにも見通しやすくなります。また、5の実験コードとアルゴリズムのパラメータが散らかる問題も、パイプラインでまとめることで一箇所で管理しやすくなります。

1.3.5 開発/本番環境の言語/フレームワークは揃える

最後に、開発と本番環境の言語/フレームワークはできる限り揃えましょう。例えば、予測モデルの開発はRで行い、アプリケーション側の予測処理はJavaで再実装をすると考えます。この場合、両言語での実装が必要になるため、アルゴリズムの変更コストが跳ね上がってしまいます。せっかくRで高

速なプロトタイピングをしたのに、本番に反映するのに時間がかかってしまったり、場合によっては断念したりしてしまうことがあります。

予測モデル開発は実験を何度も繰り返すことが多く、整理されたコードではない場合も多いです。実験用のコードから本番環境のアプリケーションコードへの移行も、共通のフレームワークを利用していれば移植のコストが低くなり意思疎通もしやすくなるでしょう。

6の課題についても、言語・フレームワークを揃えることがシステムの複雑性を抑える取り組みになります。

ただし、昨今では microservices^{†8} と呼ばれる機能やサービスごとに API やメッセージキューを通じてやりとりをするアーキテクチャも増えてきています。このアーキテクチャは、大きな1つのアプリケーションを作る monolithic なアーキテクチャと比較すると、システムを細かく分割し API などでやりとりをするため、機械学習の処理部分を切り出しやすくなります。この考え方のつどり、機械学習用の REST や gRPC などの API サーバを立てたりすることも増えてきています。Docker などのコンテナ技術とそれをデプロイしやすいクラウドサービスの登場により、機械学習の学習や予測の機能を切り出し API サーバを構築にも取り組みやすくなっています。特に、言語の選択は開発チームのスキルセットにも影響を与えてきますので、システム全体の設計やメンバーのスキルセットと照らし合わせて判断してください。

このように、実システムの機械学習プロダクトでは変化に耐えるための工夫をすることが重要です。より詳細なベストプラクティスを知りたい場合は、「Rules of Machine Learning: Best Practices for ML Engineering」[mlbestpractice] を参考にしてください。

scikit-learn はなぜ古典的機械学習の デファクトスタンダードになったのか？

scikit-learn は、深層学習以前の古典的機械学習のデファクトスタンダードとなったと言つても過言ではないでしょう。その一番の理由は、学習には `fit()` 関数、予測には `predict()` 関数といったように統一した API が提供されていることだと思います。統一した API のおかげで、scikit-learn では交差検証用の関数 `cross_val_score()` などの補助的な関数やクラスを様々なアルゴリズムに対して実施できます。

また、複数のアルゴリズムの切り替えも容易です。昔のライブラリは、单一か限られた数のアルゴリズムしか実装されておらず、アルゴリズムごとの性能比較をするには多くのコストが必要でした。それが、きれいな抽象化された API のおかげで、前処理のしきい値のリストを作成する

^{†8} <https://martinfowler.com/articles/microservices.html>

ことで容易にパラメータ探索を実行できたり、学習用のアルゴリズムのリストを作成することで各アルゴリズムのモデルを一度に作成できたりと、パラメータやアルゴリズムの探索を自動化するのも簡単です。

更に前処理やアルゴリズムのパイプラインを作成し、性能が最も良い組み合わせを保存することができます。これにより、ベストなパイプラインを予測時に再利用するといったことも簡単に行えます。

これらを統一的に操作できるメリットのため、Sparkなど他のライブラリでも同様のデザインを採用することが増えています。

1.4 機械学習を含めたシステムを成功させるには

機械学習を用いたシステムづくりには、ある種ギャンブルの要素があります。通常のコンピュータシステムでは、適切な設計がされていれば、何かしら動くものはできます（もちろん、ただ動くだけでは意味がなく、きちんとユーザーに対して価値のあるものを届けることは難しいです）。一方で、機械学習を含めたシステムの場合は、数週間から数ヶ月かけても、意味のあるアウトプットが何もできないという最悪のケースもあります。

分類のモデルを作ったけれどランダム出力のほうが良い性能となることも頻繁に起こります。機械学習を含めたシステムの開発は、通常のWebシステムの開発で期待されるような1,2週間という短いサイクルでイテレーションを回すのは難しく、数ヶ月かかることも珍しくありません。

では、機械学習を含めたプロダクトをビジネスとして成功させる上で重要なのは、どのようなチームでしょうか？私は以下の4者が重要だと考えます。^{†9}

1. プロダクトに関するドメイン知識を持った人
2. 統計や機械学習に明るい人
3. データ分析基盤を作れるエンジニアリング能力のある人
4. 失敗しても構わないとリスクを取ってくれる責任者

ドメイン知識を持った人はとても重要です。解くべき課題は何か？プロダクトのどこに機械学習の手法を使えばいいのか？を考える際に、ドメイン知識がないと全く見当違いの手法を選んでしまいかねません。また特徴量を決めたりデータを収集したりするときにも、ドメイン特有の知識が重要なカギとなります。

^{†9} 4人は別々の人格の場合もありますし、一人で複数役を行う場合もあります。ですが、一人でなんでもできる人を集めると属性が高くなり、プロジェクトの持続可能性が下がってしまうことに気をつけてください。

機械学習の知識がある人は、おそらく読者の皆さんが目指しているポジションだと思います。実装をすることに加えて、問題設定をするために他人達とのコミュニケーションを深めていくことも求められるでしょう。

昨今ではデータエンジニアと呼ばれる、データを活用できるようにする分析基盤を作るエンジニアが重要視されてきています。こうした人達と協力しながら、機械学習の基盤はどうあるべきかを考えていきましょう。

最後に、リスクを取ってくれる責任者が重要です。これは、機械学習がリスクの大きい投資だということを認識した上で、それでも機械学習を使わないとできない価値を生み出すことに背中を押してくれる存在です。できれば、機械学習やデータ分析の経験がある人の方が、詳細なリスク説明を省くことができ、意思決定までの時間が短くなって、現場は楽になります。そうでない場合は、プロダクトに関わる人がリスクを取ってでもやる必要性を説明するしかありません。時にはこの責任者に、不確実性に対する強権を発動してでも機械学習に投資をしてもらう必要があります。プロダクトを進める上で、強力な味方になってくれそうな人を探しましょう。



機械学習のプロダクトを作っていくうえでの進め方は、以下に示す2つの資料が役に立つと思います。通常のプロジェクトマネジメントと似ている所、違う所を含めて参考にして下さい。

<http://www.slideshare.net/shakezo/mlct4>

<http://www.slideshare.net/TokorotenNakayama/2016-devsumi>

1.5 この章のまとめ

本章では、機械学習プロジェクトの進め方の流れとそのポイントについて説明しました。

- 解くべき問題の仮説を立て、MVPを作りコンセプトの検証を最優先する
- 機械学習をしないことを恐れない
- 機械学習に適している問題設定かを見極める
- 予測性能とKPIの両方のモニタリングし、継続して改善を続ける

機械学習のプロジェクトは、どういう予測結果になるか特性が分からぬ未知のデータに対して探索的に試行錯誤をするため、通常のプロジェクトと比較して手戻りが発生しやすいです。ビジネスの目的明確にし、仮説をきちんと立てて、価値を出すためにはどうしたらよいかを考えながらプロジェクトを進めていきましょう。

2章

機械学習で何ができる？

機械学習を使うと何ができるのでしょうか。本章では機械学習でできることを、分類、回帰、クラスタリング、次元削減、その他に分けて解説します。まずは、その前にいろいろある機械学習に関するアルゴリズムをどう選ぶべきかについて、見てみましょう。

2.1 どのアルゴリズムを選ぶべきか？

どのアルゴリズムを使えばいいのかを考えるには、それぞれのアルゴリズムの特徴を知っている必要があります。まずは、機械学習にどんな種類のものがあるか大まかに押さえておきましょう。

分類

正解となる離散的なカテゴリ（クラス）と入力データの組み合わせで学習し、未知のデータからクラスを予測する

回帰

正解となる数値と入力データの組み合わせで学習し、未知のデータから連続値を予測する

クラスタリング

データを何かしらの基準でグルーピングする

次元削減

高次元のデータを可視化や計算量削減などのために低次元にマッピングする

その他

推薦：ユーザーが好みそうなアイテムや、閲覧しているアイテムに類似しているアイテムを提示する

異常検知：不審なアクセスなど、普段とは違う挙動を検知する

頻出パターンマイニング：データ中に高頻度に出現するパターンを抽出する

強化学習：囲碁や将棋のような局所的には正解が不明確な環境で、とるべき行動の方針を学習する

これだけいろいろあると、どれを選べばいいか悩むかもしれません。scikit-learnのチュートリアルに、便利なフローチャート^{†1}があるので、それをベースにしながらどのアルゴリズムを選べばいいか整理していきましょう（ただし、推薦、異常検知、頻出パターンマイニング、強化学習の場合は除きます）。図2-1に選択基準をまとめました。

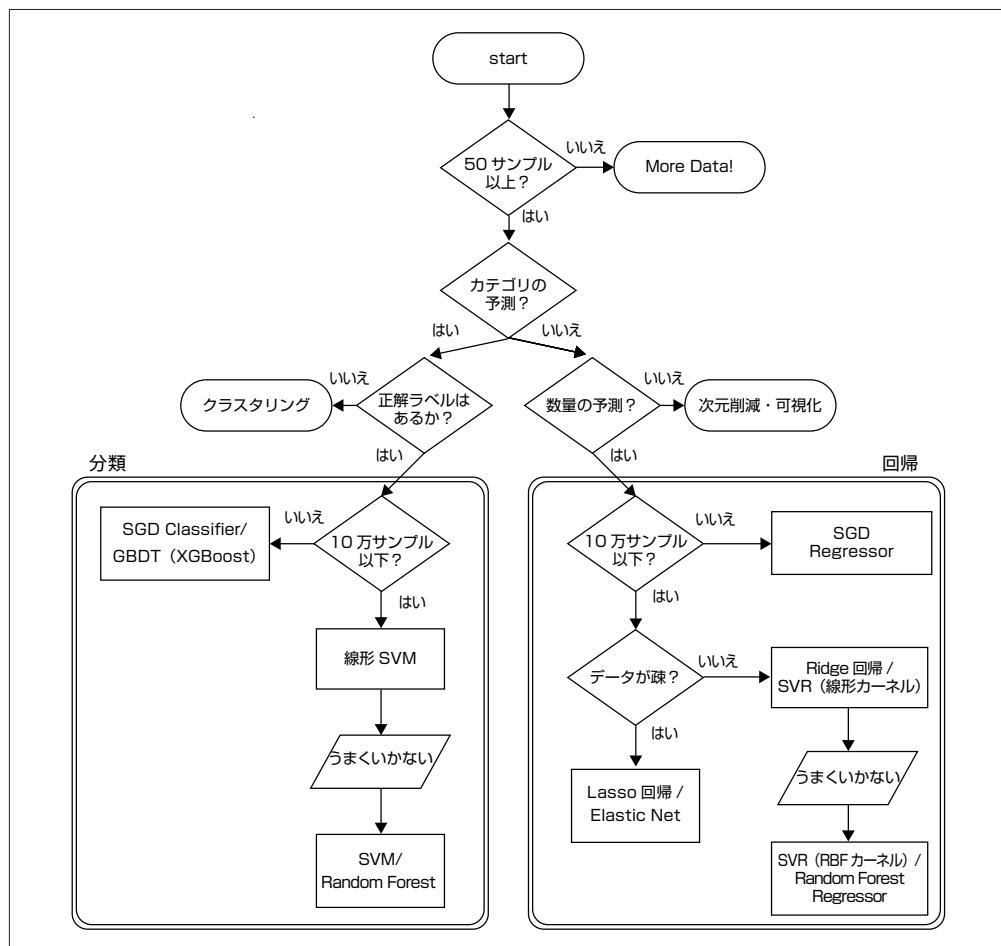


図2-1 どのアルゴリズムを選ぶべきか？

^{†1} http://scikit-learn.org/stable/tutorial/machine_learning_map/

学習に用いることができるデータ数、予測したい対象が離散的なカテゴリ（クラス）かどうか、正解ラベルが存在するかどうか、がポイントになります。特にデータ数が多すぎる場合は、後ほど紹介するオンライン学習のアルゴリズムを使う必要があります。

scikit-learn にあるオンライン学習アルゴリズム

scikit-learnには分類用のクラスとしてSGDClassifierが、回帰用としてSGDRegressorがあります。SGDClassifier、SGDRegressorは、損失関数と正則化項を何にするかでSVMやロジスティック回帰、SVRに近い分類や回帰ができます。なお、線形分類器としてのscikit-learnのオンライン学習アルゴリズムでは、Passive Aggressiveは実装されていますが、SCW (exact Soft Confidence Weighted) や AROW (Aaptive Regularization of Weight Vectors)などの新しめのアルゴリズムは実装されていません。^{†2}他のアルゴリズムが使いたい場合は、自分で実装をするか別のフレームワークを使うのが良いでしょう。

2.2 分類

分類 (Classification) は、教師あり学習の1つで、予測対象はカテゴリなどの離散的な値（クラス）を予測します。例えば、メールがスパムかどうかや、画像に映っているのがどういった物体か、といった離散値で表現できるものを予測する場合に分類のモデルを作ります。クラスの数が2の場合を二値分類または二クラス分類、3以上の場合は多値分類または多クラス分類 (Multiclass Classification) といいます。この章では、便宜上二クラス分類で説明をしていますが、多クラス分類の場合も基本的には同じような考え方できます。scikit-learnの多クラス分類の説明が分かりやすいので、詳細はそちらを参照ください。^{†3}

本節では、分類について以下のアルゴリズムを紹介します。

- パーセプトロン (Perceptron)
- ロジスティック回帰 (Logistic Regression)
- SVM (Support Vector Machine, サポートベクターマシン)
- ニューラルネットワーク (Neural Network)

^{†2} バージョン0.18.0からはニューラルネットワークのためにADAMの実装が入り、MLPClassifierクラスの最適化手法の1つとして使えます

^{†3} <http://scikit-learn.org/stable/modules/multiclass.html>

- k-NN (k近傍法, k-Nearest Neighbor Method)
- 決定木 (Decision Tree)
- ランダムフォレスト (Random Forest)
- GBDT (Gradient Boosted Decision Tree)

パーセプトロン、ロジスティック回帰、SVM、ニューラルネットワークの4つは、2つのクラスを分類する面を表現する関数を学習します。ここで言う関数とは面を表現している式と想っていただいて構いません。もちろん、これらのアルゴリズムを使い二クラス分類だけでなく多クラス分類の問題も解くことができます。



この2つのクラスを分類する面を決定境界 (Decision Boundary) と言います

k-NNは最近傍法とも呼ばれ、学習済みのデータから距離が近いデータを元に判断をします。決定木、ランダムフォレスト、GBDTは、木構造で表現されたルールの集合を学習します。

また、本章では詳しく扱いませんが、この他にもテキスト分類などでよく使われるナイーブベイズ (Naive Bayes) や、音声認識で伝統的に使われてきたHMM (Hidden Markov Model) などがあります。これらのアルゴリズムは、データの背景に隠れた確率分布を推測することで、データをモデル化する手法です。

分類問題の多くは、後ほど詳しく説明する目的関数と決定境界を知ることで、その違いが分かりやすくなると思います。式はできるだけ使わないように書きましたが、グラフを中心に読んでいただければと思います。

それでは、個々のアルゴリズムについて説明しましょう。

2.2.1 パーセプトロン

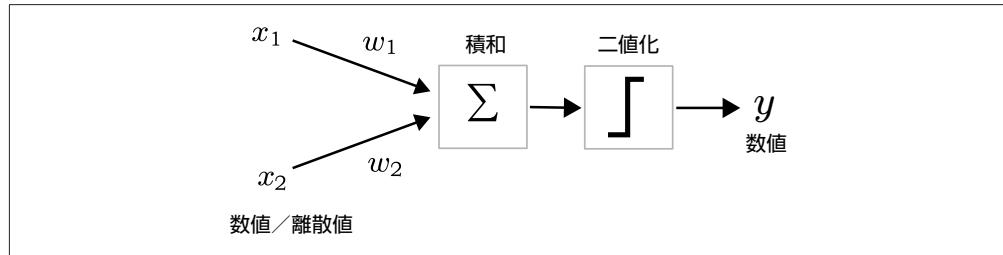


図2-2 パーセプトロン

パーセプトロン (perceptron、単純パーセプトロンと呼ぶこともあります) は、入力ベクトルと学習した重みベクトルを掛けあわせた値を足して、その値が0以上の時はクラス1、0未満の時はクラス2と分類するというシンプルなアルゴリズムです。^{†4} パーセプトロンを多層に重ねたものが、後述するニューラルネットワークになります。このシンプルなパーセプトロンを使って、どのように分類をしているのか学びましょう。

パーセプトロンの特徴

パーセプトロンには、以下のような特徴があります。

- オンライン学習で学習する
- 予測性能はそこそこで、学習は速い
- 過学習しやすい
- 線形分離可能な問題のみ解ける

見慣れない用語がいくつか出てきましたね。これらは他のアルゴリズムの説明にも出てくるので、順番に説明したいと思います。

オンライン学習 (Online Learning) とその対義語のバッチ学習 (Batch Learning) は、ひとまずここではデータを1つずつ入力して最適化をするか (オンライン学習)、全部を入れて最適化をするか (バッチ学習) だと思ってください。詳しくは、「4.2.1 混乱しやすい「バッチ処理」と「バッチ学習」」で紹介します。

過学習 (Overfitting) した予測モデルについては前の章で説明しました。「学習に使ったデータに対してはきちんと正解できるが、知らないデータに対してはまったく正解できない」という状態になっているモデルでした。過学習は、機械学習をしている上でよく発生する現象の1つで、特微量を減らす、後述する正則化項を導入する、今過学習しているものよりシンプルなアルゴリズムを使ったりすることで避けられます。

伝統的なパーセプトロンに、過学習を抑える仕組みは組み込まれていません。

過学習と反対の現象を未学習 (Underfitting) と言い、モデルに入力と出力の関係が反映されていない状況です。未学習は、ドメイン固有の特微量が含まれていない、モデルの表現力が足りない、正則化項の影響が強すぎるなどの理由で起こります。

^{†4} ここでは、パーセプトロンの活性化関数（後述）はステップ関数を設定していますが、他の関数を利用することもできます。

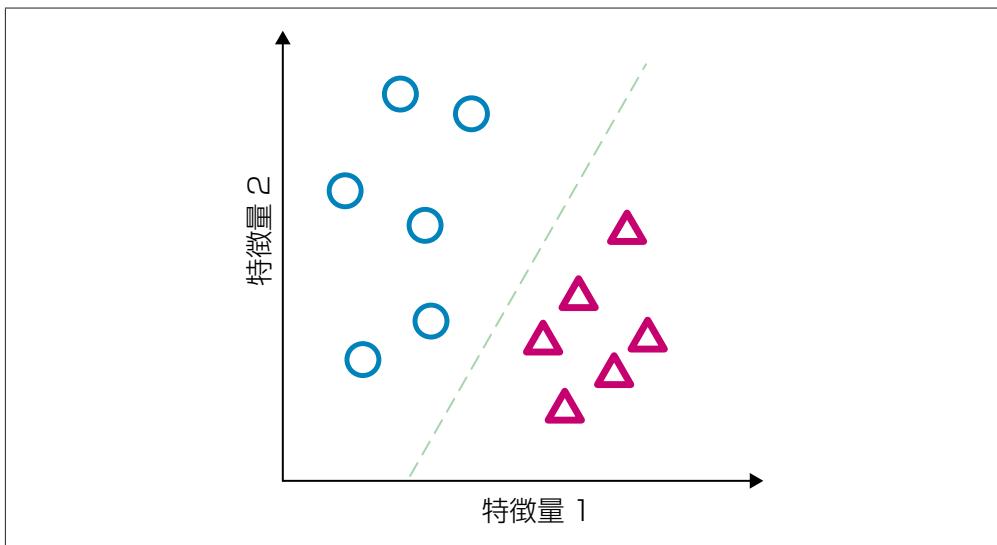


図2-3 線形分離可能なデータ

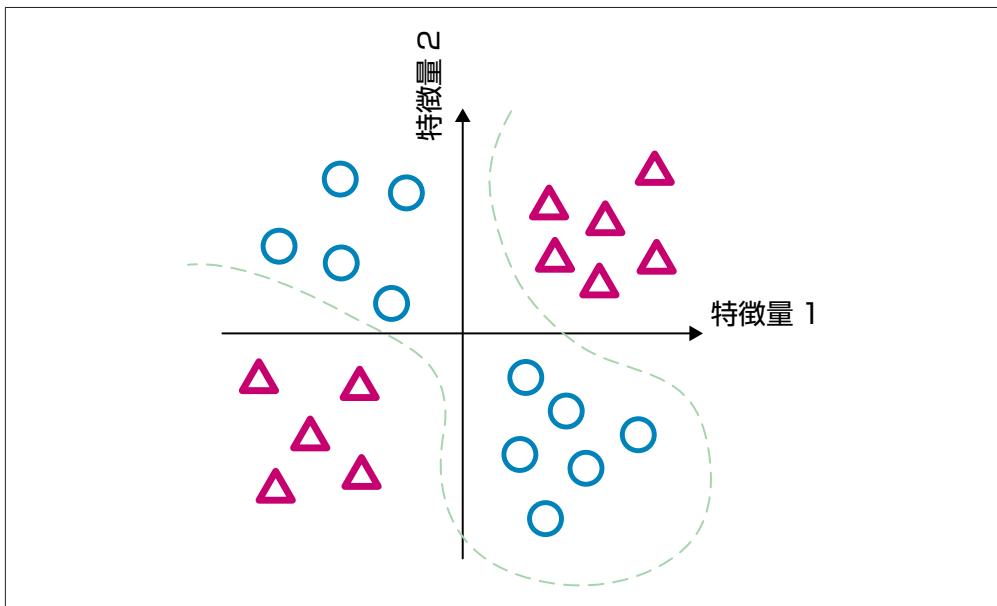


図2-4 線形分離不可能なデータ

パーセプトロンは線形分離可能 (Linearly Separable) な問題のみを解くことができます。線形分離可能とは、図2-3のように、データがある直線で切りよく2つに分けられるデータのことを言います。この分類する直線を少し専門的な言い方で超平面 (hyperplane) と言います。二次元の時は直線ですが、

三次元だと平面になります。高次元空間での平面ということで超平面と言います。

逆に図2-4のように、直線で分けられないデータのことを線形分離できないので線形分離不可能（あるいは非線形分離可能）なデータといいます。例としてよくあげられるのは排他的論理和（XOR、Exclusive or）のデータです。図2-4のようにXORは、原点を中心として右上（横軸と縦軸が共に正の領域）と左下（横軸と縦軸が共に負の領域）が1つのクラス、右下（横軸が正で縦軸が負の領域）と左上（横軸が負で縦軸が正の領域）が1つのクラスになるようにデータが存在します。そのため、1本の直線を引くだけでは2つのクラスを適切に分離することはできません。この、「1本の直線を引くだけで2つのクラスを分離できない」のが線形分離不可能ということです。

パーセプトロンの決定境界

パーセプトロンを実際に学習したモデルの決定境界が、図2-5と図2-6です。図2-5は線形分離可能なデータを生成しています（多少入り組んでいるのはノイズが乗っているからです）。図2-5は図2-4のように、原点を中心として右上と左下に●のデータが、右下と左上に▲のデータが生成されています。パーセプトロンは非線形的な分離はできないので、決定境界は直線になっています。なので、XORは分離することができていません。なお、この決定境界を描画するノートブックは、レポジトリの chap02/Decision_boundary.ipynb にあります

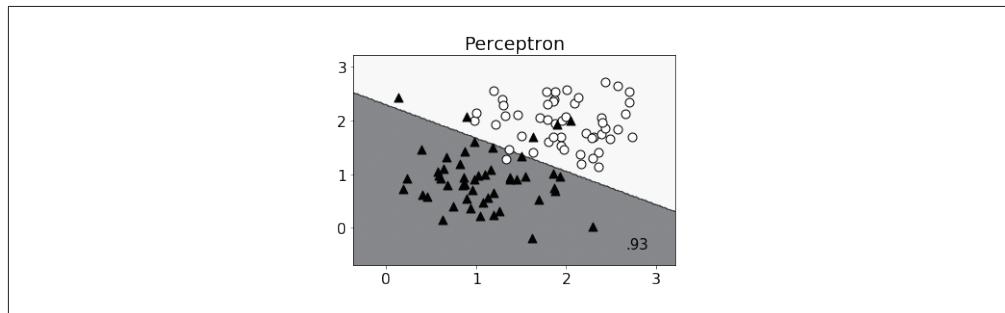


図2-5 パーセプトロンの決定境界（線形分離可能）

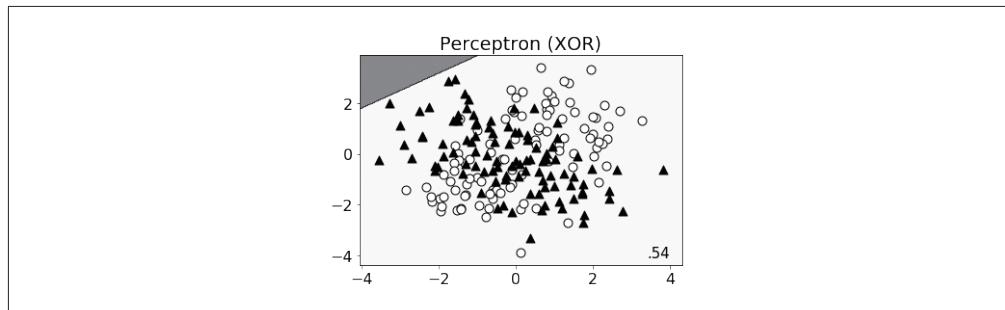


図2-6 パーセプトロンの決定境界（線形分離不可能）

パーセプトロンの仕組み

パーセプトロンの入力データが2種類の情報からなるリストだとします（これを2次元の特徴量と言います）。入力をリストx、特徴量の重要度を表す重みをリストwとしてコードで書くと、パーセプトロンはまず入力と重みをかけ合わせ合計を出します。これをコードで表現すると以下のようにになります。

```
sum = b + w[0] * x[0] + w[1] * x[1]
```

この処理を数学記号で \sum と表現します（図2-2で「積和」と表現している部分です）。なお、bはバイアス項と呼ばれ、入力と掛け合わせない特殊な重みになります。計算の簡略化のためにバイアスを無視すると、数値計算ライブラリNumPyのnumpy.dotという関数を使って掛け合わせることができます。以降はバイアスを無視して例示します。例えば、重みベクトルwが[2, 3]、入力ベクトルxが[4, 2]だったとすると、以下のように書くことが出来ます。

```
import numpy as np
w = np.array([2, 3])
x = np.array([4, 2])
sum = np.dot(w, x)
```

次にこの合計値sumが、正なのか負なのかでクラスを判断します（図2-2で「二値化」と表現している部分）。これをコードで表現すると以下のようにになります。

```
if sum >= 0:
    return 1
else:
    return -1
```

の前段は特徴量と重みを掛けさせて足すところを、後段は正負を判断するところを表しています。これらをまとめると、パーセプトロンの予測コードは以下のようになります。

```
import numpy as np
# パーセプトロンの予測
def predict(w, x):
    sum = np.dot(w, x)

    if sum >= 0:
        return 1
    else:
        return -1
```

では、どうやって適切なパラメータ（この場合w）を推定すればいいのでしょうか？それには真の値と予測値とのズレを表す関数を用います。この関数を損失関数（Loss Function）または誤差関数（Error Function）といいますが、本書では以降、損失関数と言うことにします。この損失関数を使うことで、今学習している予測モデルがどれくらい良いものなのかを測ります。

例えば、誤差の二乗を損失関数とすると、

$$\text{損失関数} = (\text{真の値} - \text{予測値})^2$$

となります。

パーセプトロンの損失関数は、 w が重みベクトル、 x が入力ベクトル、 t の正解ラベル（1か-1）としたとき、 $\max(0, -twx)$ というヒンジ損失 (Hinge Loss) を使います^{†5}。図2-7を見ると分かりますが、蝶つがい（ヒンジ）のように見えることからこの名前が付いています。ヒンジ損失を使うと、0以下の値を取る時、つまり誤分類されたときに損失が大きくなり、正解した時の損失は0となります。予測値が大きく間違えば間違うほど、損失も線形に増えていくのが特徴です。

パーセプトロンのヒンジ損失関数を使い、全データに対して和を取るコードは以下のようになります。

```
import numpy as np
def perceptron_hinge_loss(w, x, t):
    loss = 0
    for (input, label) in zip(x, t):
        v = label * np.dot(w, input)
        loss += max(0, -v)
    return loss
```

誤分類されるデータができるだけ少なくなるような重みを見つけることで、この全データに対する損失の和を最小化します。

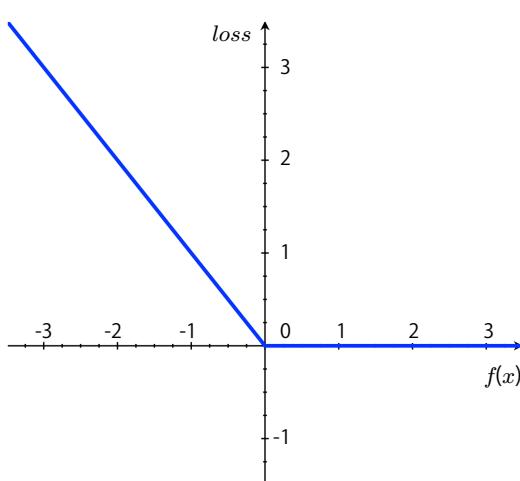


図2-7 パーセプトロンのヒンジ損失（パーセプトロン規準）

^{†5} パーセプトロン規準とも呼ばれることもあります。一般的にヒンジ損失というとSVMでよく用いられるヒンジ損失の $\max(0, 1-twx)$ を指すことが多いです。違いは横軸との交点になります。

ここで、損失関数をもう少し一般化して、どれくらいモデルがデータに合っているのかというのを表す関数を目的関数 (Objective Function) と呼ぶことにします（目的関数は評価関数と呼ばれることもあります）。ここで、パーセプトロンの目的関数は

$$\text{目的関数} = \text{損失関数の全データでの和}$$

となり、目的関数を最小化することが、誤りの少ない最適な分類ができる状態になると言えます。こうなる重みベクトル w を得ることが「モデルを学習する」ということになります。

では、どのようにパラメータである重みベクトル w を推定すればよいのでしょうか。パラメータの最適化には確率的勾配降下法 (Stochastic Gradient Descent, SGD) と呼ばれる方法がよく使われます。この方法では、目的関数の山の上から谷に向かって少しづつ降りることで、最適なパラメータを得られます。実際に谷の場所は直接見えない状態で、周囲の限られた範囲しか見えません。そのためのように、坂の傾きが大きい下り方向に向かって一歩一歩進んでいき、パラメータを修正します。そして、目的関数が最も小さいところにたどり着けば、そのパラメータが最適な値となります（これを解が収束すると言います）。

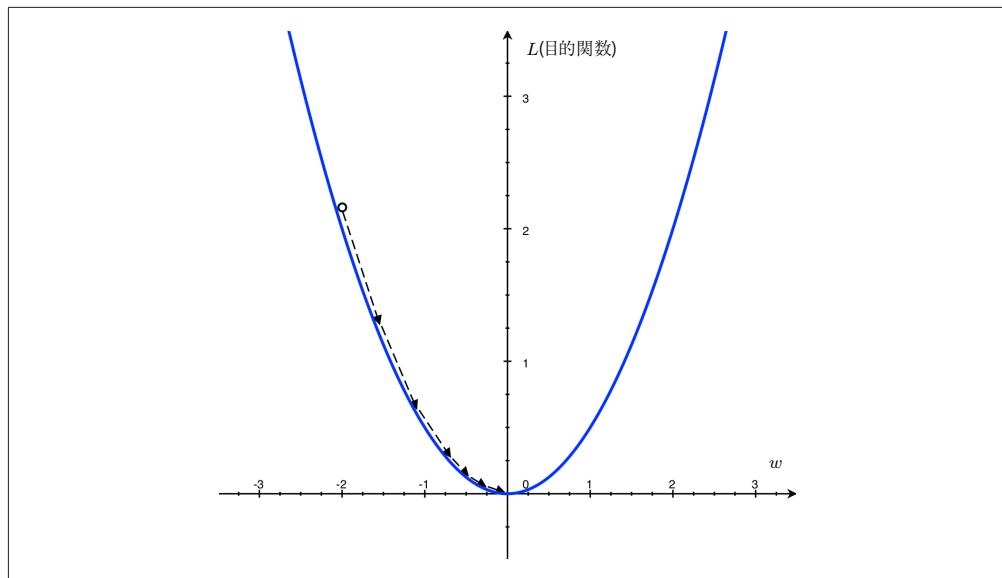


図2-8 確率的勾配法のイメージ

ちなみに、どれくらいの幅でパラメータを修正するかを決めるハイパーパラメータを学習率 (Learning Rate) と言います。修正する幅は学習率 × 山の傾きで決まります。学習率が大きい値だと速く収束するかもしれません、谷を行き過ぎて最適な解に収束しない場合もあります。学習率が

小さい場合は、収束するまでに必要な繰り返し回数が増えるため、学習時間が長くなります。シンプルな方法では学習率が固定のまま学習しますが、後述するニューラルネットワークではこの設計が肝になってくることもあり、動的に変化させる様々な工夫が提案されています。

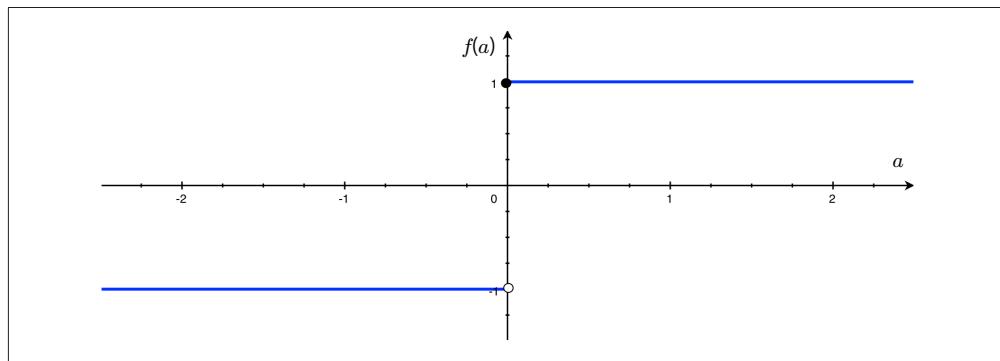


図2-9 ステップ関数

さて、パーセプトロンの予測値は重みベクトルと入力ベクトルを掛けた結果の正負で判定しましたね。これは掛けあわせた結果を、ステップ関数 (Step Function) という関数に通しているとも言えます。ステップ関数とは図2-9のような関数で、入力の値を+1または-1してくれます。^{†6}特に、パーセプトロンにおけるステップ関数のような、出力値を非線形変換する関数を活性化関数 (Activation Function) と呼びます。

パーセプトロンは、その後の様々なアルゴリズムに影響を与えた、歴史的に重要なアルゴリズムです。引き続き、パーセプトロンの仲間のアルゴリズムについて学んでいきましょう。

2.2.2 ロジスティック回帰

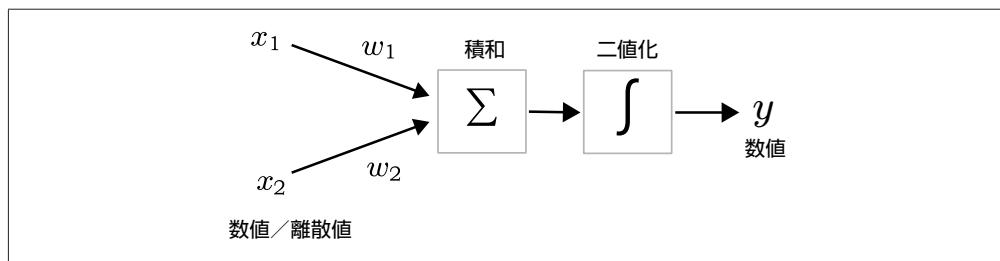


図2-10 ロジスティック回帰

^{†6} 通常のステップ関数は0か1を出力しますが、パーセプトロンによる2クラス分類は、計算のしやすさから2つのクラスを-1、+1とすることが多いです。

ロジスティック回帰(Logistic Regression)は、回帰という名前とは裏腹に分類のためのアルゴリズムです。ロジスティック回帰はシンプルな方法ながら、パラメータ数もそれほど多くなく予測に時間がかかるないこともあります。様々な機械学習アルゴリズムを比較する際のベースライン(基準となる比較対象)としてよく使われています。例えば、Google マップでは駐車場の空き具合を推定するのに、工夫した特徴量とロジスティック回帰を使っています。^{†7}

ロジスティック回帰の特徴

ロジスティック回帰はパーセプトロンと似ていますが、以下のような特徴があります。

- 出力とは別に、その出力のクラスに所属する確率値が出せる
- 学習はオンライン学習でもバッチ学習でも可能
- 予測性能はまずまず、学習速度は速い
- 過学習を防ぐための正則化項が加わっている

特に出力の確率値が出せるという特徴のため、広告のクリック予測にもよく使われています。

ロジスティック回帰の決定境界

ロジスティック回帰も線形分離可能な対象を分離するアルゴリズムなので、決定境界は直線になります。

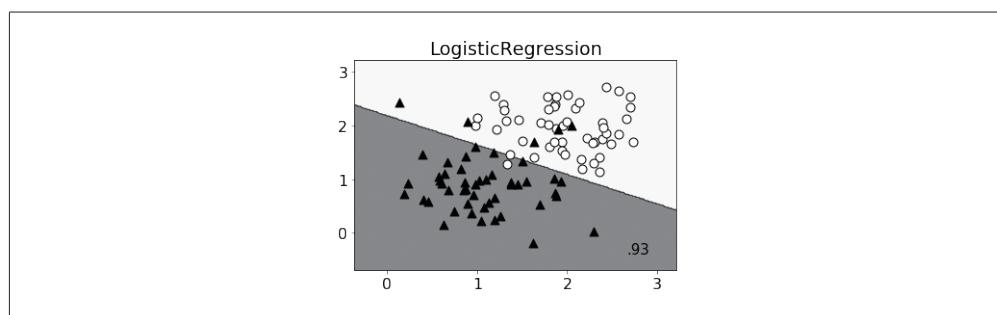


図2-11 ロジスティック回帰の決定境界（線形分離可能）

^{†7} <https://research.googleblog.com/2017/02/using-machine-learning-to-predict.html>

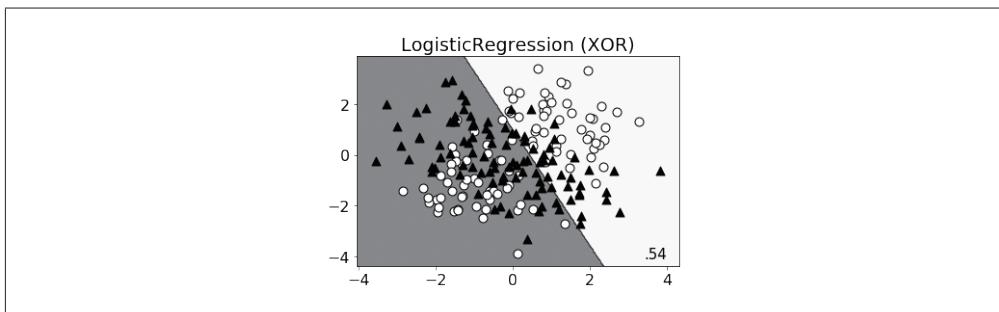


図2-12 ロジスティック回帰の決定境界（線形分離不可能）

ロジスティック回帰の仕組み

パーセプトロンとの違いは、活性化関数がシグモイド関数 (Sigmoid Function) (ロジスティック・シグモイド関数 (Logistic Sigmoid Function) とも言います) であること、損失関数が交差エントロピー誤差関数 (Cross-entropy Error Function) と呼ばれる損失関数であること、正則化項 (Regularization Term) (または罰則項 (Penalty Term)) が加わっているためパーセプトロンより過学習を防ぎやすいこと、オンラインでもバッチでも学習可能なことが挙げられます。

見知らぬキーワードがいくつかでてきたので、1つずつ説明していきます。

シグモイド関数は図2-13に示すような形状をしています。入力が0の時は0.5をとり、値小さくなるほど0に、大きくなるほど1に近づく関数です。ちなみに、シグモイド関数を一般化したものはロジスティック関数と呼ばれており、ロジスティック回帰の由来となっています。このシグモイド関数を通した値があるクラスに属する確率になり、0.5以上かどうかでそのクラスに所属するかどうかを決めます。なお、この0.5というしきい値をハイパーパラメータと考え、求める性能によって調整することもあります。

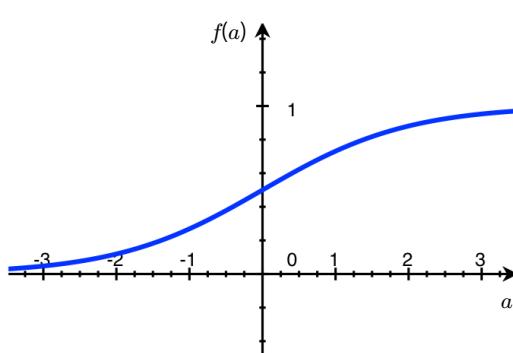


図2-13 シグモイド関数

シグモイド関数を記述するコードは、以下の通りです。

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

つまり、出力 y は $y = \text{sigmoid}(\text{np.dot}(w, x))$ と表すことができます。

2値分類の時の交差エントロピー誤差関数は、 N 個のデータに対して y を出力、 t を正解ラベル（正しい場合は1、間違っている場合は0とする）、 \log を底が e の自然対数とすると、次のような式で表すことができます。

$$E = - \sum_{n=1}^N t_n \log y_n + (1 - t_n) \log (1 - y_n)$$

この損失は、正解 ($t=1$) のときは $\log y_n$ を考えれば良く、重みと入力の積和（パーセプトロンでいうところの $\text{np.dot}(w, x)$ ）が0より小さいと損失が急激に大きくなり、0より大きいと段々と小さくなっていきます。こうすることで、重みと入力の積和を大きくしようとなります。不正解 ($t=0$) のときは、 $\log(1 - y_n)$ を考えればよく、正解のときと左右反転したグラフになります。重みと入力の積和を小さくしようとなります。詳しくは脚注のリンクを参考してください^{†8}。

2値分類の時の交差エントロピー誤差関数をコードで書くと以下のようにになります。

```
def cross_entropy_error(y, t, eps = 1e-15):
    y_clipped = np.clip(y, eps, 1 - eps)
    return -1 * (sum(t * np.log(y_clipped) + (1 - t) * np.log(1 - y_clipped)))
```

`np.clip(y, eps, 1 - eps)`は、 y の値が0または1にならないように、`eps`という微小な値を足しています。これは、対数に0を渡す(`np.log(0)`)と $-\infty$ の値を取るので、それを避けるための操作です。`numpy.array`を想定した処理として`np.clip()`関数を使っていますが、データが1個の場合、`max(min(y, 1- eps), eps)`と同じです。

^{†8} <http://gihyo.jp/dev/serial/01/machine-learning/0018>

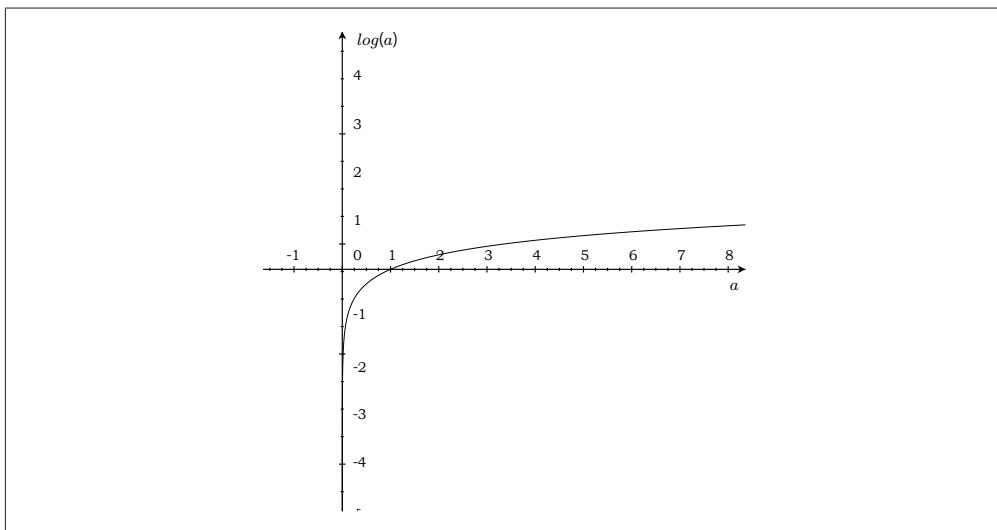


図2-14 対数のグラフ

正則化 (Regularization) は、学習時にペナルティを与えることで決定境界をなめらかにする働きを持ちます。入力データに対して最適化する際に正則化項を加えることで、既知の教師データの影響を受けすぎないようにします。言い換えると、モデルをシンプルに保つための補正項です。こうすることで過学習を抑え、汎化性能を手に入れることができます。回帰分析の例になりますが、図2-15に正則化のイメージを示します。駅からの距離と家賃の関係を表すモデルを得るときに、正則化が弱すぎると右の図のように学習データに対して必要以上にフィットした線になってしまいます。それに対して、正則化が強すぎると今度は学習データの特性を大雑把にしか獲得できていません。

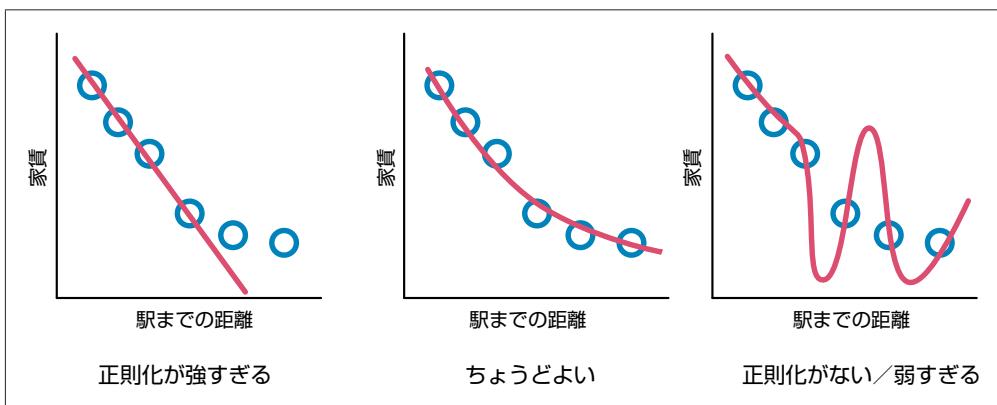


図2-15 正則化のイメージ

正則化項を加えると目的関数は以下のように表現できます。

$$\text{目的関数} = \text{損失関数の全データでの和} + \text{正則化項}$$

この目的関数を最小化するパラメータを推定することで、ロジスティック回帰を学習できます。ペセプトロンと同じく確率的勾配降下法を使うことで最適化できます。

数式で説明する正則化

ある学習手法による決定境界が $f(\mathbf{x}) = \mathbf{w}\mathbf{x} = \sum_{i=1}^m w_i x_i$ という式だとします。

ただし、 \mathbf{x} は m 次元の入力データの特徴ベクトル、 \mathbf{w} は学習した重みとします。この時、正則化項は例えば $\lambda \sum_{i=1}^m w_i^2$ と書くことが出来ます。これを L2 正則化と言います。重みパラメータの2乗をペナルティとして損失関数に加えます。 λ はペナルティの影響度をコントロールするための正則化パラメータです。

ここで、目的関数を改めて書くと $\text{目的関数} = \text{損失関数の全データでの和} + \lambda \sum_{i=1}^m w_i^2$ となり、損失関数を最小化する際に、大きすぎる重みパラメータに対してペナルティを加えることになります。 λ にどの値が良いかは、交差検証を用いて決定します。

また、正則化項には、 $\lambda \sum_{i=1}^m |w_i|$ という絶対値の形で表現する正則化もあります。これを L1 正則化と呼びます。L1 正則化を用いると、重み w_i が多くの i で 0 となるため、特徴選択の効果があることが知られています。

なお、Lasso 回帰は正則化項として L1 正則化を用いた線形回帰、Ridge 回帰は L2 正則化を用いた線形回帰、Elastic Net はその両方を混ぜ合わせたものです。

2.2.3 SVM

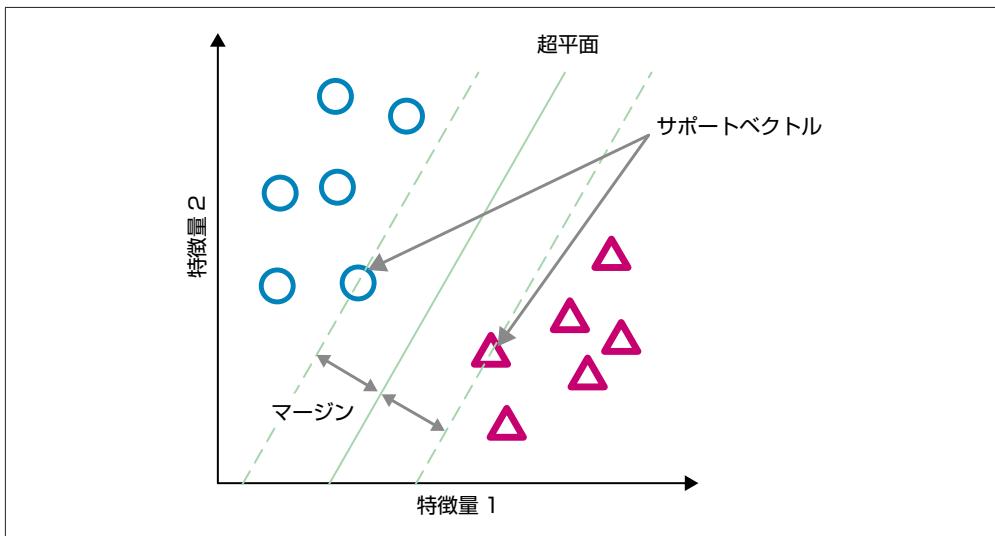


図2-16 SVM

SVM (Support Vector Machine) は分類問題を解くときに非常に良く利用されるアルゴリズムです。パーセプトロンを拡張したアルゴリズムと考えることができます。線形分離可能な問題だけでなく非線形な分離が必要な問題にも適用できます。様々なアルゴリズムやライブラリが開発されており、高速な学習ができます。

SVMの特徴

SVMには、以下のような特徴があります。

- マージン最大化することで、なめらかな超平面を学習できる
- カーネル (Kernel) と呼ばれる方法を使い、非線形なデータを分離できる
- 線形カーネルなら次元数の多い疎なデータも学習可能
- バッチ学習でもオンライン学習でも可能

見慣れない言葉が多くありますが、後ほど詳しく説明します。まずは、どういった決定境界をとるか見てみましょう。

SVMの決定境界

SVMでは、後述するカーネルと呼ばれる方法を使うことで、線形分離可能な問題にも非線形分離可能な問題にも適用が可能です。SVMでよく使われる線形カーネルとRBFカーネルの決定境界を見て

みましょう（それぞれのカーネルについては後ほど説明します）。

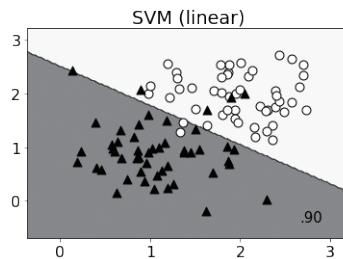


図2-17 SVM（線形カーネル）の決定境界（線形分離可能）

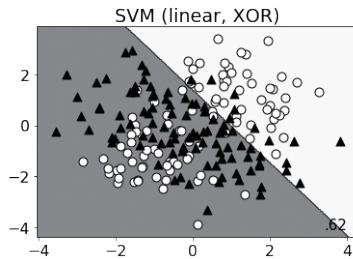


図2-18 SVM（線形カーネル）の決定境界（線形分離不可能）

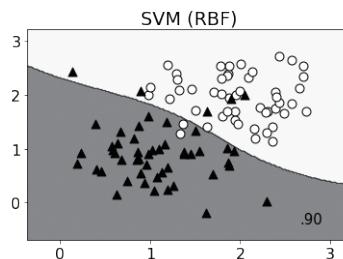


図2-19 SVM（RBF カーネル）の決定境界（線形分離可能）

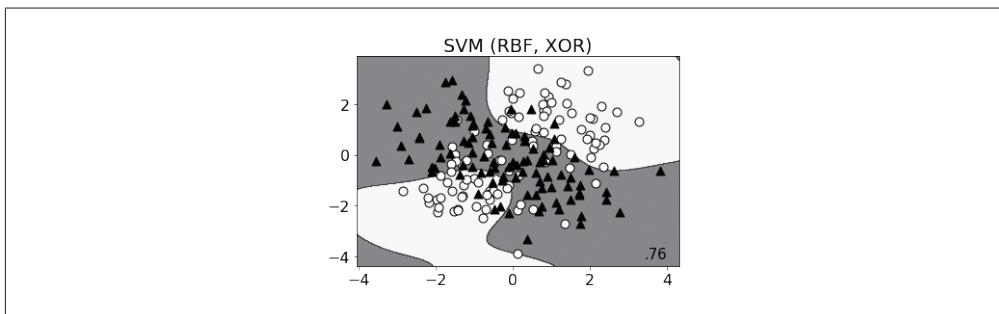


図2-20 SVM (RBF カーネル) の決定境界 (線形分離不可能)

線形カーネルは直線で分離をし、RBFカーネルは非線形に分離していることが分かります。

SVMの仕組み

損失関数はパーセプトロンと同じく、ヒンジ損失を使います(図2-21)。厳密にはパーセプトロンとは、横軸との交点の場所が違います。この交点の違いにより、決定境界ギリギリで正解をしているデータにも弱いペナルティを与えて、決定境界に対してマージンを持たせることができます。

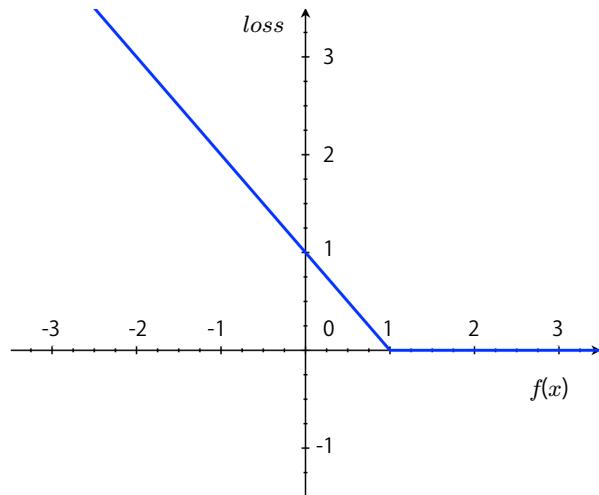


図2-21 SVMのヒンジ損失

SVMの特徴は大きく2つあります。

1つ目の特徴は、マージン最大化をすることで、正則化項と同じように過学習を抑えることができます。マージン最大化は、図2-16のように超平面をどのように引けば、2クラスそれぞれ最も近いデータ

(これをサポートベクターと言います)までの距離が最大化できるかを考えます。超平面は2クラスを分離する平面のことでしたね。マージン最大化とはつまり図中のマージンが最大になるような超平面の引き方を決めてることで、既知のデータに対してあそびが生まれます。このあそびのおかげで、既知のデータに特化し過ぎない余裕のある決定境界を得ることが出来ます。すなわち、過学習を抑えることができるのです。最適化手法には様々な種類があり詳細は割愛しますが、バッチ学習のためのアルゴリズムもオンライン学習のためのものも両方あります。

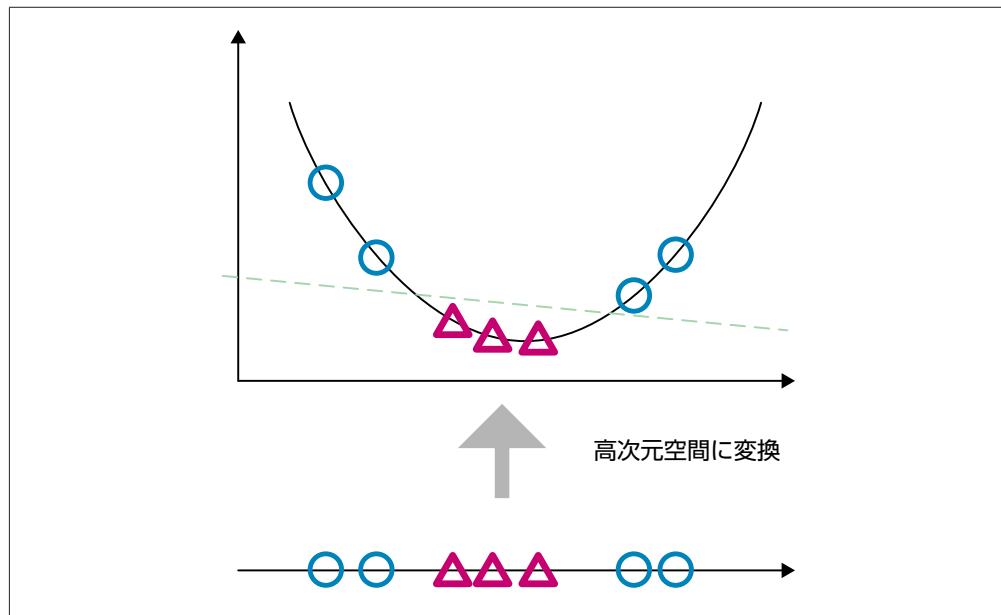


図2-22 カーネルを使った決定境界の例

2つ目の特徴は、カーネルと呼ばれるテクニックです。これは線形分離不可能なデータでも、カーネルと呼ばれる関数を使って特微量を擬似的に追加してデータをより高次元のベクトルにすることで、線形分離可能にするという方法です。図2-22のようなイメージで、1次元では線形分離できなかったのが、2次元に変換することで、線形分離可能になっています。

カーネルには、線形カーネル (Linear Kernel)、多項式カーネル (Polynomial Kernel)、RBF カーネル (Radial Basis Function Kernel、動的基底関数カーネル) などがあります。図2-17から図2-20に、線形カーネルとRBFカーネルの時の決定境界を示します。特に、図2-18と図2-20を見比べると分かるように、RBFカーネルの方がXORに対して適切に分離できているのが分かります。線形カーネルは処理の速さから主にテキストなどの高次元で疎なベクトルのデータに、RBFカーネルは画像や音声信号などの密なデータによく使われます。

これに対して、疎なベクトルとは入力ベクトルのほとんどが0でたまに値が入っているベクトルのことです。テキストデータは、例えば単語の頻度を入力ベクトルにすると単語の種類数だけ入力の次元が増え、10,000次元以上の入力ベクトルになることもあります。反対に、入力ベクトルのほとんどの要素が0以外の値が入っている場合、密なベクトルと言います。画像のベクトルは、例えば 16×16 のサイズにまで画像をリサイズし、256次元の0がほとんどないベクトルを使うなど、密なベクトルになることが多いです。

2.2.4 ニューラルネットワーク

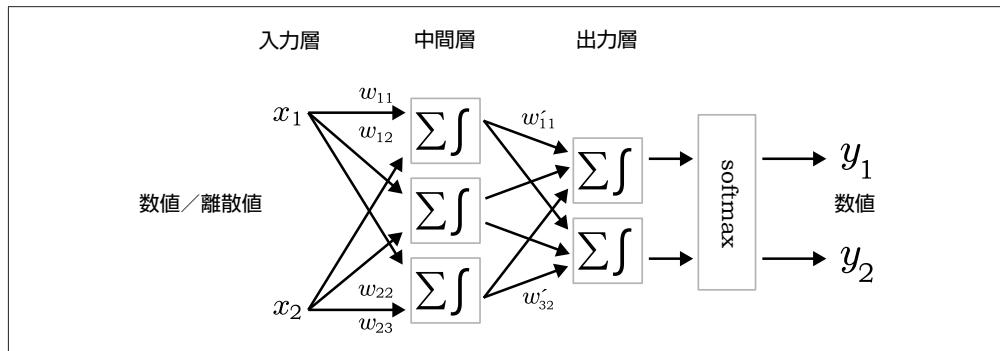


図2-23 ニューラルネットワーク

ニューラルネットワーク (Neural Network) は多層パーセプトロンとも呼ばれ、パーセプトロンを1つのノードとして階層上に重ねたもののことです。脳の神経細胞 (ニューロン) がシナプスで結合をし、電気信号で情報伝達をする神経回路から着想を得たため、この名前で呼ばれています。

ニューラルネットワークの特徴

ニューラルネットワークには以下のような特徴があります。

- 非線形なデータを分離できる
- 学習に時間がかかる
- パラメータの数が多いので、過学習しやすい
- 重みの初期値に依存して、局所最適解にはまりやすい

パーセプトロンと比較しての大きな特徴として、非線形なデータを分離できるという点が挙げられます。また、ニューラルネットワークの計算は層が深くなると計算に時間がかかるという問題がありました。しかし近年、GPUを活用することで学習に時間がかかる問題を高速化できることが分かったため、深い層のニューラルネットワークが実時間で計算可能となりました。また、多くのフレームワークがGPU対応したニューラルネットワークの計算を取り入れて、ニューラルネットワークが普及したの

です。ニューラルネットワークはパラメータ数も多くなりがちなので、データ量はほかの手法と比較しても十分に用意したほうが良いでしょう。

ニューラルネットワークの決定境界

ニューラルネットワークの場合層を多段に重ねるため、パーセプトロンとは異なり直線でない決定境界も取ることができます。

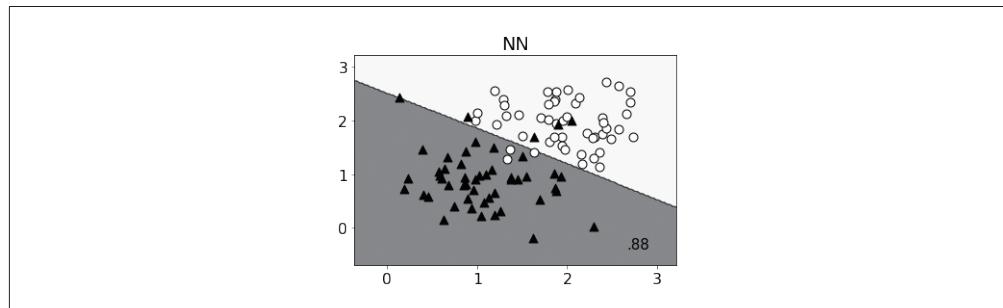


図2-24 ニューラルネットワークの決定境界(線形分離可能)

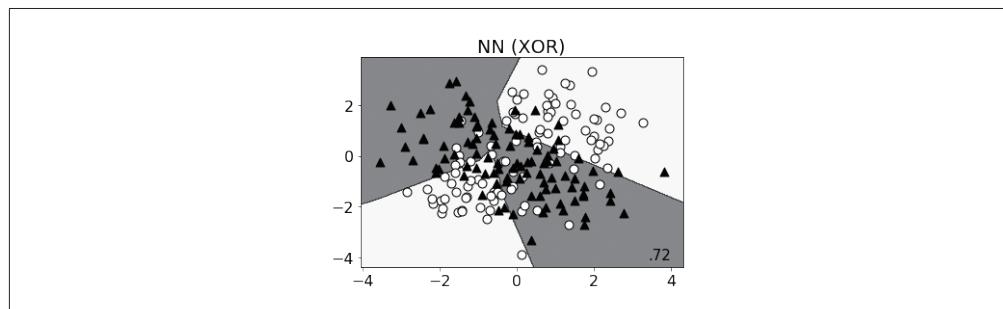


図2-25 ニューラルネットワークの決定境界(線形分離不可能)

ニューラルネットワークの仕組み

ニューラルネットワークはいろいろな形状がありますが、一番基本的な3階層のフィードフォワード型ニューラルネットワークを考えます。^{†9} 入力層、中間層、出力層という順番に入力と重みを掛けあわせて計算していく、出力層に分類したいクラスだけノードを用意します。最後に出力層の計算した値をsoftmax関数と呼ばれる方法で正規化を行い、確率とみなせるようにすることができます。softmax関数は、最も数値が大きかったクラスを答えるクラスと考えます。入力層、出力層の数に応じて中間層(隠れ層とも言います)の数を用意します。

^{†9} 流儀によってはこれを2層と呼ぶ場合もあります

活性化関数には、最初期はステップ関数が、その後シグモイド関数が良く利用されてきました。近年では深層学習において ReLU (Rectified Linear Unit) の性能が良いためよく使われています（図 2-26）。

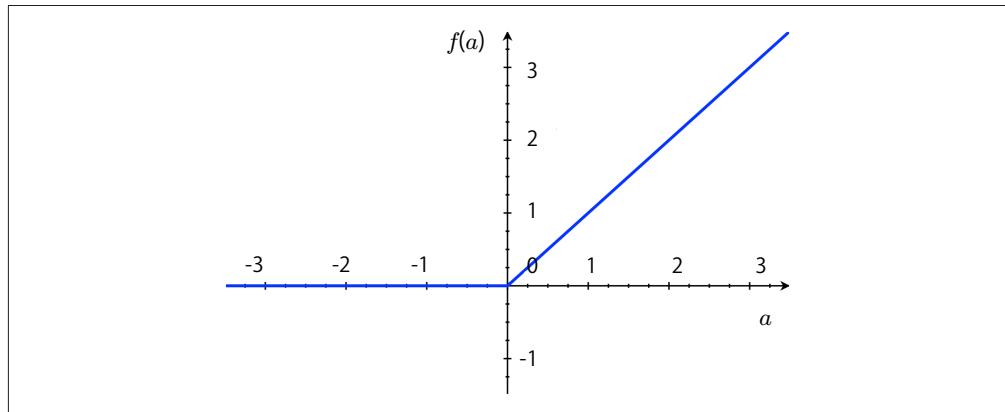


図2-26 ReLU

フィードフォワード型のニューラルネットワークは、誤差逆伝播法（バックプロパゲーション、Backpropagation）と呼ばれる方法で学習します。ランダムに初期化した重みの値を使って出力値をネットワークの順方向に計算し、計算した値と正解となる値との誤差をネットワークの逆方向に計算して重みを修正します。そして重みの修正量が規定値以下になるか、決められたループ回数を繰り返したところで学習を打ち切ります。

単純にニューラルネットワークの中間層の層数を増やすと、誤差逆伝播法では学習ができない問題があります。様々な工夫をすることで、それを解決して深いネットワークも学習できるようにしたのが深層学習です。

なお、scikit-learn には 0.18.0 から多層パーセプトロンの実装が入りました。他にも TensorFlow や Chainer、PyTorch、MXNet、CNTK など各種ディープラーニング向けのフレームワークも人気です。

2.2.5 k-NN

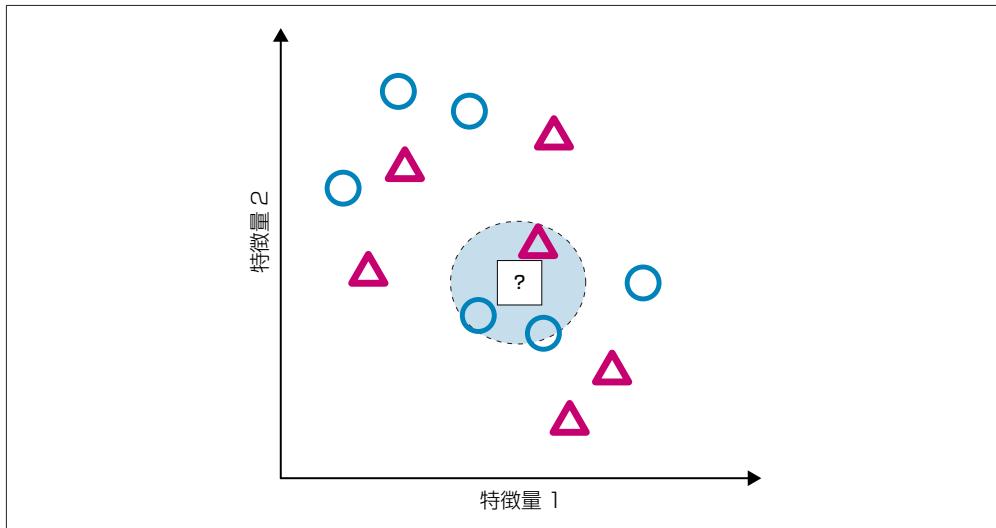


図2-27 k-NN

k-NN (k近傍法、k-Nearest Neighbor Method) は、未知の1個のデータが入力された時、そのデータのクラスを近くの k 個の既知データの所属するクラスの多数決で決めるという、最近傍探索のアルゴリズムの1つです。シンプルな発想のため、単純な分類器として直接使うだけでなく、k-NN的なアプローチで類似アイテムを探査するといった使い方も可能です。

k-NNの特徴

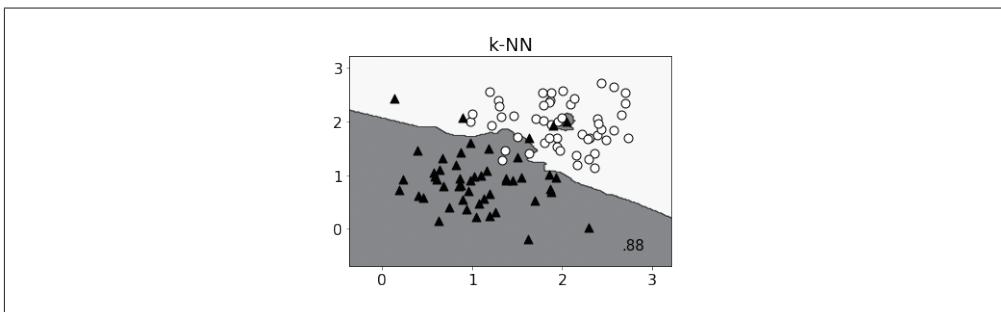
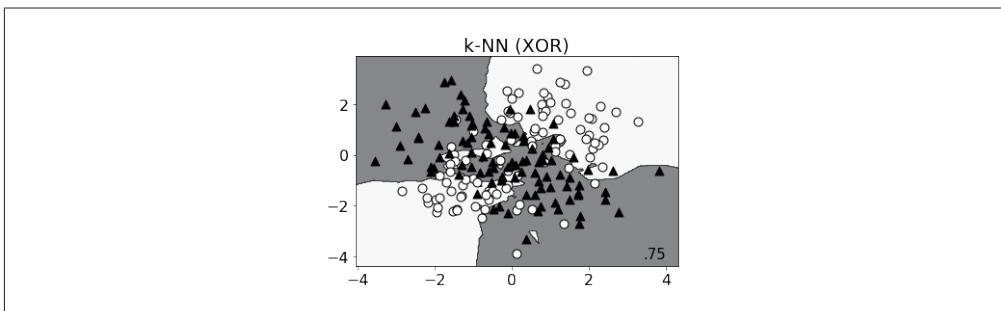
k-NNには以下のような特徴があります。

- データを1つずつ逐次学習する
- 基本的に全データとの距離計算をする必要があるため、予測計算に時間が掛かる
- k の数によるがそこそこの予測性能

ただし、例えば特徴量 A より特徴量 B が平均して10倍大きいなど、データの軸ごとのスケールが大きく違うとうまく学習できないので、特徴量間のスケールを揃えるために正規化を忘れないましょう。

k-NNの決定境界

近傍 k 個の既知データのうち、最も数が多いクラスを予測したクラスとするというアルゴリズムの性質上、 k というハイパーパラメータに応じて決定境界が変わります。 k の値が増えるほど決定境界が滑らかになりますが、処理時間が増加することに注意してください。

図2-28 k-NN ($k=3$) の決定境界 (線形分離可能)図2-29 k-NN ($k=3$) の決定境界 (線形分離不可能)

k-NNの仕組み

k は投票する個数のことを意味し、 $k = 3$ のときはデータに最も近い3点のうち、得票数が多いクラスに所属しているとみなします。図2-27の例を見てみましょう。新しいデータである■が、●と▲のどちらのクラスかということを考えます。■の周囲3つのデータを見ると●になることが分かります。もちろん、 k の数を変えると所属するクラスは容易に変わります。

k の数は交差検証で決めます。この時、どのデータが「近い」のかというのを決めるためには、「距離」を定義する必要があります。多く用いられるのは2つの点を結んだ直線の長さであるユークリッド距離 (Euclidean Distance) ですが、あるクラスのデータ群の平均からの近さだけでなく、データの散らばる方向 (分散) を考慮することができるマハラノビス距離 (Mahalanobis Distance) と呼ばれる距離が用いられることもあります。ユークリッド距離は、点の座標を表すベクトル a とベクトル b があったとき、以下のコードで求めることができます。^{†10}

```
def euclidean_distance(a, b):
    return np.sqrt(sum((x - y)**2 for (x, y) in zip(a, b)))
```

^{†10} 実際には、NumPyを使ってnp.linalg.norm(a - b)で求めたほうが高速です。

自然言語処理など、高次元で疎なデータの場合は予測性能がでないことが多いです。こうしたときには、次元削減手法で次元圧縮をすると性能が改善することが知られています。

k-NNはシンプルな手法のため、気軽に試すには良い方法です。また、距離さえ定義できれば応用が効くので、例えば Elasticsearch のような全文検索エンジンのスコアを距離とみなして k-NN を使うようなこともできます。計算時間がかかる問題については、近似的に近傍探索をするなどいくつかの手法が提案されています。

2.2.6 決定木、ランダムフォレスト、GBDT

本項では、ツリー型のアルゴリズムの代表である決定木 (Decision Tree) およびその発展形である、ランダムフォレスト (Random Forest) と Gradient Boosted Decision Tree (GBDT) について紹介します。

ツリー型のアルゴリズムは、今まで紹介した分類のアルゴリズムとは少し違いますので、仕組みと傾向を知っておくと良いでしょう。

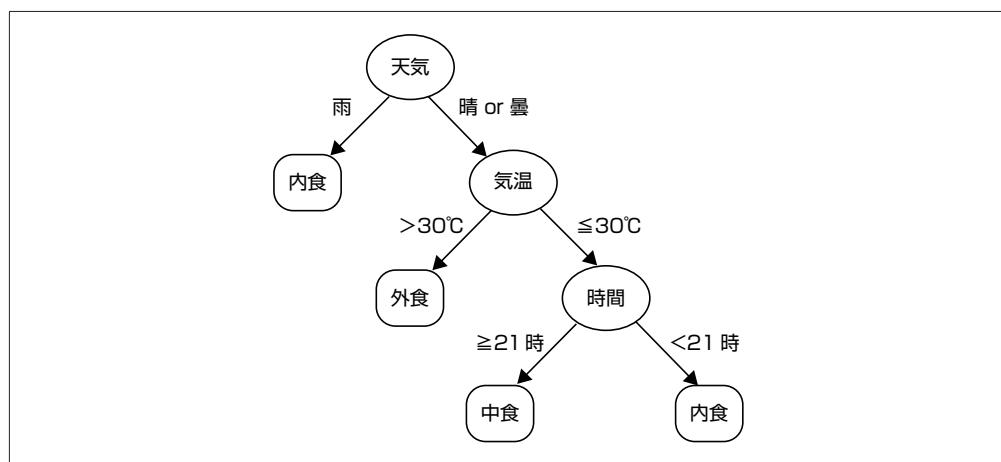


図2-30 決定木

決定木の特徴

決定木の特徴をまとめると、以下のようになります。

- 学習したモデルを人間が見て解釈しやすい
- 入力データの正規化がいらない
- カテゴリ変数やその欠損値（計測漏れなどで値が存在しない）などを入力しても内部で処理してくれる
- 特定の条件下では過学習しやすい傾向にある

- 非線形分離可能だが、線形分離可能な問題は不得意
- クラスごとのデータ数に偏りのあるデータは不得意
- データの小さな変化に対して結果が大きく変わりやすい
- 予測性能はまずまず
- バッチ学習でしか学習できない

決定木の大きな特徴は、学習したモデルを可視化して解釈しやすいという点があげられます。これは、学習結果としてIF-THENルールが得られるため、例えば工場のセンサー値から製品の故障を予測したい場合に、どのセンサーが異常の原因なのかといった、特定の分類結果に至った条件が必要とされるシーンで有効でしょう。パーセプトロンやロジスティック回帰とは異なり、線形分離不可能なデータも分類できます。一方で線形分離可能な問題はそこまで得意ではありません。また、データを条件分岐で分けていくという性質上、木の深さが深くなると学習に使えるデータ数が少なくなるため、過学習しやすくなる傾向にあります(図2-31)。これについては、木の深さを少なくしたり枝刈り(剪定、Pruning)したりすることである程度防ぐことが出来ます。



残念ながら、scikit-learn 0.18では枝刈りはまだ実装されていません。

また、特徴の数が多い時も過学習しやすくなるため、事前に次元削減や特徴選択をしておくと良いでしょう。

決定木の決定境界

決定木の決定境界は、直線にはなりません。これは、領域の分割を繰り返していく方法で決定境界を作っていくためです。ですので、線形分離可能な問題に使うよりは不可能な問題に利用したほうが良いでしょう。

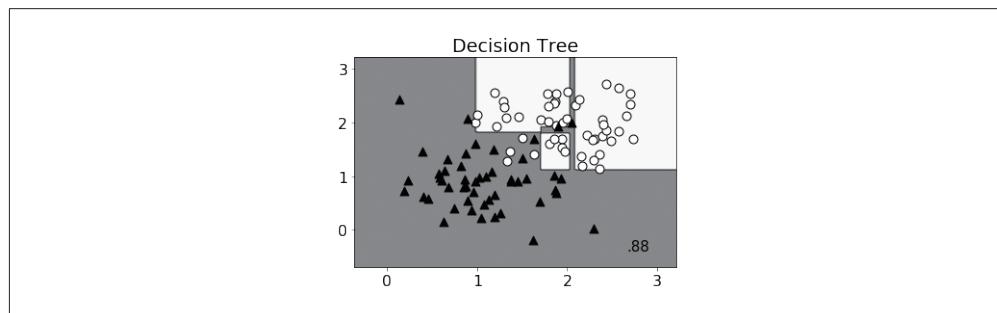


図2-31 決定木の決定境界(線形分離可能)

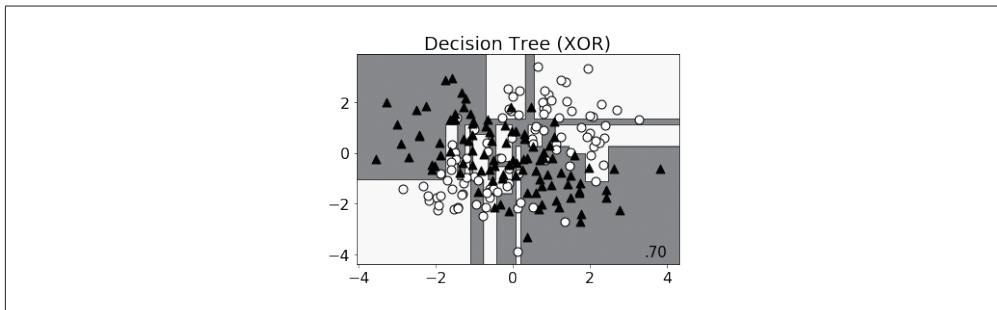


図2-32 決定木の決定境界（線形分離不可能）

決定木の仕組み

決定木は、教師データから条件式を作り、予測の際には木の根 (root node、条件式の一番上) から順番に条件分岐をたどっていき、葉 (leaf node、条件式の末端) に到達すると予測結果を返すアルゴリズムです。不純度 (Impurity) と呼ばれる基準を使って、できるかぎり同じクラスがまとまるように条件分岐を学習していきます。具体的には情報ゲイン (Information Gain) やジニ係数 (Gini Coefficient) などの基準を不純度として使い、不純度が下がるようにデータを分割します。決定木を使うことで、データからうまく分類できるようなIF-THENルールのツリーを、図2-30のように得ることができます。

決定木から派生したアルゴリズム

決定木を応用した手法に、ランダムフォレスト (Random Forest) や Gradient Boosted Decision Tree (GBDT) があります。

ランダムフォレストは、利用する特徴量の組み合わせをいくつか用意して、性能がよかった学習器複数の予測結果を多数決で統合します。複数の木を独立に学習できるため並列で学習できます。また、決定木の枝刈りをしないため主なパラメータは2つと GBDT と比べて少ないですが、過学習しやすい傾向にあります。予測性能は決定木より高く、パラメータ数が少ないためチューニングも比較的簡単で手頃です。図2-33の決定境界を見ても分かるように、決定木ベースのアルゴリズムのため傾向は似ています。

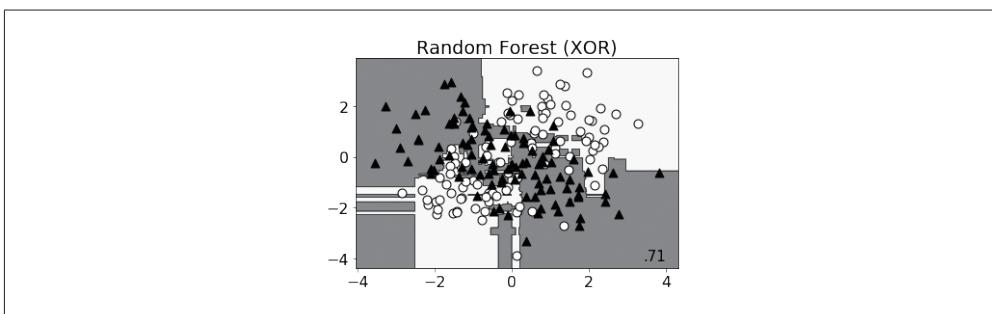


図2-33 ランダムフォレストの決定境界（線形分離不可能）

ランダムフォレストが並列的に学習して予測結果を利用するのに対して、GBDTはサンプリングしたデータに対して直列的に浅い木を学習していく勾配ブースティング法（Gradient Boosting）を使うアルゴリズムです[sgb][gb]。予測した値と実際の値のズレを目的変数として考慮することで、弱点を補強しながら複数の学習器が学習されます。直列で学習するため時間がかかることが、ランダムフォレストと比べてもパラメータが多いためチューニングにコストがかかりますが、ランダムフォレストよりも高い予測性能を得られます。XGBoost[xgboost]やLightGBM^{†11}という高速なライブラリが登場したこともあり大規模なデータでも処理しやすく、機械学習コンペサイトのKaggle^{†12}でも人気です。特にXGBoostは確率的な最適化をしているため大規模データにも高速に処理できますし、LightGBMはXGBoostより早く処理できるとうたわれています。

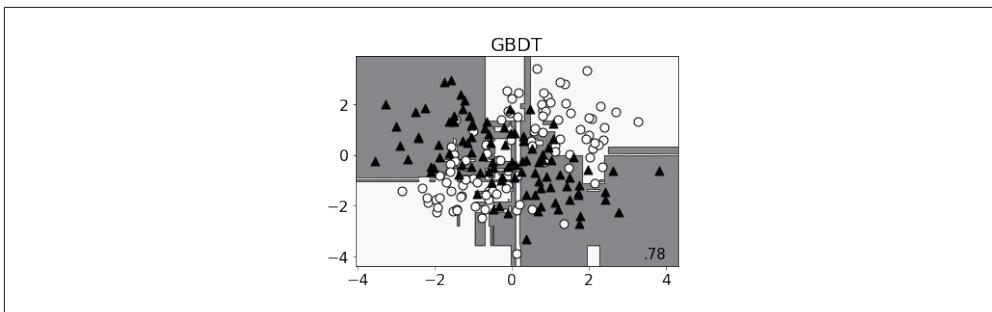


図2-34 勾配ブースティング法の決定境界（線形分離不可能）

ランダムフォレストやGBDTのように、複数の学習結果を組み合わせる手法をアンサンブル学習（Ensemble Learning）といいます。^{†13}単純な決定木はデータの追加を行うと学習結果が大きく変わること

^{†11} <https://github.com/Microsoft/LightGBM/>

^{†12} <https://kaggle.com/>

^{†13} アンサンブル学習は通常、ロジスティック回帰やルールが1つだけの決定木などのシンプルな学習器（弱学習器（Weak Learner））といいます）を複数組み合わせて学習をします。

のに対して、ランダムフォレストには学習結果が安定しやすくなるといったメリットがあります。また、予測性能もアンサンブルをしたほうがより良くなることが知られています。アンサンブル学習は、「三人寄れば文殊の知恵」ということわざのように、それぞれの予測モデルが弱点を補い合っていると考えると良いでしょう。

2.3 回帰

回帰は、教師あり学習の1つで、ある入力データから連続値を予測します。例えば、都市の電力消費量やWebサイトのアクセス数など、連続値で表現できる値を予測したい場合にモデルを学習します。回帰について各アルゴリズムのおおよその傾向について説明します。

- 線形回帰 (Linear Regression)、多項式回帰 (Polynomial Regression)
- Lasso回帰 (ラッソ回帰、Lasso Regression)、Ridge回帰 (リッジ回帰、Ridge regression)、Elastic Net (エラスティックネット)
- 回帰木 (Regression Tree)
- SVR (Support Vector Regression)

それぞれの概要は以下のとおりです。

- 線形回帰はデータを直線で、多項式回帰は曲線で近似したもの
- Ridge回帰は学習した重みの2乗を正則化項 (L2正則化) に、Lasso回帰は学習した重みの絶対値を正則化項 (L1正則化) に、Elastic Netはその両方を線形回帰に正則化項を追加したもの
- Lasso回帰やElastic Netは、L1正則化によりいくつかの重みが0になり、特徴を絞り込む性質がある
- 回帰木は決定木ベースの回帰で、非線形なデータに対してフィッティングできる
- SVRはSVMベースの回帰で、非線形なデータに対してもフィッティングできる

回帰木やSVRは、それぞれ分類器としての決定木やSVMに似た性質を持っています。線形なデータだと分かっているときには、線形回帰やそれに正則化項を追加したLasso回帰、Ridge回帰、Elastic Netなどを用い、それでうまくいかない場合は回帰木やSVRなど非線形な回帰を用いるのが良いでしょう。

2.3.1 線形回帰の仕組み

回帰の中でも、一番シンプルな線形回帰について紹介します。

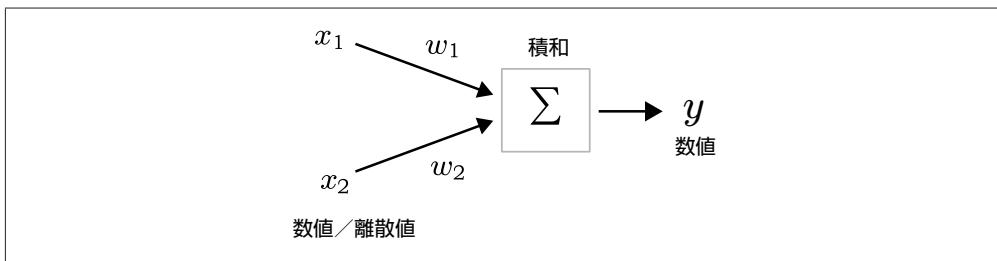


図2-35 線形回帰のイメージ図

線形回帰のイメージは図2-35のようになります。これは、パーセプトロン「2.2.1 パーセプトロン」の二值化の部分がなくなり、数値を直接出力するものと同じです。

線形回帰の目的関数は、

$$\text{目的関数} = \text{損失関数の全データでの和}$$

となっており、損失関数には二乗誤差が用いられます。

つまり入力データを、二乗誤差を最小とするような直線で近似し、その係数をパラメータとして得るのが線形回帰です。例を見ながら説明しましょう。

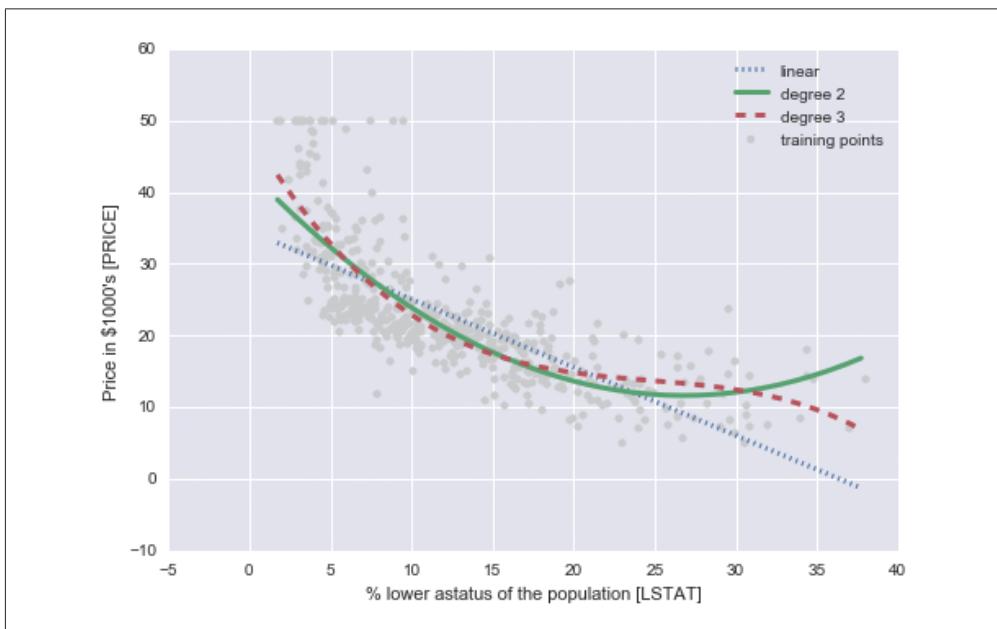


図2-36 線形回帰と多項式回帰の例

図2-36に、アメリカの家賃データに対する線形回帰及び多項式回帰の例を示します。このように、直線でざっくりとデータを表現してしまうのが線形回帰、2次曲線や3次曲線の多項式を使った曲線でデータを近似して表現するのが多項式回帰です。横軸の値を入れると直線や曲線で推定した家賃の値が帰ってくるという形になります。例えば、線形回帰で家賃と平均年収との関係性を学習するとき、 $\text{家賃} = a \times \text{平均年収} + b$ という直線の式を考えます。この時、 a と b という係数（これが線形回帰のパラメータです）を、できるだけいい感じになるように学習します。つまり、線形回帰や多項式回帰のモデルを学習して得られるのは、各変数に掛け合わせる重みになります。

2.4 クラスタリング・次元削減

本節では、クラスタリングと次元削減について説明します。

2.4.1 クラスタリング

クラスタリング (Clustering) とは、教師なし学習の1つの方法で、主にデータの傾向をつかむために使われます。似ている組み合わせを順番にまとめていく階層的クラスタリング (Hierarchical Clustering) や、距離の近いもの同士を k 個のグループに分割する k -meansなどがあります。

k -meansはクラスタリングを行う上でシンプルなのでデータの傾向を見るのによく使われます。図2-37に k -meansのイメージを示します。データ（●）の重心（▲）を見つけて、 k 個のクラスタ（かたまり）に分割します。

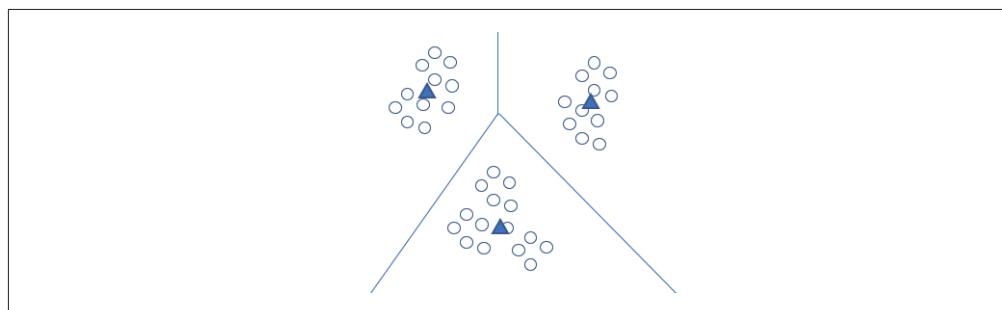


図2-37 k -meansのイメージ図($k=3$)

この他のクラスタリング手法については、scikit-learnのドキュメントに傾向がまとまっているのでご参照ください。^{†14}。

^{†14} http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

2.4.2 次元削減

次元削減 (Dimension Reduction) とは、高次元のデータからできるだけ情報を保存するように低次元のデータに変換することです。例えば、100次元のデータがあったときに、人間はそれを直接見ることはできませんが、できるだけ元の特徴を損なわないように2次元に表現できれば、何かデータの特徴が見えてくるかもしれません。前述のクラスタリングをした結果も、その傾向をつかむために次元削減をして可視化することで、データ同士の関係性を図示したりします。次元削減は、可視化だけではなく疎なデータから密なデータに変換し圧縮することもできます。また、次元削減をしたデータを更に教師あり学習の教師データにして学習をすることもあります。

手法としては主成分分析 (PCA, Principal Component Analysis) が有名ですが、最近では t-SNE[tsne] も人気です。t-SNEは特に可視化のために用いられることが多く、PCAより分かりやすく関係性が可視化できることで Kaggle でも人気です。

2.5 その他

ここまで登場しましたが、機械学習やデータマイニング関連のタスクとしてよく取り組まれるトピックについて紹介します。以下の4つは手法ではなく、分類や回帰などと同じように機械学習でできることだと考えると良いでしょう。

- 推薦 (Recommendation)
- 異常検知 (Anomaly Detection)
- 頻出パターンマイニング (Frequent Pattern Mining)
- 強化学習 (Reinforcement Learning)

2.5.1 推薦

推薦は、ユーザーが好みそうなアイテムや、閲覧しているアイテムに類似しているアイテムを提示します。ECサイトの「この商品を買った人はこれも買っています」や音楽の「このアーティストに関連するアーティスト」のように、関連するおすすめアイテムの提示がそれにあたります。ユーザーの行動履歴やアイテムの閲覧傾向をもとに、似たユーザー同士や似たアイテム同士を利用します。

詳しくは7章で説明をします。

2.5.2 異常検知

異常検知は、クレジットカードの不正な決済やDoS攻撃による不正検知など、異常を検出するためのデータマイニング手法です。外れ値検知 (Outlier Detection) とも言います。通常異常のデータの件数はとても少ないため、単純に分類モデルを学習しようとすると常に「正常」の値を出力してしまいま

す。3章で詳しく説明をしますが、偏りのあるクラスを学習する場合、例えば0.5%しか異常値が得られない場合に、その特徴を掴みきれずに全て「正常」と出力してしまうことが往々にしてあります（こうした問題を、**不均衡データ (Imbalanced Data)**と言います）。そして、全て「正常」と出力するモデルにおいて、評価にも注意を払わないと、単純に正解率を計算すると99.5%の「正常」のデータに対しては正解していることになってしまいます。データに極端な偏りがあるという特徴から、異常検知には教師なし学習が使われることが多いです。非常に幅広い内容のため詳細は他書 [[ideanomaly](#)] [[ideanomaly2](#)]に譲りますが、データ点がよく集まっている部分から離れた部分を異常値として検出するイメージです^{†15}。scikit-learnを使う場合はSVMベースの**One Class SVM**などで異常検知ができます。

2.5.3 頻出パターンマイニング

頻出パターンマイニングは、データ中に高頻度に出現するパターンを抽出する手法です。俗にいう「ビールと紙おむつがよく買われる」という例え話のように、購買情報から頻出するパターンを抽出します。有名な方法として、**相関ルール (Association Rule)**と呼ばれるものがあり、**Apriori**アルゴリズムを使って解くことができます。残念ながらscikit-learnには頻出パターンマイニングの実装はありませんが、SPMF^{†16}などのツールを使うことで試すことができます。

また、時系列の分析を行うときは、**ARIMA (自己回帰移動平均モデル)**といったアルゴリズムがよく使われます。

2.5.4 強化学習

強化学習は、経験を元に試行錯誤をし、ある目的のためにこの場合こうすれば良いといったような最適な行動の方針を獲得する方法です。他の学習とは異なり、例えば囲碁や将棋のように「ゲームに勝つ」というメタな目的に向かって何かしらの行動をとり、その行動結果の良し悪しを元に次の手を決めていきます。

よく使われる例としては、赤ちゃんが試行錯誤を繰り返すことで歩く方法を獲得するように、目的のために大量の試行を繰り返しながら最適化をしていきます。

強化学習は、自動運転やゲームAIなどの分野で非常に注目を集めており重要な機械学習の1ジャンルですが、本書では強化学習は対象としないため、興味がある人は [[reinforcement-learning](#)]などを参考にしてください。

^{†15} 単純な方法としては、データの分布から正規分布を当てはめてあげたときに、 2σ の外を異常とするといった方法が考えられます。

^{†16} <http://www.philippe-fournier-viger.com/spmf/>

2.6 この章のまとめ

本章では、教師あり学習として分類と回帰を、教師なし学習としてクラスタリング、次元削減、さらにその他のアルゴリズムについて説明しました。

教師あり学習では、決定境界を表現する関数を学習するもの、距離ベースで判断するもの、木構造のルールを学習するものを学びました。教師なし学習として、データの隠れたカテゴリを見つけ出すクラスタリングとデータの可視化を支援する次元削減について学びました。

どのアルゴリズムを選ぶかは、機械学習をする上で1つの腕の見せ所です。データの傾向を見ながらできるだけ色々なアルゴリズムを試してみることが、遠回りに見えますが成功への近道となるでしょう。

3章

学習結果を評価しよう

システムに機械学習を組み込んだときに、最初から満足できる結果が得られることはまれです。では、満足できる結果かどうかはどのように測るのでしょうか？本章では、機械学習の結果を評価する方法について解説します。

3.1 分類の評価

本節では、スパム分類の例を考えながら以下の4つの指標について紹介します。

- 正解率 (Accuracy)
- 適合率 (Precision)
- 再現率 (Recall)
- F値 (F-measure)

また、これらを考えていく上で重要な3つの概念について説明します。

- 混同行列 (Confusion Matrix)
- マイクロ平均 (Micro-average)、マクロ平均 (Macro-average)

3.1.1 正解率を使えば良いのか？

分類のタスクでは、正しく分類されたか否かで分類器の性能を見極めます。まずは一番シンプルな正解率 (Accuracy) について考えてみましょう。

正解率は

$$\text{正解率} = \frac{\text{正解した数}}{\text{予測した全データ数}}$$

として求めます。

例として迷惑メール（スパム）と普通のメール（非スパム）を分類する、メールのスパム分類について考えます。2値分類の例ですね。

届いたメール100件を人の手で確認した所、スパムの数が60件で普通のメールが40件でした。もし、すべてをスパムとする分類器があったとすると、正解率は60%です。

分類問題の場合、一般的にはランダムで出力した結果が性能の最低水準となります。2値分類の場合は2クラスがランダムに出力されるので正解率は50%に、3値分類の場合は3クラスがランダムに出力されるため平均的に正解すると正解率は33.3%になります。この60%という正解率はランダムに2クラスを予測するモデルより良い数字に見えます。でも、すべてスパムと予測しているためか、適切に評価されている気がしません。現実の問題では、分類したいクラスにはそれぞれ偏りがあることが多く、偏りがあるデータに対して単純な正解率はあまり意味を成さないことが多いです。

3.1.2 データ数の偏りを考慮する適合率と再現率

では、何に注目すれば良いのでしょうか。ここでは、適合率（Precision）と再現率（Recall）という考え方を使います。

適合率は精度とも呼ばれ、出力した結果がどの程度正解していたのかを表す指標です。再現率は、出力した結果が実際の正解全体のうち、どのくらいの割合をカバーしていたかを表す指標です。

スパム分類の例で言うと、適合率はスパムと予測したメールのうち、本当にスパムだったメールの割合です。スパムと予測したメールが80件で、実際のスパムメールが55件だった場合、適合率は次のようにになります。

$$\text{適合率} = \frac{55}{80} = 0.6785 \approx 0.68$$

ランダムの0.5よりは良い値ですが、あまり高いとはいえませんね。

再現率はどうでしょうか。再現率は、全データに含まれるスパムの数（今回の例では60件）のうち、スパムであると予測した正解（今の例だと55件）がいくつ含まれているかの割合です。

つまり、再現率は

$$\text{再現率} = \frac{55}{60} = 0.916... \approx 0.92$$

となり、こちらは比較的高い数字になっています。このような場合、再現率の方が1に近い値であることから、分類は「再現率重視」であると評価します。では、この再現率重視とはどういうことでしょうか？

適合率と再現率はトレードオフの関係となっており、問題設定によってどちらを重視するかは異なります。

見逃しが多くてもより正確な予測をしたい場合には、適合率を重視します。メールのスパム判定で言えば、重要なメールがスパムと誤判定をされて見逃されるよりは、たまにスパムがすり抜けても構わない。つまりスパムと予測したものが確実にスパムであるほうが安心できます。

一方、誤りが多少多くても抜け漏れを少なくしたいという場合には、再現率を重視します。再現率重視は、後から全データを眺めた時にどのくらい漏れが少ないかを重視する方法だとも言えます。例えば発生する件数の少ない病気の検診で、病気であると誤判定するケースが多少あっても、再検査をすればそれで良いという考え方です。

今回のスパム判定の場合、再現率重視はあまり喜しくなさそうです。

3.1.3 F値でバランスの良い性能を見る

こうしたトレードオフがあることを踏まえた上で、実際に分類器の比較をするのに使われるのがF値(F-measure)と呼ばれる、適合率と再現率の調和平均です。

F値は次のように計算します。

$$F\text{ 値} = \frac{2}{\frac{1}{\text{適合率}} + \frac{1}{\text{再現率}}} = \frac{2}{\frac{1}{0.68} + \frac{1}{0.92}} \doteq 0.78$$

再現率と適合率のバランスが良ければF値が高くなります。つまり、F値が高いとは、2つの指標がバランス良く高いことを意味しています。

3.1.4 混同行列を知る

続いて混同行列(Confusion Matrix)について解説します。分類のタスクではこの表を元に考えることが多いので、知っておくと良いでしょう。

図3-1に混同行列を示します。混同行列には2つの軸があります。1つ目の予測結果の軸では、何かしらの分類モデルが予測した結果が、陽性(Positive、スパム判定の例ではスパム)か陰性(Negative、スパム判定の例では非スパム)かということを考えます。2つ目の実際の結果の軸では、人の手で本当にスパムかどうかを確認した結果、実際の結果が陽性か陰性を考えます。

これら2つの軸を元に、表にまとめたものが混同行列です。

		予測結果が	
		Positive (スパム)	Negative (非スパム)
実際の結果が	Positive (スパム)	真陽性 (True Positive、TP)	偽陰性 (False Negative、FN)
	Negative (非スパム)	偽陽性 (False Positive、FP)	真陰性 (True Negative、TN)

図3-1 混同行列の表

「真陽性」などの言葉は、字を見てもらえば分かるように、実際の結果に対する正誤で一文字目の真偽 (True/False) が、予測結果によって二文字目の陽性／陰性が変わってきます。

先程のスパムの例を混同行列にしてみましょう。もう一度条件を確認しておくと「100件のメールのうち、実際のスパムは60件。ある分類器がスパムと予測したメールが80件、本当にスパムだったメールが55件」という条件でした。この条件を元に混同行列を作ってみましょう。

まず、スパムと予測した80件のうち、本当にスパムだったメールは55件でした。これが真陽性 (TP) の件数となります。

		予測結果が	
		Positive (スパム)	Negative (非スパム)
実際の結果が	Positive (スパム)	55	
	Negative (非スパム)		

図3-2 混同行列の例：Step 1 真陽性の件数が分かった

次に、スパムと予測した80件のうち、非スパムなのに間違えてスパムと予測したメールは $80 - 55 = 25$ 件だったことになります。これが、偽陽性 (FP) の数になります。

		予測結果が	
		Positive (スパム)	Negative (非スパム)
実際の結果が	Positive (スパム)	55	
	Negative (非スパム)	25	

図3-3 混同行列の例：Step 2 偽陽性の数が分かった

続いて、全メール100件中、人手で確認したスパムの数は60件でした。スパムと予測して正解した数が55件だったので、間違えてスパムと予測した非スパムは $60 - 55 = 5$ 件あることが分かります。これが偽陰性(FN)の数になります。

		予測結果が	
		Positive (スパム)	Negative (非スパム)
実際の結果が	Positive (スパム)	55	5
	Negative (非スパム)	25	

図3-4 混同行列の例：Step 3 偽陰性の数が分かった

最後に、非スパムと予測したのは $100 - 80 = 20$ 件ですが、本当に非スパムだったのは、間違えてスパムと予測した5件を引いた $20 - 5 = 15$ 件です。

		予測結果が	
		Positive (スパム)	Negative (非スパム)
実際の結果が	Positive (スパム)	55	5
	Negative (非スパム)	25	15

図3-5 混同行列の例：Step 4 真陰性を入れて完成

ここで、この混同行列を元にして適合率、再現率を再び計算してみましょう。適合率は予測結果がどのくらい正しく正解しているか、再現率は本当にスパムと確認されたメールのうちどのくらいをカバーできているかを示す値でした。

$$\text{適合率} = \frac{TP}{TP + FP} = \frac{55}{55 + 25} \doteq 0.69$$

$$\text{再現率} = \frac{TP}{TP + FN} = \frac{55}{55 + 5} \doteq 0.92$$

比較のために、正解率を計算してみましょう。正解率は以下のようになります。これは両方のクラスをまとめて見ているため、かなりざっくりとした指標でしかないことが改めて分かると思います。

$$\text{正解率} = \frac{TP + TN}{TP + FP + TN + FN} = \frac{55 + 15}{55 + 5 + 25 + 15} = 0.7$$

なお、scikit-learnには、混同行列を計算する`confusion_matrix`という関数が用意されています。混同行列を計算するコードの一例を下記に示します。`print(cm)`で出力された行列は、先ほどの表と同じ順番で混同行列の値が出力されています。

```
from sklearn.cross_validation import train_test_split
from sklearn.metrics import confusion_matrix

# データと正解ラベルを学習用とテスト用に分割する
data_train, data_test, label_train, label_test = train_test_split(data, label)

# 何かしらの予測をする。例として線形SVMで予測している
classifier = svm.SVC(kernel='linear')
label_pred = classifier.fit(data_train, label_train).predict(data_test)

# 混同行列を計算する
```

```
cm = confusion_matrix(label_test, label_pred)
print(cm)
# [[55  5]
# [25 15]]
```

実際に混同行列を作るとときは、`confusion_matrix`関数を使い交差検証を行います。これは、訓練データで予測モデルを構築をし、教師ラベル付きの検証データも元に混同行列を作ることで、モデルの比較・評価が自動でできるのです。

3.1.5 多クラス分類の平均のとり方：マイクロ平均、マクロ平均

なお、多クラス分類の場合は、クラス全体の平均を取る方法が2種類あります。1つ目のマイクロ平均は、全てのクラスの結果をフラットに評価します。例えば3クラス分類における適合率のマイクロ平均 (**Micro-average**) は、それぞれのクラスの真陽性・偽陽性の数を $TP_1, FP_1, TP_2, FP_2, TP_3, FP_3$ とすると、以下のように求めます。

$$\text{適合率}_{\text{マイクロ平均}} = \frac{TP_1 + TP_2 + TP_3}{TP_1 + TP_2 + TP_3 + FP_1 + FP_2 + FP_3}$$

これに対し、もう1つのマクロ平均 (**Macro-average**) はクラスごとに適合率を計算し、最終的な評価値は適合率の合計をクラス数で割るという方法で算出します。

例として3クラス分類のマクロ平均を考えてみましょう。各クラス1、2、3に対してそれぞれクラス1か否かといった分類結果の混同行列を作り、適合率をそれぞれ計算します。計算した適合率はクラスごとに適合率₁、適合率₂、適合率₃だったとすると、適合率のマクロ平均は以下のようになります。

$$\text{適合率}_{\text{マクロ平均}} = \frac{\text{適合率}_1 + \text{適合率}_2 + \text{適合率}_3}{3}$$

マイクロ平均はクラスをまたいだ全体のパフォーマンスの概要を知るのに向いています。クラスごとにデータ数の偏りがある場合は、マクロ平均を用いたほうが偏りの影響を考慮した評価ができるでしょう。

3.1.6 分類モデルを比較する

モデル同士の性能を比較する場合には、データに偏りがある場合も多いためF値をもとにすることが多いです。実問題を解くときには、適合率を重視するのか再現率を重視するのかを考えて、最低限保証したい方の値を決めてからチューニングするのが望ましい方法です。

誤りが許されない問題の場合、例えば「適合率が0.9以上にならないモデルは採用しない」という最低ラインを決めて、その上でF値が高くなるようにパラメータチューニングをして、モデル選択をするのが良いでしょう。

また、本書では詳しく扱いませんが、F値以外にもROC曲線(Receiver Operating Characteristic Curve)やそれに基づくAUC(Area Under the Curve)などの評価値が使われることもあります。

分類の性能はあくまでもビジネスに応用する上での品質を保つための最低基準だと思ったほうが良いです。筆者もたまに陥るのですが、性能のチューニングにどっぷりハマっていると、それ自身が目的となってしまいがちです。学習モデルの性能が高いこととビジネスゴールを満たすことは別の問題なので、そのモデルで何が実現したかったのかを考える癖をつけるようにしましょう。

そのためには、プロダクトをリリースするために必要な性能を決めた後も、最終的なゴールとしての数値を満たせるかどうかを観測し、継続的に改善できる仕組みを導入しましょう。

3.2 回帰の評価

3.2.1 平均二乗誤差

回帰は電力消費量や価格など、連続する数値を予測する問題でした。回帰の評価では主に平均二乗誤差(Root Mean Squared Error、RMSE)を使います。

数式で書くと以下になります。

$$RMSE = \sqrt{\frac{\sum_i (\text{予測値}_i - \text{実測値}_i)^2}{N}}$$

予測した値と実測の値の2つの配列について、各要素の差を2乗した値を総和し、それを配列の要素数で割った値の平方根を取ります。

コードで書くと次のようになります。

```
from math import sqrt

sum = 0

for predict, actual in zip(predicts, actuals):
    sum += (predict - actual) ** 2

sqrt(sum / len(predicts))
```

実際には、scikit-learnにmean_squared_errorという関数が用意されているのでそれを使えば良いでしょう。

```
from sklearn.metrics import mean_squared_error
from math import sqrt

rms = sqrt(mean_squared_error(y_actual, y_predicted))
```

実は回帰分析で全データの平均値を出力する予測モデルを考えたとき、その平均二乗誤差はデータのばらつき具合を表す**標準偏差 (Standard Deviation)**となるのです。

そのため予測モデルの出力する値の平均二乗誤差を比較し、最低限のベースライン（分類問題で言うところのランダム出力をする予測モデル）として標準偏差と比較することで、予測モデルの良し悪しを検討できます。

そのfor文、本当に必要？

先ほどの平均二乗誤差の計算の説明の時に、`for`文を使って誤差の計算をしました。上の例では、`actual`、`predict`のデータ量が少なければ大きな問題はありませんが、データが大きくなればなるほど計算に時間がかかるようになります。

Pythonなどの処理系ではそのまま計算を`for`文を使って自分で実装するのではなく、それに特化したNumPyやSciPyといった計算用のライブラリを使った方がはるかに高速に計算できます。こういった計算用ライブラリがどうして速いかというと、Pythonのコードの裏側で行列を高速に計算するCやFortranのライブラリに処理を任せているからです。つまり数値計算を実行するときには、できるだけPythonで処理を行わせないようにすることが高い速度を保つうえで重要なともいえます。

ちなみに、Pythonと同じように統計解析に良く使われるR言語も、`for`文を使った表現が非常に遅くなることが知られています。これはNumPy、SciPyと同じように数値計算を裏側のライブラリに任せているためです。

数値計算を得意とするJulia言語は、ソフトウェアエンジニアに馴染みやすいようにと行列計算も`for`で書けるという特徴を持っていますが、多くの言語でこうした演算は裏側のライブラリに任せた方が高速で実行できるでしょう。

3.2.2 決定係数

平均二乗誤差は別に、回帰した方程式の当てはまりの良さを表現する**決定係数 (Coefficient of Determination)**という評価指標もあります。決定係数は、数式では R^2 と表現されます。一般的には以下ののような数式で表現されます。

$$\text{決定係数 } (R^2) = 1 - \frac{\sum_i (\text{予測値}_i - \text{実測値}_i)^2}{\sum_i (\text{予測値}_i - \text{実測値の平均値})^2}$$

これは、平均二乗誤差の分母を平均値と予測値の差の二乗の総和で割った値を、1から引いたものです。決定係数は、常に平均値を出力する予測モデルに比べて、相対的にどのくらい性能が良いのかを表します。決定係数の値が1に近ければ近いほど良い性能であることを示し、0に近ければあまり良くない性能であることを示します。

scikit-learnには`r2_score`という関数があるので、それを使って簡単に計算できます。

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x, y)

from sklearn.metrics import r2_score
r2 = r2_score(y, lr.predict(x))
```

また、回帰モデルの場合は`lr.score(x, y)`を使うと決定係数を得ることができます。

3.3 機械学習を組み込んだシステムのA/Bテスト

通常の機械学習で得られたモデルの評価より少し範囲の広い話ですが、WebサービスではA/Bテストと呼ばれるテストを良く行います。^{†1} 例えば、サービスの登録ボタンの色や文言を少しずつ変えて比較して、良かったパターンを採用するというようなテストです。

A/Bテストを行うメリットは、一定数のユーザーにパターンを出し分けることで、同一期間内に比較ができます。

期間をずらして比較をすると、季節などの影響が大きく意味を成さないことがあります。例えば、おもちゃを販売するサイトでクリスマスには購入ボタンを赤くするのが良かったとして、クリスマス以降もそのボタンが有効でしょうか？一年中赤いボタンにするのは、季節性の要素を無視しており意味がなさそうに思えます。

デザインや文言だけでなく機械学習のモデルを検証する際にも、こうしたA/Bテストは有効です。多くの場合、オフラインで評価できるのは適合率や再現率などの指標であり、実際に購入や登録などのコンバージョンに至ったかどうかという、ビジネス的なKPIは別途追いかける必要があります。また、想定していなかったトレンドの変化など、事前に考慮していない要因が大きく影響をあたえる場合もあります。そこでオンラインでの評価として、モデルを適用しない場合のパフォーマンスや、複数のモデルによる予測結果を使った場合のパフォーマンスを比較することで、より最適なモデルを選択することができます。

また、A/Bテストできるようなシステム構成にしておくことで、副次的に新しいモデルの段階的リリースや切り戻しも可能となります。そうすることで、検証のイテレーションサイクルを素早く回すこ

^{†1} A/Bテストと言いますが別に2パターンだけではなく、複数パターン行うことが多いです。

とが可能となりますし、機会損失を抑えることができます。

オフラインで複数のアルゴリズムやパラメータチューニングを行ったモデルを用意し、A/Bテストで選別をし、更に良いモデルを作成しオフラインで検証し、A/Bテストに投入する、という検証のサイクルを回していくのが良いでしょう。詳しくは、6章を参照してください。

3.4 この章のまとめ

本章では学習結果の評価方法について学びました。

分類の指標として、正解率や適合率、再現率、F値について学びました。実際には混同行列を見ながら、どのクラスがどれくらいの性能であれば良いかを考えていくのが重要です。

回帰の評価指標として、平均二乗誤差と決定係数について学びました。どちらの指標を用いても良いですが、何を基準とするかという点について常に気をつけましょう。

また、機械学習におけるA/Bテストの重要性も学びました。特に、機械学習の評価指標の良し悪しと、ビジネス上のゴールとしてのKPIの良し悪しとは別になってきます。この2つの違いを常に意識することで、予測モデルの評価指標のみを追いかけてしまわないように気をつけましょう。オフラインで評価指標の目標を達成することは、機械学習を使ったビジネスのスタートラインに立つための最低条件です。

4章

システムに機械学習を組み込む

機械学習をシステムに組み込むにはどうすればいいでしょうか。本章では、機械学習を組み込むシステムの構成とそれに深く関わる教師データを獲得するためのログ収集方法について説明します。

4.1 システムに機械学習を含める流れ

「1.2 機械学習プロジェクトの流れ」でも書きましたが、システムに実際に機械学習を適用する際には、以下のような流れで進めていきます。

1. 問題を定式化する
2. 機械学習をしないで良い方法を考える
3. システム設計・誤りをカバーする方法を考える
4. アルゴリズムを選定する
5. 特微量、教師データとログの設計をする
6. 前処理をする
7. 学習・パラメータチューニング
8. システムに組み込む

本章では、この中でも「システム設計」「ログ設計」について説明をしていきます。

4.2 システム設計

機械学習にはいくつかの種類がありますが、ここでは最も活用ケースが多い教師あり学習について、システムに組み込む場合の構成を説明します。

分類や回帰などの教師あり学習の場合、学習と予測の2つのフェーズがあります。更に学習のタイミングによって、バッチ処理での学習とリアルタイム処理での学習という二種類のタイミングがあります。

本節で、それぞれの場合のシステム構成とそのポイントについて学びますが、まずその前に重要でありながら混乱しがちな用語について整理しておきましょう。

4.2.1 混乱しやすい「バッチ処理」と「バッチ学習」

機械学習において、「バッチ」という言葉は特別な意味を持ちます。いわゆるバッチ処理と語源は同じですが、多くの場合、機械学習の文脈で「バッチ」というと「バッチ学習」のことを指します。ここでは「バッチ学習」と一般的な「バッチ処理」との違いについて説明します。

本章では、バッチ処理の対義語をリアルタイム処理と呼ぶことにします。バッチ処理は一括で何かを処理すること、またその処理そのものを指します。対してリアルタイム処理は、刻々と流れてくるセンサーデータやログデータに対して逐次処理をすること、と本書では定義します。^{†1}

なお、本章では「バッチ処理」との混同を避けるために、これ以降はバッチ学習を一括学習、オンライン学習を逐次学習と表現します。

一括学習と逐次学習とでは、モデル学習時のデータの保持の仕方が異なります。一括学習では、重みの計算のためにすべての教師データを必要とし、全データを用いて最適な重みを計算します。一般的に一括学習の場合、教師データが増えると必要とするメモリ量はその分増加していきます。例えば、求める重み w_{target} がすべての重みの平均だったとします。重みが w_1 から w_{100} まで 100 個ある時の平均を一括学習で求めるには以下のように計算します。

```
sum = w_1 + w_2 + w_3 + ... + w_100
w_target = sum / 100
```

一方、逐次学習では教師データを1つ与えて、その都度重みを計算します。例えば、次のような平均の求める処理をする場合、メモリに保持されるデータはその時のデータと、計算された重みのみになります。ある時点での重みが w_{tmp} としたとき、平均を計算するのに必要なのは総和 sum と、要素の数 cnt だけです。コードで表現すると以下のようになります。

```
sum = 0
cnt = 0
while has_weight():
    w_tmp = get_weight()
    sum += w_tmp
    cnt += 1

w_target = sum / cnt
```

繰り返しになりますが、一括学習と逐次学習では学習時の必要とするデータの塊が違います。つまり

^{†1} 「リアルタイム処理」というと、「何 ms で処理ができるの?」と思われる方もいるかもしれません、本書では便宜上速度に関係なく逐次的な処理をすることをリアルタイム処理と呼んでいます。

り、学習時の最適化の方針が違うだけなのです。^{†2}

では、バッチ処理は何を処理するのでしょうか？実は「バッチ処理」というだけでは、特に規定されていません。機械学習の文脈では学習をすることもあるし、予測をすることもあります。リアルタイム処理も同様に、学習をすることもありますし予測をすることもあります。

ここで問題です。以下の組み合わせの中で、取りうる処理と学習の組み合わせはどれでしょうか？

1. バッチ処理で一括学習
2. バッチ処理で逐次学習
3. リアルタイム処理で一括学習
4. リアルタイム処理で逐次学習

良くある誤解は「一括学習はバッチ処理でしかできず、逐次学習はリアルタイム処理でしかできない」というものです。実は3以外はすべてあります。1と4について、特に違和感はないかもしれません。では、2の「バッチ処理で逐次学習」とはどのようなものでしょうか？逐次学習は、最適化時にデータを1レコードずつ処理をする最適化方針だと説明しました。つまりバッチ処理でまとまったデータを一括処理をするけれど、最適化方針は逐次学習するということはあり得るのです。

予測フェーズについては、学習フェーズでの最適化の方針や処理方法にかかわらずバッチ処理での予測もリアルタイム処理での予測も共に存在します。

実際に学習をする際は、データを保持できない場合を除いて学習フェーズはバッチ処理であるのが試行錯誤しやすくて良いでしょう。

ここからは、バッチ処理で学習を行う3つの予測パターンとリアルタイム処理のパターンについて構成を見て行きましょう。以下にパターンを列挙します。

1. バッチ処理で学習+予測結果をWebアプリケーションで直接算出する（リアルタイム処理で予測）
2. バッチ処理で学習+予測結果をAPI経由で利用する（リアルタイム処理で予測）
3. バッチ処理で学習+予測結果をDB経由で利用する（バッチ処理で予測）
4. リアルタイム処理で学習をする

4.2.2 バッチ処理で学習+予測結果をWebアプリケーションで直接算出する（リアルタイム処理で予測）

3つの予測パターンの中で、最も素朴な方法がこのパターンです。このパターンは、バッチ処理で一

^{†2} なお、一括学習と逐次学習の中間となるミニバッチ学習(mini-batch training)という方法もあります。ある程度のデータをサンプリングしたグループを作り、このグループに対する一括学習を繰り返します。確率的勾配法(SGD)が有効だと知られてから急速にミニバッチ学習が広まりました。深層学習ではミニバッチ学習が使われることが主流です。

括学習をし、そこで得られた予測モデルをWebアプリケーションでリアルタイム処理で利用するというものです。monolithic^{†3}なWebアプリケーションに予測処理を組み込み、予測結果はライブラリのAPIから取得し、それをWebアプリケーションに渡します(図4-1)。

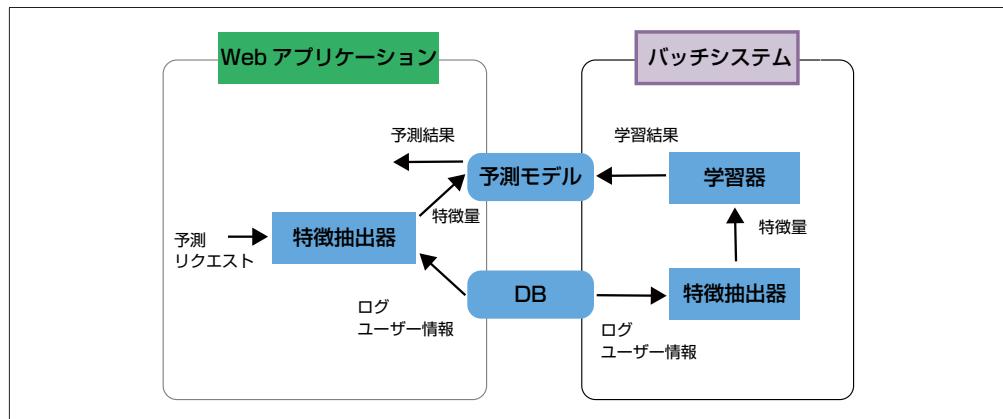


図4-1 パターン1: バッチ処理で学習したモデルを直接Webアプリケーションで使う

このパターンの特徴としては

- 予測はリアルタイム処理が必要
- Webアプリケーションと機械学習をするバッチシステムの言語が同一

という2点があります。

このパターンは比較的単純な構成のため試すのも容易で、小規模で試してみるのに適したパターンです。

バンディットアルゴリズムを用いた広告配信の最適化など、入力データが事前に用意できず、予測の結果を低レイテンシに使いたい場合にもこの構成が取られます。レイテンシを抑えるためには、データのフェッチ、前処理、特徴抽出、予測といった一連の処理が低レイテンシで完結することが望ましいのです。そのためRDBやKey Value Storeなどのデータベースに、予め前処理済みのデータや、特徴抽出の工程を減らすためにある程度の段階までの処理を終えた特徴量を格納しておくなどの工夫を施します。また、予測モデルもメモリに容易に載せられるサイズで、予測処理の負荷が低いアルゴリズムというように、空間計算量、時間計算量の観点でコンパクトなモデルが望ましいでしょう。

一方で、機械学習の処理部分とWebアプリケーションが密結合になりやすいという側面があります。アプリケーションが大規模になると、コード変更やデプロイのコストが増えて、機械学習部分の開発も

^{†3} バッチシステムとWebアプリケーションなど複数の機能や役割をまとめた一つの大きなシステムをとるアーキテクチャ

保守的になります。

この制約を嫌って、機械学習のプロトタイプはPythonで行い、Webアプリケーションで利用しているJavaScriptやRubyへ予測ロジックを移植したり、C++で書かれたライブラリのバイナリングを作成したりする場合もあります。

このシステム（図4-1）では、Webアプリケーションとバッチシステムが同じ言語で組まれています。ほとんどの部分をWebアプリケーションとバッチシステムで共有して使いまわします。1つのDBから取得したログやユーザー情報から（図では別のモジュールになっていますが）共通の特徴抽出器を用いて特徴量を抽出します。「1.2.5 特徴量、教師データとログの設計をする」で学んだ通り、特徴抽出器は、テキストなどの情報から学習器が理解できる形に変換をする部分でした。この特徴抽出の処理が異なると、いかに同じ学習済みモデルを用いても同じ予測結果にはなりません。



ログ設計については「4.3 ログ設計」で詳しく説明します。

学習フェーズ（図4-2）では、バッチシステムによってDBから予め蓄積されたログやユーザー情報を取得し、特徴量を抽出します。ここで得た特徴量をもとに何らかのモデルを学習します。学習結果は学習済みモデルをシリアル化して保存したものをストレージに保持します。

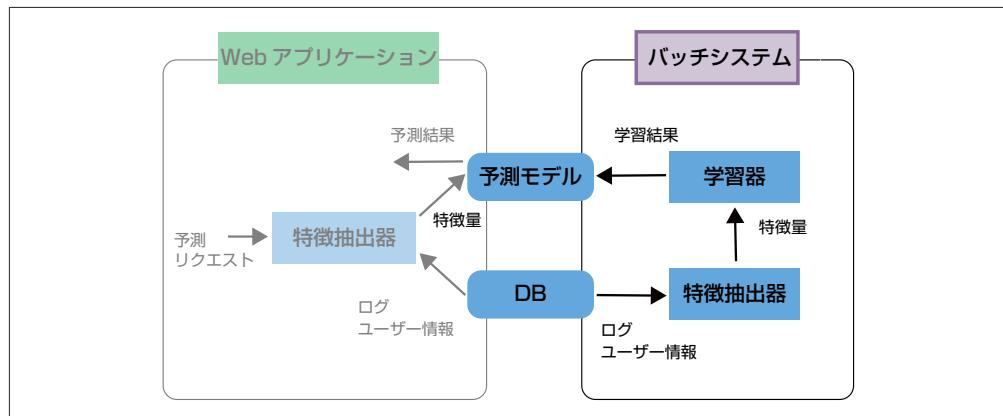


図4-2 パターン1: 学習フェーズ

予測フェーズ（図4-3）になると、Webアプリケーションが何かしらのイベントをトリガーに予測を要求します。例えば、スパムかどうかを判定したいコメントが投稿されたとします。イベント発生時に予測したい対象（例：コメント）の情報をDBから（あるいはリクエスト情報から直接）取得し、特徴量

を抽出します。シリアル化して保存していた学習済みモデルを読み込み、抽出した特徴量を入力して予測結果（例：スパム/非スパム）を出力します。その結果を元に、ユーザーにフィードバックをするなどして、次の処理へつなげていきます。

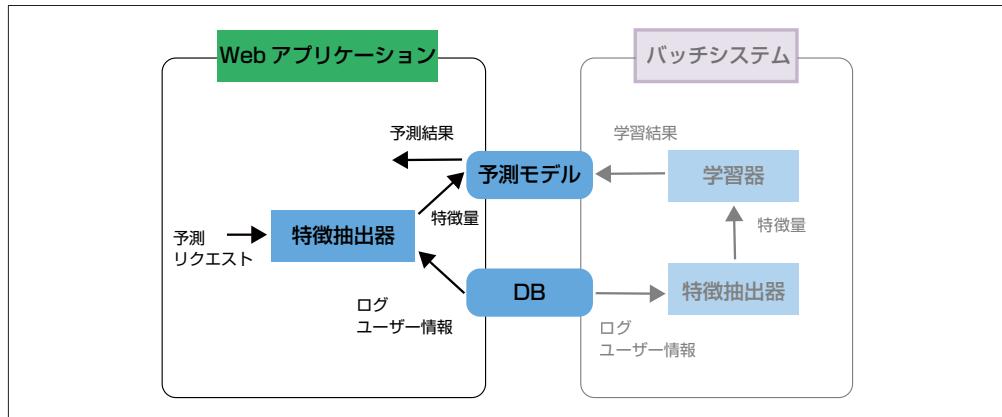


図4-3 パターン1: 予測フェーズ

4.2.3 バッチ処理で学習+予測結果をAPI経由で利用する (リアルタイム処理で予測)

Web アプリケーションとは別に、予測処理を薄くラップしたAPIサーバーを用意するのがこのパターンです（図4-4）。このパターンでは、バッチ処理で学習を行うことは他のパターンとは代わりませんが、Web アプリケーションから予測結果を利用する場合にはAPI経由のリアルタイム処理で予測を行います。HTTPやRPCのリクエストに対して、予測結果をレスポンスとして返すAPIサーバーを用意するのが特徴です。

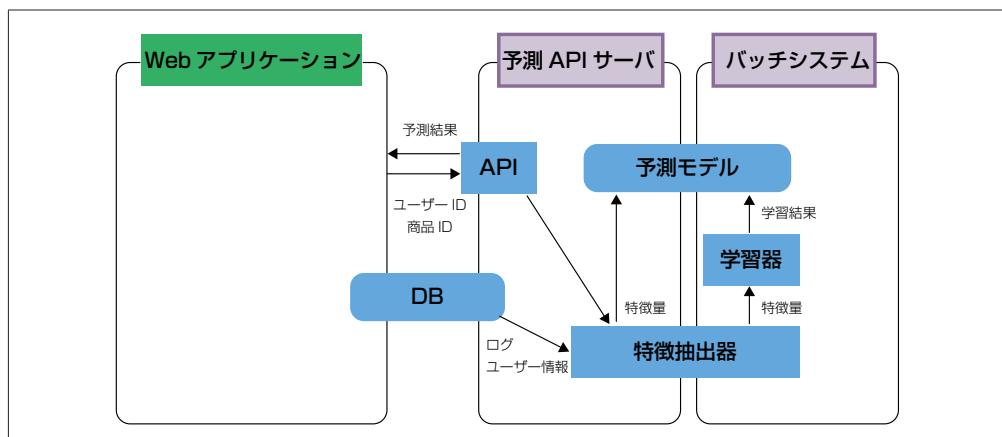


図4-4 パターン2: API を介してバッチ処理で学習した予測結果を取得する

このパターンの特徴は、

- Web アプリケーションと機械学習に使うプログラミング言語を分けられる
- Web アプリケーション側のイベントに対してリアルタイム処理で予測できる

というものです。

機械学習環境を自由に選べるためプロトタイピングを高速に回せる反面システムの規模が大きくなるため、リアルタイム処理での予測が重要でない場合には取りづらい構成です。scikit-learnなどのライブラリを使って構築するには自前でAPIサーバーを実装し、予測サーバの前にロードバランサを配置し、負荷に応じて予測サーバを増減できるようにするといった、スケールするような工夫が必要です。もし、手軽に試してみたい場合は、Azure Machine Learning、Amazon Machine Learningなどの機械学習サービスや、Apache PredictionIO (incubating)などの予測サーバーまで含んだフレームワークを利用するという方法もあります。最近ではAWS Lambdaを使った予測APIを作ることで、イベント駆動でスケールしやすい予測を行うことも容易になってきました。また、予測モデルをAmazon S3などのオブジェクトストレージに格納し、APIサーバのDocker image作成することで、Amazon EC2 Container ServiceやGoogle Container Engineを使い、スケールしやすい構成を組むこともやりやすくなっています。

このパターンを採用すると、Web アプリケーションとの結合が疎になることから、学習に使うアルゴリズムや特徴量を変えた複数のモデルによるA/Bテストを行う場合に、モデル間の比較がしやすいといったメリットもあります。

ただし、パターン1に比べるとAPIサーバと予測結果を利用するクライアントの間で通信が発生する分、レイテンシが大きくなることに注意してください。そのため、レイテンシをより小さくしたい場合は、HTTPやRPCなどのリクエストを投げる部分を非同期処理にして、予測処理の結果を待つ間に他の処理を並列で進めるなどの工夫を行うと良いでしょう。

4.2.4 バッチ処理で学習+予測結果をDB経由で利用する (バッチ処理で予測)

Web アプリケーションで使い勝手の良いのはこのパターンです。一番はじめに試すパターンとしてはこの方法が無難でしょう。

分類問題などについて教師あり学習のモデルを一括学習し、そのモデルを使った予測をバッチ処理で行い、その予測結果をDBに格納するという方法です。

このパターンは、予測バッチとアプリケーションの間でDBを介してやりとりをするため、Web アプリケーションと機械学習の学習・予測を行う言語がそれぞれ異なっていても良いことが大きなメリットです。また、後述するAPIパターンとは異なり、予測の処理に多少時間がかかる場合でもアプリケーションのレスポンスに影響しません。

このパターンの特徴は、以下の通りです。

- 予測に必要な情報は予測バッチ実行時に存在する
- イベント（例：ユーザのWebページ訪問）をトリガーとして即時に予測結果を返す必要がない

具体的には、商品説明など変化のしにくいコンテンツを6時間毎のバッチで分類する、ある日のユーザー閲覧履歴からどのユーザークラスターに所属するかを日次バッチで処理する、といったように、予測の頻度がおよそ一日一回以上（短くても数時間に一回）程度で問題のない対象や結果に向いています。例えば、ユーザのアクセスログからメールマガジンで送付する内容をパーソナライズする、などがこれに該当します。

このパターンのシステム構成は図4-5のようになります。Webアプリケーションと機械学習を行うバッチシステムとのやりとりはDBのみを介して行うため、両者のシステムに言語的な依存関係は特に発生しません。つまりWebアプリケーションでRuby on Railsを使っていましたとしても、特に気にすることなくPythonやRでバッチを書けるため、アルゴリズムの選定や特徴選択など機械学習の試行錯誤のサイクルがより高速に回せます。

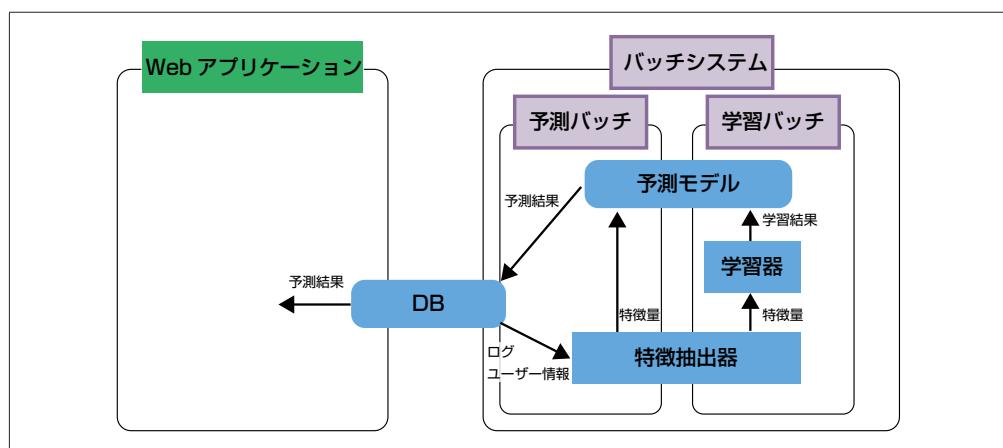


図4-5 パターン3: DBを介してバッチ処理で学習した予測結果を取得する

学習フェーズ（図4-6）では、ログやユーザー情報から特徴を抽出してモデルを一括学習します。ここで構築した学習済みモデルは、シリアル化してストレージに保持し予測フェーズで使用します。

学習バッチの実行間隔は予測の間隔よりも広くとなります。再学習を行う間隔は、予測対象がどの程度変化するかに依存します。定期的に再学習する場合は、「1.2.8 システムに組み込む」で紹介した、ゴールドスタンダードを利用するなど、学習しなおした後に精度が低下していないことを確認する工夫が必要です。

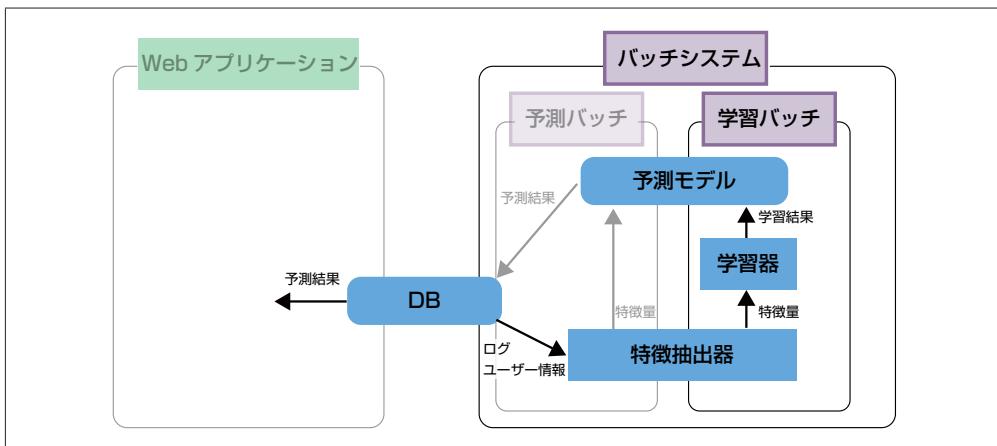


図4-6 パターン3: 学習フェーズ

予測フェーズ(図4-7)では、学習バッチで作成したモデルを用いて予測を行います。学習バッチと同様の特徴抽出器を用いて、DB中のデータから特微量を抽出し、予測します。予測結果はWebアプリケーションが利用できる形にしてDBへ格納します。

このパターンは他のパターンに比べ、予測にかけられる時間に余裕があるのが特徴ですが、予測対象となるコンテンツが増えていくと、それに比例して処理時間も増えていきます。そのため、全コンテンツに対して予測し直すようなバッチの組み方をすると、データ量の増加に対して処理時間が予想以上に膨らんでしまい、日次のジョブでは終わらないようなことが起きるので注意が必要です。

特に、モデルを頻繁に再学習したり、使用する特微量やアルゴリズムを変えて複数モデルを作成したりする場合は予測にかかる時間に十分注意する必要があります。もし、データの特性がそこまで大きく変化しないことが保証されているあれば、新規で登録された差分のコンテンツに対してのみ予測を行うという戦略を取ることもできます。すべてのデータに対して予測し直す必要がある場合は、並列数を増やして予測処理を行うか、Sparkなどの分散処理が可能な環境で予測するのが良いでしょう。

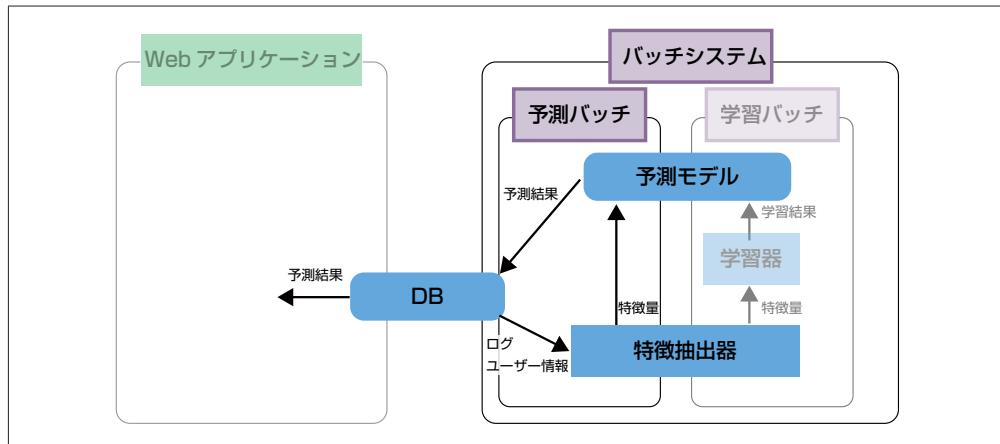


図4-7 パターン3: 予測フェーズ

4.2.5 リアルタイム処理で学習をする

「4.2.1 混乱しやすい「バッチ処理」と「バッチ学習」」では、リアルタイム処理での学習はないと言いましたが、実は全くないわけではありません。リアルタイム処理で学習が必要な場合とは、どのような場合でしょうか。

バンディットアルゴリズムなど一部のアルゴリズムやリアルタイムレコメンドでは、リアルタイム処理を使って即時にパラメータの更新が必要となる場合があります。その場合、メッセージキューなどを使い入出力のデータをやりとりします。しかし分類や回帰などで、そこまで即時にモデルを更新する必要がある場合は多くないでしょう。

もし、ある程度短い間隔でモデルを更新する必要がある場合は、1時間おきなど任意のタイミングで蓄えたデータに対してバッチ処理で学習をし、最適化方針は追加学習のできるミニバッチ学習を採用するという方法が良いでしょう。

リアルタイムレコメンドの構成としては、Oryx^{†4}という、リアルタイムの更新を分散メッセージキューであるApache Kafkaと組み合わせたフレームワークがあります。こちらのアーキテクチャが参考になるでしょう。また、Jubatus^{†5}という逐次学習向けのフレームワークも、このパターンで使われることを想定したものになります。

4.2.6 各パターンのまとめ

各パターンの特徴を表4-1にまとめます。

^{†4} <https://github.com/OryxProject/oryx>

^{†5} <http://jubat.us/ja/>

表4-1 システム構成のパターンのまとめ

パターン	一括学習+直接予測	一括学習+API	一括学習+DB	リアルタイム
予測	リクエスト時	リクエスト時	バッチ	リクエスト時
予測結果の提供	プロセス内API経由	REST API経由	共有DB経由	MQ経由
予測リクエストから結果までのレイテンシ	○	○	○	○
新規データ取得から予測結果を渡すまでの時間	短	短	長	短
一件の予測処理にかけられる時間	短	短	長	短
Webアプリケーションとの結合度	密	疎	疎	疎
Webアプリケーションのプログラミング言語と	同一	独立	独立	独立

選択の際には、特にWebアプリケーションと独立した機械学習のライブラリが充実した言語での開発と、データ取得から予測結果を返すまでのサイクルの時間のトレードオフが重要になってきます。

開発のスピードと処理速度のトレードオフを考え、適切なパターンを選んで下さい。

Pythonで学習したモデルをPython以外で利用する

豊富なアルゴリズムやユーザーの多さから、いまやデファクトスタンダードとなっているscikit-learnですが、scikit-learnで学習をしたモデルを別の言語で扱えるようにするという事例もあるようです。筆者が知る範囲では、SwiftやJavaScriptでモデルを扱えるようにしたという事例があります。

後者については実装した人に直接聞いたところ、Webアプリケーション側のコードがNode.jsだったため、決定木やロジスティック回帰といったアルゴリズムをNode.jsで再実装しているとのことです。^{†6} 機械学習のライブラリを実装するのは普通のプログラミングよりバグの検証が困難であり、なかなか厳しい道だと思います。

こうした問題を解決するため、PMML^{†7}やPFA^{†8}といった、言語やフレームワークをまたいでモデルをインポート／エクスポートするための規格もありますが、サポートしているフレームワークも限定的で2017年の段階で銀の弾丸とはなり得ていません。

またTensorFlowはPythonのAPIも備えたフレームワークですが、iOSやAndroidでも学習済みのモデルを使うことができます^{†9}。今後、フレームワークレベルで複数のプラットフォームをサポートするソフトウェアが増えてくるかもしれません。またAppleは、iOS 11からCore

^{†6} <http://www.slideshare.net/TokorotenNakayama/mlct>

^{†7} <http://dmg.org/pmml/v4-3/GeneralStructure.html>

^{†8} <http://dmg.org/pfa/index.html>

^{†9} <https://www.tensorflow.org/mobile/>

MLと呼ばれるiOS向けのフレームワークを用意しています。特筆すべきは、scikit-learnやXGBoost、Kerasなど様々な機械学習フレームワークで学習したモデルをiOS向けに変換することができるようになったことです。^{†10}これにより学習したモデルをiOS用に変換し、高速に予測できるようになることが期待されます。

4.3 ログ設計

本節では、機械学習システムの教師データを取得するためのログの設計と、特微量について説明します。

機械学習、特に教師あり学習を行う場合、Webサーバのアプリケーションログや、どこをどうクリックしたかなどのユーザの行動ログなどを集めて、そこから特微量を抽出します。



機械学習の入力となる教師データはシステムのログから作成するのが一般的です。

ログは、DBなどのデータと異なりスキーマがない、記録していないデータを後から改めて取得するのが困難、といった特徴があり、システムに組み込むにあたっては様々なコツがあります。ログ設計は特微量を決めるための重要なポイントです。例えば、複数の自社で展開するWebサービスごとにユーザIDが異なる場合、CookieなどにUUIDを仕込んでIDの名寄せを試みるなどをする必要があります。しかし、UUIDを記録していなければユーザIDもマッピングできないため、複数のサービスをまたいだ特微量を得られません。特微量は、Feature Engineeringという言葉があるように試行錯誤が必要なものですが、ログにない情報を作る工夫をするよりはログをあらかじめ仕込めるのであれば、そちらの方が簡単です。考える必要なログを取得するためにも、どういった情報が必要かを考えましょう。

本節では、どこにあるどういった情報を利用し、教師データに活用するかについての概要を説明します。なお、具体的な教師データの詳細な収集方法は5章にて説明します。

4.3.1 特微量や教師データに使う情報

特微量や教師データに使えそうな情報としては、大きく以下の3つがあります。

1. ユーザー情報

^{†10} https://developer.apple.com/documentation/coreml/converting_trained_models_to_core_ml

2. コンテンツ情報
3. ユーザー行動ログ

ユーザー情報は、ユーザーに登録してもらう時に設定してもらう、例えば性別のようなユーザーの属性情報のことです。コンテンツ情報は、ブログサービスにおけるブログ記事や商品などのコンテンツ自身の情報です。これらは一般的には、MySQLを中心としたOLTP (Online Transaction Processing) 向けのRDBMSに保持されています。ユーザー行動ログは、ユーザーがどのページにアクセスしたか(アクセスログ) やユーザーが商品の購入などイベントを起こしたことのログです。特にユーザー行動ログは、広告のクリックイベントや商品の購買などコンバージョンに繋がる情報を持つことが多く、教師データになりやすいので、適切に収集できるようにしましょう。ユーザー行動ログはデータ量が多くなるので、オブジェクトストレージや分散RDBMS、Hadoop上のストレージなどに保存することが多いです。

4.3.2 ログを保持する場所

ユーザー行動ログはデータ量が多くなるので、保存場所には気をつける必要があります。MySQLやPostgreSQLなど業務用のRDBMSに格納すると、後々全データの傾向を見たりして当たりをつけることが難しくなります。こうしたデータは、機械学習用途だけでなく、レポートやダッシュボードなど、集計処理を経て可視化されることも多くあります。機械学習の前に対話的な分析をすることを踏まえると、以下のようなデータの保持方法が考えられます。

- 分散RDBMSに格納する
- 分散処理基盤HadoopクラスターのHDFSに格納する
- オブジェクトストレージに格納する

これらの保持方法に共通しておすすめなのは、SQLでデータにアクセスできるようにすることです。SQLでデータにアクセスできるようにしておくと、他のプログラミング言語を書かなくても様々な分析ができます。データの中の必要な情報を選別した上で転送をするといった操作が容易になるので、データの転送コストが下がります。近年では、AmazonのAmazon RedshiftやGoogleのGoogle BigQueryなどのフルマネージドなクラウド型の分散DBサービスが展開されており、いわゆるデータウェアハウスを手軽に用意できるようになりました。

あるいはApache Hadoopを用いた分散ファイルシステムHDFS (Hadoop Distributed File System)に格納するのも良いでしょう。Apache Hive、Apache Impala(Incubating)、PrestoなどHadoop上で動くSQLクエリエンジンを用いることで、SQLを使ったデータアクセスが容易になります。

2つ目と似ていますが、クラウドストレージに直接格納するのも選択肢の1つです。その場合は、Amazon Elastic MapReduce (EMR) やGoogle Cloud Dataproc、Azure HDInsightなどマネージドな

分散処理サービスを使うことで、SQLやMapReduceだけでなくApache Sparkを使った複雑な処理も可能です。特に近年では、Amazon S3のようなオブジェクトストレージにデータを格納して、そこに対してImpalaやHive、PrestoやAWS Athenaでクエリを直接実行するといったスタイルも増えてきました。

これらの生のデータからSQLを使った集計処理等をした後、機械学習用のデータセットとして利用します。

既にWebアプリケーションを運用している場合、クラウドストレージや分散データベースにアクセスログなどのログデータを保管していると思います。下記のようなマネージドのクラウドサービスを利用することで、管理コストが低減されるでしょう。

- クラウドストレージ
 - Amazon S3
 - Google Cloud Storage
 - Microsoft Azure BLOB Storage
- マネージド分散DB
 - Amazon Redshift
 - Google BigQuery
 - Treasure Data

こうしたログは、FluentdやApache Flume、Logstashといったログ収集ソフトウェアをWebアプリケーションサーバーに入れて、保管先に転送します。また、最近ではEmbulkのようなバッチでデータを転送するソフトウェアや、分散メッセージングシステムApache Kafkaを活用してスケーラブルなログ収集基盤を作る、というように選択肢が増えています。

4.3.3 ログを設計する上の注意点

機械学習を含んだシステムの開発を進めるときに、特徴量を抽出するには試行錯誤を繰り返すことがほとんどで、最初から有効な特徴量を見つけるのは困難です。つまり、必要そうなユーザー情報、コンテンツ情報についてはサービス設計時にあらかじめ想定しておく必要があります。

KPIの設計時にはできるだけ少ない指標にする方が良いのですが、機械学習に使える情報は出来る限り多い方が望ましいのです。後から必要に応じて特徴選択のロジックを加えたり次元圧縮をしたりすることは可能ですが、保存していない情報を増やすのは困難です。例えばあるユーザーが広告をクリックするかを予測する際に、性別や午前中／夕方に訪問した、掲出する広告のカテゴリなど、予測に関する情報の多様性を確保する必要があります。

また、現在取得しているログで教師データを作れるか、という視点も必要です。実際にあった例としては「広告をクリックしたログ」は保存していたが「広告を表示したログ」はデータ量が多すぎるため

破棄していた、ということがありました。この場合「広告が表示されたがクリックされなかった」というログが存在しないことになるため、教師データがうまく作れず、クリック予測が行えませんでした。

このほかにも、マスターデータの変更履歴を保存していなかった、という事例もありました。商品の説明文と売れ行きの関係を調査してほしいという依頼があり、購買ログと商品マスターを受け取りました。しかし、商品の説明文の変更是、商品マスターを直接書き換えて運用しており、いつからいつまでどのような説明文で販売していたのか、という情報が欠落していました。そのため、十分な調査を行うことができませんでした。

このように、システム開発・運用をすると人と分析をする人が分かれていると、検証に必要なコンバージョンしなかったなどのネガティブなデータや、マスターデータの変更履歴など重要なデータを捨ててしまう事があるので注意が必要です。

もう1つ気をつけて欲しいのが、ログ形式の変化についてです。機械学習をするにはデータ量が多いほうが満足の行く性能に達する可能性が上がります。長い期間のデータを集める上で、サービスの機能追加や仕様変更により、ログの形式が変化し取得する情報が変わることがあります。しかし、入力に使う特徴量のセットを途中で変化させることはまずありません。従って、長い期間の古い情報量の少ない時の特徴量のセットを使うか、短い期間で新しい特徴量のセットを使うかどちらが良いか検討が必要です。

大規模データの転送コスト

大規模データの機械学習における最も大きなボトルネックは、データの転送時間です。筆者の経験では、1GBを超えたログの生データを一括でダウンロードしてオンメモリで処理をするのは、やめたほうが良いと思います。

scikit-learnを使った学習をする場合に、どうしても機械学習のバッチ処理を行うサーバーにデータを転送しなければならないのですが、その時間を抑えるためには、分散RDBMSを利用したデータウェアハウスにMySQLなどのOLTPサーバーのデータを同期し、出来る限り分散RDBMS上でSQLを使って前処理ができるようにするのが望ましいでしょう。

大規模なデータに対して、複雑な前処理を定期的に実行する必要がある場合は、例えばAmazon S3に置いたデータをAmazon Elastic Map Reduceで加工するなど、出来る限りローカルマシンにダウンロードしない工夫をすることが必要です。

4.4 この章のまとめ

本章では、機械学習を情報システムに組み込むための設計と、ログ設計について説明しました。

一括学習をして得られたモデルから、予測結果をどのように呼び出すかによって4つのパターンがあります。

- バッチ処理で学習+予測結果をWebアプリケーションで直接算出する（リアルタイム処理で予測）
- バッチ処理で学習+予測結果をDB経由で利用する（バッチ処理で予測）
- バッチ処理で学習+予測結果をAPI経由で利用する（リアルタイム処理で予測）
- リアルタイム処理で学習

ログ設計に合わせて特徴量や教師データをすることが重要になってきます。これらを考えるときは、できるだけ手戻りが少なくなるように設計しましょう。

5章

学習のためのリソースを収集しよう

ビジネスにおける知見を得るためにには教師なし学習を用いますが、分類や回帰などの教師あり学習や推薦システムなどの性能を上げるためにには、ラベル付きのデータやコーパス、辞書などより多くの良質なリソースが必要です。本章では、教師あり学習を行うために必要な、学習のためのリソースを収集する方法について解説します。

本番環境で使うための機械学習の教師データは、世の中で公開されている既存のデータセットでは自分たちと問題設定が異なるなどの理由で不十分なことがほとんどです。この章では、少し泥臭いですが重要な教師データの作りについて学びます。

5.1 学習のためのリソースの取得方法

教師あり学習に欠かせない教師データですが、そもそも教師データには何が含まれているのでしょうか。教師あり学習の教師データに含まれる情報には、大きく以下の2つがあります。

- 入力：アクセスログなどから抽出した特徴量
- 出力：分類ラベルや予測値

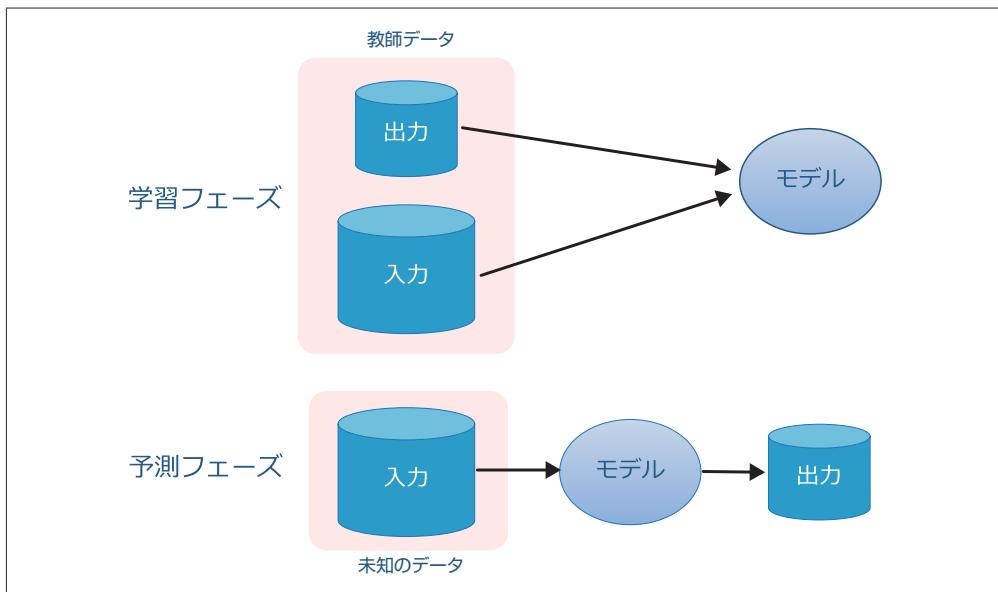


図5-1 機械学習(教師あり学習)の概要(再掲)

特徴量についての試行錯誤は前章に書きましたが、ヒューリスティックに判断をして追加する形になります。出力のラベルや値は以下のような方法で付与できます。

- サービスの中にログ取得の仕組みを用意してそこから抽出する(完全に自動)
- コンテンツなどを人が見て付与する(人力で行う)
- 機械的に情報を付与して、人手で確認する(自動+人力)

本章では、教師データを作るのは誰かという観点から説明を進めています。

1. 公開されたデータセットやモデルを活用する
2. 開発者自身が教師データを作る
3. 同僚や友人などにデータ入力してもらう
4. クラウドソーシングを活用する
5. サービスに組み込み、ユーザに入力してもらう

5.2 公開されたデータセットやモデルを活用する

既に世の中に存在する学習済みモデルや、コンペティション用に用意されたデータセットを使ってベースラインを学習し、それを転用していく手法です。本節ではこうしたリソースについての収集方法

を説明します。

有名なところでは、UCI Machine Learning Repository^{†1}や、機械学習コンペティションサイトのKaggle^{†2}の各種コンペティション及び一般の人が共有したデータセットがあります。画像認識の世界では一般物体認識用のImageNet^{†3}などタグ付きの画像を公開しているところもあります。また、直接の学習用データではありませんが、深層学習用のライブラリCaffeでは、学習済みのモデルを共有するModel Zoo^{†4}という取り組みが行われています。同様に、TensorFlowでは物体認識用のAPIを提供し、その中に一般物体認識用の学習モデルを含んでいます。^{†5}

日本語のテキストデータとしてよく使われるのが、Wikipediaのダンプデータや有償の新聞コーパスです。この方法の例としては、Wikipediaを形態素解析の辞書のリソースとして使ったハッカドール^{†6}があります。

しかし、本節で紹介した方法には、いくつか気にするべき問題点があります。

- モデルやデータセットは商用利用可能なライセンスか？
- 学習済みモデルやデータセットを自分のドメイン（自分たちが運用するシステム・サービス）に適用できるか？

1つ目について、特にラベル付きのデータは大学などが科研費を使って作成していることが多く、ライセンスが研究目的に限定されていることが良くあります。これらのリソースのライセンスは多くの場合、ソフトウェアのOSSライセンスのように定型化されていません。そのためWebに公開しているリソースであっても、問い合わせてみると商用利用できないといったケースもあります。モデルやデータセットが商用利用可能かどうかは必ず確認しましょう。また、リソースを利用して作成したモデルなどを再配布する際にも、元のリソースの制限を受ける場合があります。再配布をするしないにかかわらず、参照元が明確になるように管理するのが良いでしょう。

2つ目については、配布されているデータのドメインが実際に使うものと違う場合は、何かしら自分のドメインに適用するための工夫が必要となることもあります。詳細は触れませんが、興味がある読者は半教師あり学習(Semi-Supervised Learning)や転移学習(Transfer Learning)[transfer]について調べると良いでしょう。特に、画像の物体認識を行うタスクでは、転移学習を用い、既存の学習モデルに自分の解きたい画像の正解データセットを追加することで、少ない追加コストで目的の画像を認識するモデルを学習できることが知られています。この転移学習のアプローチで、ドラマ“Silicon

^{†1} <http://archive.ics.uci.edu/ml/>

^{†2} <https://www.kaggle.com/>

^{†3} <http://www.image-net.org/>

^{†4} <https://github.com/BVLC/caffe/wiki/Model-Zoo>

^{†5} <https://research.googleblog.com/2017/06/supercharge-your-computer-vision-models.html>

^{†6} http://www.slideshare.net/mosa_siru/ss-40136577

Valley”で使われたホットドッグ判定器も作られました^{†7}。

実際に既存のデータセットだけで解ける問題は限定的なので、続いて自分でデータセットを作る方法を考えてみましょう。

5.3 開発者自身が教師データを作る

既存のリソースをあてにできないときは、最初に開発者が自ら教師データを作ります。特にどのデータを特徴量にするかが性能に直結することも多いので、自分の手を動かして教師データを作るのはとても重要なことです。

はじめに、解きたい問題から分類をするべきなのか、回帰をするべきなのかを考えます。

例えば、ソーシャルブックマークサービスのカテゴリを予測する問題を考えてみましょう。^{†8}ソーシャルブックマークは、オンラインでお気に入りのWebサイトを共有できるサービスで、整理しやすくするためにカテゴリを自動付与しています。

まず「政治」「芸能」「テクノロジー」「生活」などのカテゴリと、その定義を決めます。この問題は、決まった数のカテゴリを予測する問題なので、分類問題ということになります。

試しに各カテゴリに所属するコンテンツを1000件程度ずつ収集して、人の手で分類します。「収集」と言ってもいろいろな方法があると思いますが、例えば既存のコンテンツがある場合、特定のキーワードが含まれるコンテンツを正解データとする方法があります。このように何かしらの基準でコンテンツを正解のカテゴリに分類していきます。これが最初の開発データ作りです。

なお、「1000件程度」と書いたのはあくまで目安です。実際にはもう少し少なくとも解ける問題もありますが、ひとまず1カテゴリでこれくらいのデータ量があれば最初のステップとしては十分でしょう。

機械学習で解ける問題は、およそ人が見て分かることがほとんどです。人がどんな情報を使ってカテゴリを分けているのかということを、教師データを作成しながら注意深く洞察します。

データを眺めていると、人が見ても曖昧なデータが存在することに気づくはずです。例えば「アイドルが大臣と選挙応援のイベントを行った」という内容のニュース記事があったとすると、これは「芸能」にも「政治」にも入りそうです。この時、この分類問題は排他的なカテゴリの分類をするのか、それとも1つのコンテンツが複数のカテゴリに存在するのか、といったことを考えるでしょう。それに応じて採用すべきアルゴリズムや予測の方法が変わってきます。データを見る前に決めたカテゴリはそのまで良いのか、分類の定義を更新した方が良いのかどうかも考えて分類を進めます。

あわせて、データを見ながら人手で分類をしていると、例えば「記事のタイトルに含まれる単語がカテゴリを分けるのに必要そうだぞ」というような重要な情報がわかってきます。「必要そう」と思った情

^{†7} <https://hackernoon.com/ef03260747f3>

^{†8} <https://ja.wikipedia.org/wiki/%E3%82%BD%E3%83%BC%E3%82%B7%E3%83%A3%E3%83%AB%E3%83%96%E3%83%83%E3%82%AF%E3%83%9E%E3%83%BC%E3%82%AF>

報を特微量に含めると、性能が改善することがあります。

こうしてプラスアップしながら、すべてのデータに対してカテゴリを付与する頃には、他の人にも説明できるような分類の定義ができあがっているはずです。出来上がった基準や分類に困ったコンテンツについては、判断基準と実例を残しておくと良いでしょう。ソースコードと同じで、一ヶ月後の自分は他人だと思って、言葉で説明できる基準を整理します。

この方法は、学習のためのデータセット作りの第一歩としては良いですが、カテゴリを付与したいコンテンツの量が増えたりすると、たちまち行き詰まるようになります。また一人の感覚で分類をしていくと、思い込みによる偏りが発生し、ユーザーの感覚とズレが生じてしまうこともあります。

では次に、これらの問題を解決するために、一人では取り組まない方法を考えてみましょう。

5.4 同僚や友人などにデータ入力してもらう

多くのデータが必要になった場合の解決法にはいくつかありますが、そのうち一番取り組みやすいのが同僚や友人などに協力を募る方法です。もちろん、外注に依頼する場合もありますが、おそらく予算を使わずにできる方法がまず最初に取り組みやすいでしょう。

もっともシンプルな方法として、スプレッドシートに判断対象のデータを列挙してラベルを付与してもらう方法があります。Google スプレッドシートなどブラウザで共有できるアプリケーションであれば、簡単に取り組めて作業の重複も起こりにくいで最初の一歩には向いているでしょう。

もちろん、可能であればラベル付与を支援するツールを作成し、作業してもらうのが理想です。作業の重複や作業者同士の干渉といった問題を気にする必要が少なくなります。そもそも、画像の領域選択などスプレッドシートだけでは完結できない問題の場合は、はじめからアノテーションツールを作るか、既にある専用のツールを利用するしかありません。

複数名に作業を依頼する場合は、事前にデータの内容を言葉で表現できるようにしておき、作業内容や判断基準についてきちんと説明しておくことが重要です。これは、一人で作業している時には自分の中に暗黙の基準があるはずですが、複数人で作業する際に自分の基準を人に期待しても、大抵の場合でその基準がぶれてしまうからです。特に分類問題は、できる限り対象をクリアに表現するデータを用意することが質の高いデータの獲得に重要であるため、暗黙の基準を文書化しておくことが望ましいでしょう。

複数名で作業をするときに、同一のデータに対して複数名に正解を付与してもらうことも重要です。正解ラベルの方向性が揃うように基準を用意しても、人によって判断がぶれてしまう課題もあります。例えば「人間の声色だけで喜怒哀楽を判断する」といったように、そもそも人間でも判断の難しい課題が存在するからです。複数名で作業をするときには、付与された正解データが作業者間でどれくらい一致しているかを把握することも重要です。単純に作業者間の一致率を見ることで、課題の難易度も把握できます。人間同士でも5割一致しないタスクは、そのデータを使って機械学習をしても解けない

可能性が非常に高いです。また、偶然に一致する可能性を考慮した基準である κ 係数（カッパ係数、Kappa Coefficient）を使って課題の難易度を判断することもあります。

複数人で作業をするときには、他の作業者の正解データを見せないようにする必要があります。作業者同士が互いの作成したデータを見てしまうと、それがバイアスとなって偏ったモデルが学習されてしまう恐れがあるからです。

5.5 クラウドソーシングを活用する

データの量を集めるために使う別の方法として、クラウドソーシング (Crowd Sourcing) を使う方法があります。日本でクラウドソーシングと聞くと、一昔前までランサーズ^{†9}やクラウドワークスなど^{†10}で中心だったコンペ型を中心とした作業を思い浮かべる方も多いかもしれません。クラウドソーシングにはコンペ型以外にも、Amazon Mechanical Turk^{†11}やYahoo!クラウドソーシング^{†12}、CROWD^{†13}などを中心に行われているマイクロタスク型と呼ばれる、データ入力など短時間でできる単純な作業を依頼する形態があります。マイクロタスク型の特徴は、多数の一般の人が集まり作業をしてくれるという点であり、ここがコンペ型との大きな違いです。

特に機械学習の教師データ作成と、マイクロタスクとしてのクラウドソーシングはとても相性が良く、クラウドソーシングと機械学習の研究も国内外に沢山あります。また、企業でクラウドソーシングを使って教師データを付与するケースも増えてきています。

クラウドソーシングを使ったデータ作成のメリットには以下のようなものがあります。

- 専門の作業者を雇うより作業がとても速く、金額も専門家への依頼より比較的安い
- 作業が速く終わるために、試行錯誤しやすい
- 作業コストが低いことから、複数人に同一タスクを依頼するなど冗長性を持ったデータ作成が可能

特に、少量の適切な難易度の課題であれば1、2時間程度で終わることも多く、より良いデータが獲得できる方法について試行錯誤を繰り返すことができるのが魅力です。

一方で、以下のような注意すべき点があります。

- 作業者が短時間で解けるようにする必要があり、タスクの設計が難しい
- 高い専門性が求められる作業は、手順の分割、詳細化などが必要
- 作業結果の質を担保するために、結果を利用する際の工夫が必要

^{†9} <http://www.lancers.jp/>

^{†10} <http://crowdworks.jp/>

^{†11} <https://www.mturk.com/mturk/welcome>

^{†12} <http://crowdsourcing.yahoo.co.jp/>

^{†13} <http://www.realworld.jp/crowd/>

特に結果の質を担保するために、その部分での試行錯誤やノウハウが必要になります。同じタスクを複数人に出して多数決を取るなど冗長性をもたせたタスク設計にしたり、事前に練習問題を解いてもらったりアンケートをするなどして作業者をスクリーニングするといった工夫をすれば、質を担保したまま大量のデータを得ることができるでしょう。

また、そもそもすべてのデータに対してチェックすることが不可能なので、データを得た後に質をどう評価するかについても予め考えておく必要があります。例えば分類用のデータの場合、「カテゴリごとにサンプリングをして適切なカテゴリが付与されていくかをチェックする」というように、少量のサンプルに対してチェックをする方法がよく取られます。

5.6 サービスに組み込み、ユーザに入力してもらう

教師データの収集を、必ずしも自分たちで行う必要はありません。正解データをサービスのユーザーに付与してもらうこともあります。この方法は、広い意味ではクラウドソーシングの1つと言えなくもないですが、自社サービスを展開している場合に、サービスをよく知っているユーザーに協力してもらえるのは大きな魅力です。

BtoCのサービスの場合に取り組みやすい方法としては、例えば直接的に簡単なアンケートを取る、コンテンツのタグをユーザーに付与してもらう、コンテンツのカテゴリを申請してもらう、検索結果や推薦結果の中の不適切なコンテンツを報告してもらうなど、データ収集の仕組みをサービスの中に埋め込んでしまう方法があります。ある程度の規模のユーザーがいることが前提になってしまいますが、その行為が何かしらのユーザーメリットがあるようにインセンティブを設計し、得られたデータを正解データとして利用します。この方法の例としては、例えば人間かどうかを判別するために画像中の文字を読ませるreCAPTCHA^{†14}があります。また、Amazonでは古くから検索結果のフィードバックをユーザーから報告してもらうフォームを設置し、積極的にユーザのフィードバックを活用しています。

このような仕組みがつくれると、新規コンテンツに対しても継続的に正解データを増やし続けることができる、変化に対して追従しやすくなるといった副次的なメリットもあります。

5.7 この章のまとめ

本章では、教師あり学習のための学習リソースを収集する方法について説明をしました。具体的には、公開されたデータセットやモデルを活用する、開発者自身が教師データを作る、同僚や友人などにデータを入力してもらう、クラウドソーシングを活用する、サービスに組み込みユーザに入力してもらうといった5つの方法について説明をしました。

†14 <https://ja.wikipedia.org/wiki/ReCAPTCHA>

十分な量の良質なデータの収集が、機械学習にとって重要なポイントになります。プロジェクトに適切な方法を採用しましょう。

6章 効果検証

「新機能をリリースしたら先週と比較して売上が20%上昇しました、プロジェクトは成功です」、果たして本当でしょうか?本章のテーマは効果の検証です。狙った効果は出たのか、その効果はどれほどなのか、効果検証はプロジェクトにおける仮説検証の総まとめです。本章では効果検証の基礎となる統計的検定と因果効果推論、検証の手法であるA/Bテストにフォーカスを当てます。

6.1 効果検証の概要

効果検証とはある施策によってもたらされた効果を推定することです。言い換えれば「Yという事象がXによってどれだけ影響を受けたか」を明らかにする行為にあたります。例えば広告配信サービスであれば、「広告表示1,000回あたりの収益額が新機能によってどれだけ増えたか」を検証します。検証結果によりリリースした機能を維持するか取り下げるかの判断、データを根拠にした意思決定が初めて可能になります。ここでの検証とは「何らかの計測値を利用して判断をすること」ですが、どの値もしくは指標を利用するかを決めるのは効果検証フェーズではありません。効果検証を成功させるには、手法の理解だけでなくプロジェクトの進め方も重要です。この章で紹介する手法は機械学習を利用した情報システムに限らず、Webサイトのデザイン選定や社会実験の評価といった幅広い分野で活用されています。

6.1.1 効果検証までの道程

手法の話に入る前に、いったん効果検証に至るまでの流れをおさらいします。

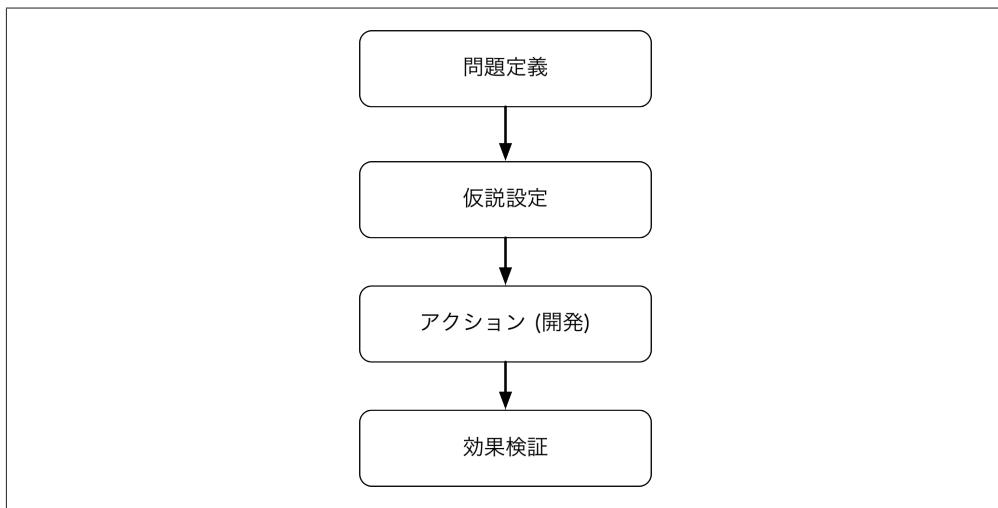


図6-1 効果検証までの流れ

何を目的として開発をするかは開発の前段階で決まるのは当たり前として、検証する指標も仮説設定の段階で決めるのがポイントです。ここで計測不可能な指標を設定してしまうと効果検証ができません。計測する仕組みがない場合は先にそちらを開発すべきです。具体例としてソーシャルネットワーカー サービスと広告配信システムの例を挙げます。

表6-1 ソーシャルネットワーキングサービス

フェーズ	内容
問題定義	ユーザーのアクティビティ率を上げたい
仮説設定	ユーザーにマッチするコンテンツを推薦して表示することで、平均滞在時間が伸びる
アクション	コンテンツのリコメンドシステムの開発
効果検証	平均滞在時間が伸びたか

表6-2 インターネット広告配信システム

フェーズ	内容
問題定義	利益率を上げたい
仮説設定	広告がクリックされるかどうかの予測精度を上げることで、広告オーディオの買いつけコストを下げられる
アクション	広告のクリック率予測器と、予測値を利用した入札ロジックの開発
効果検証	コンバージョン1件あたりのコストが下がったか

6.1.2 オフラインで検証しにくいポイント

機械学習モデルの性能評価については3章に解説がある通り、オフラインでも実施可能です。ではオフラインで検証が難しい効果には何があるでしょうか。

経済効果

事業の粗利や売上といった数値に責任を持つ人の興味は個々の予測器の性能よりも「利益をどれだけ押し上げたのか」という点に尽きます(図6-2)。特に事業部所属のデータ分析チームであれば利益貢献に関する説明は必須でしょう。仕入れ原価の最小化といったお金の動きに直結する問題を解いたのであれば予測は可能ですが、大抵は予測器の性能だけでなくサービスの利用者数やプロモーションといった変動要因に影響を受けます。C. A. Gomez-UribeらはNetflixのレコメンドシステムのビジネス価値として解約率の低下があり、10億ドル/年の効果があったとしています[Netflix_16]。これはオフライン検証でレコメンドの精度から導出するのが難しい値です。

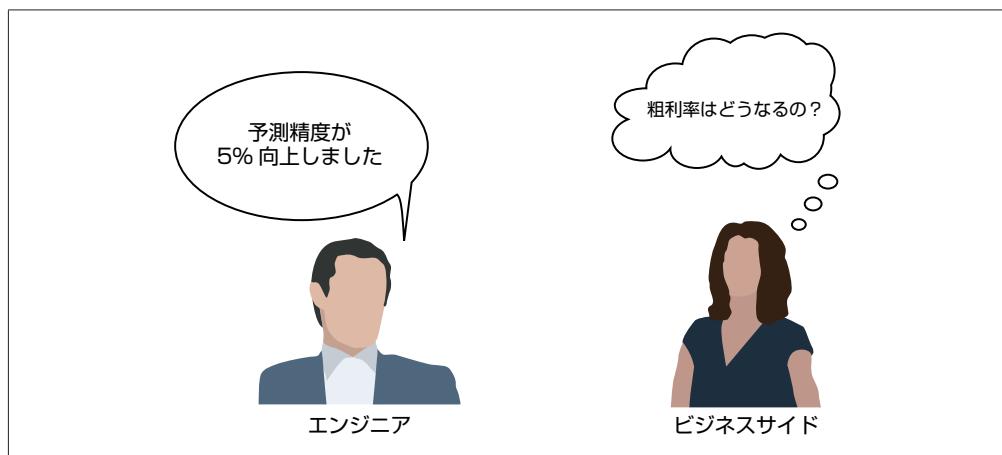


図6-2 知りたいのはそこじゃない

ログを経由した副作用

これは稼動中の機械学習モデルが他の学習データに影響を与えるケースです。例えばレコメンドシステムはユーザーの行動ログを学習データとしますが、レコメンドシステムによって提示された選択肢に対する行動ログはレコメンドによるバイアスがかかっています。自分自身の予測結果によるバイアスを含んだデータを使って実験するには実装コストの高いシミュレーターの実装が必要になります。また、同じログを学習データとしている他の予測器にも影響を与えます(図6-3)。

ログを通じて自分自身に限らない影響を残すことは、機械学習における技術的負債の1つとされます [Sculley_15]。

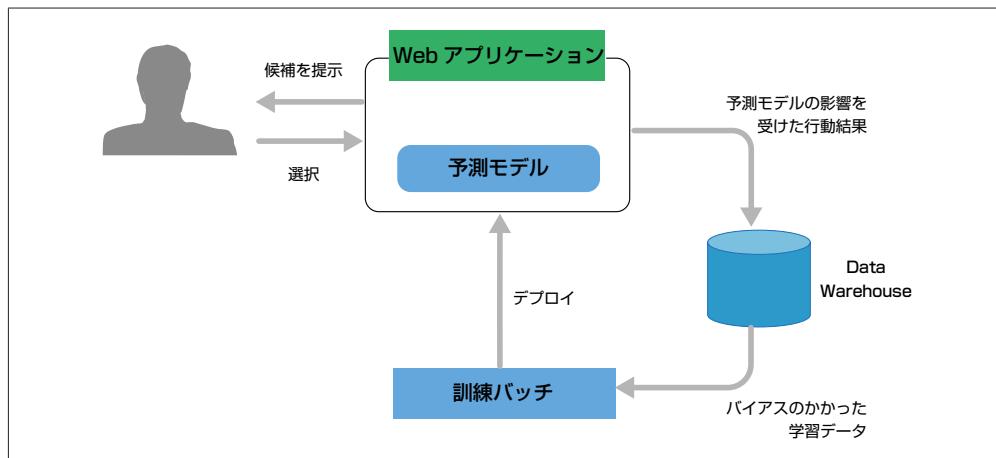


図6-3 ログの汚染

6.2 仮説検定の枠組み

仮説検定 (Hypothesis Testing) は効果検証のベースとなるもので、母集団に有意な差があることを標本 (サンプル) を使って確認する手法です。最初に基本的な例を2つ紹介します。なお本章では統計の基礎である大数の法則や中心極限定理の説明を省いています、詳しくは東京大学出版会『自然科学の統計学』[自然科学の統計学]などを参照していただければと思います。

6.2.1 コインは歪んでいるか

それでは実際の例を使って説明して行きましょう。コイントスゲームをするとして、利用するコインの直近20回の記録では表が15回、裏が5回出ているとします。直感的には表が出すぎているような気がしますが、どう判断したら良いでしょうか。細工や歪みのないコインであれば確率は半々と考えます。ここで50%で表の出るコインを20回トスした時の結果の分布を見てみましょう。これは二項分布 $\text{Bin}(20, 0.5)$ であることから次のコードで確率分布の形状が見られます。試行回数の丁度半分の10回表が出る確率が最も高いのが分かります (図6-4)。

```

import numpy as np
import matplotlib.pyplot as plt
import scipy.stats

x = np.arange(0, 21)
y = scipy.stats.binom.pmf(x, 20, 0.5)
plt.figure(figsize=(8, 2))
plt.bar(x, y)
plt.xlabel('表が出る回数')
plt.ylabel('確率')

```

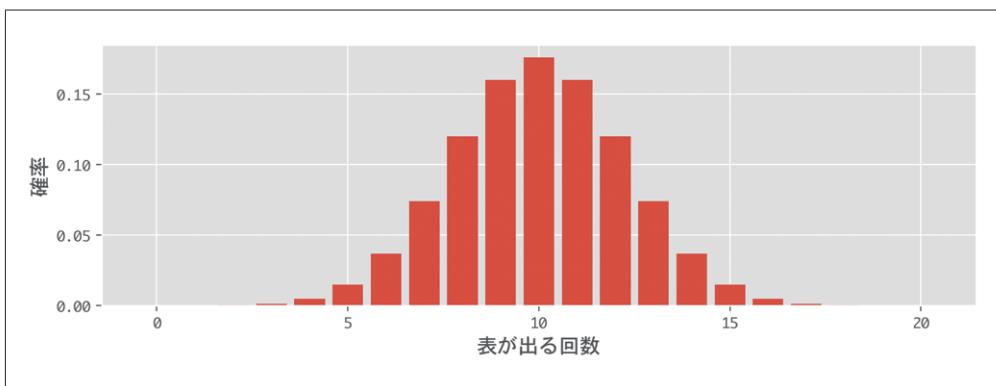


図6-4 表が出る確率が50%の時に20回のコイントスで表が出る回数の分布。

一方で分布の裾に着目すると15回以上表が出る確率 $p[\text{表が出る回数} \geq 15]$ は2%にすぎないことが分かります(図6-4)。

```
import pandas as pd
p_value = pd.DataFrame({'表の出る回数':x, '確率': y}).query(
    '表の出る回数 >= 15'
)[['確率']].sum()
print(p_value)

0.020694732666
```

コインに細工がないとしたら稀にしか起こらない結果であり、このことから「細工はある」と結論づけるのが検定の考え方です。

今回の例における検定の枠組みでは、表が出る確率は50%であるとした仮説を帰無仮説(Null Hypothesis)、表が出る確率は50%と異なるとする仮説を対立仮説(Alternative Hypothesis)、帰無仮説が真であるとした時の確率をp値(p-value)と呼びます。p値の閾値、例えば「5%より低いことが起きていたら帰無仮説を棄却すると」と判断する値を有意水準(Significant Level)と呼びます。標本は直近のトス、標本サイズは20、母集団は過去から未来まで全てのトスです。有意水準を0.05としたら、帰無仮説は棄却されます。0.01としたら棄却されません。



標本サイズはサンプルサイズもしくは標本の大きさとも呼びます。標本サイズと標本数(サンプル数)はよく混同されますが、上記コイン投げの例における標本数は1です。

6.2.2 二群の母比率の差の検定

あなたはECサイトを運営していて、集客のために広告を出稿するとします。2つの広告配信サービスに同時に出稿して1週間が経った後、それぞれの広告配信サービス経由で流入したユーザーの行動比較をしました。ユーザーが継続利用に至る確率が高い広告配信サービスの利用を継続し、低い方を止める意思決定をするためです^{†1}(図6-5)。データは表6-3の通りでした。

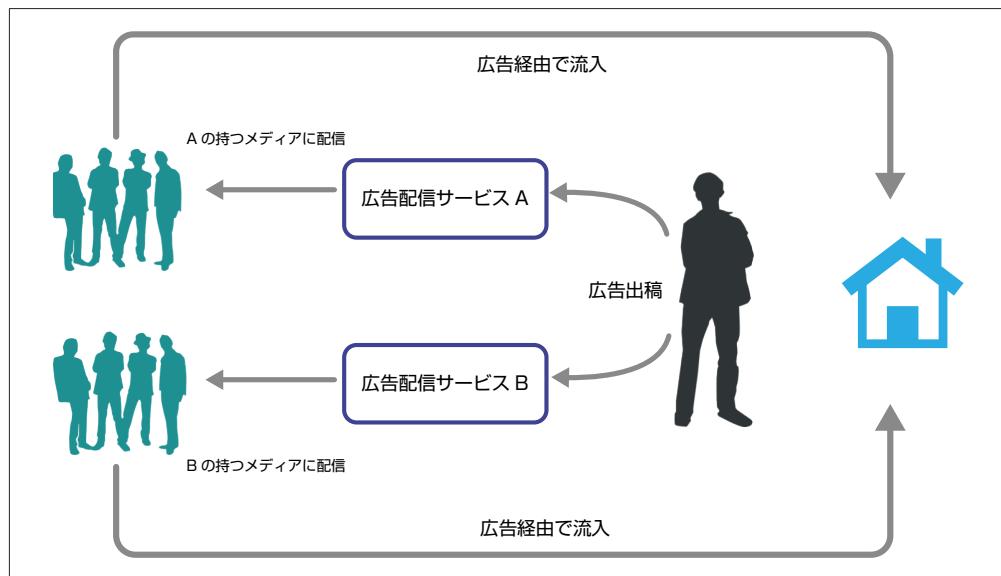


図6-5 獲得ユーザーの質に差がある？

表6-3 流入経路別ユーザーの継続化データ

流入元	流入人数	継続利用人数	継続化率
A	205	40	19.5%
B	290	62	21.4%

試しに継続化率の分布を可視化してみましょう。標本サイズがそれなりにあるので、二項分布の正規近似をします。

テストデータ。継続化人数, 離脱人数
 $a = [40, 165]$
 $b = [62, 228]$

```
print('Sample A: size={}, converted={}'.format(sum(a), a[0], a[0]/sum(a)))
print('Sample B: size={}, converted={}'.format(sum(b), b[0], b[0]/sum(b)))
```

^{†1} ユーザー獲得1人あたりのコストは同じとします。

```
Sample A: size=205, converted=40, mean=0.195
Sample B: size=290, converted=62, mean=0.214
```

```
x = np.linspace(0, 1, 200)

# 流入元がAの標本
n = sum(a)
p = a[0]/n
std = np.sqrt(p*(1-p)/n)
y_a = scipy.stats.norm.pdf(x, p, std)

# 流入元がBの標本
n = sum(b)
p = b[0]/n
std = np.sqrt(p*(1-p)/n)
y_b = scipy.stats.norm.pdf(x, p, std)

plt.figure(figsize=(7, 2))
plt.plot(x, y_a, label='Sample A')
plt.plot(x, y_b, label='Sample B')
plt.legend(loc='best')
plt.xlabel('新規ユーザーの継続化率')
plt.ylabel('尤度')
```

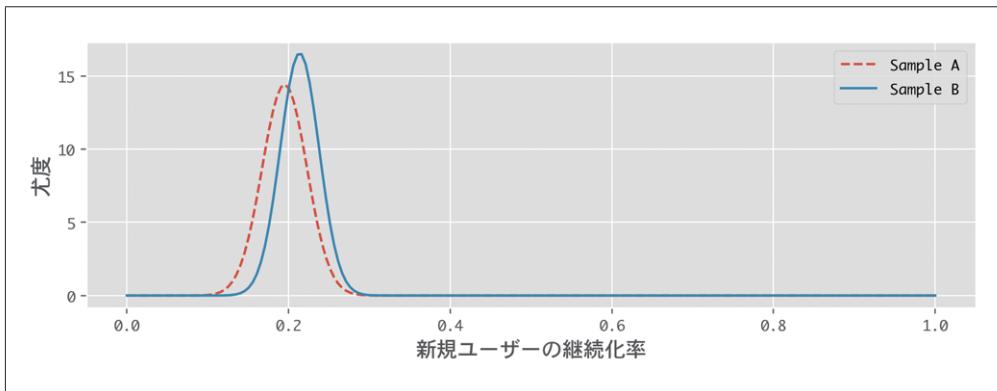


図6-6 流入経路別ユーザーの継続化率の推定値

Bの方が良さそうに見えますが、誤差なのか判断できません（図6-6）。

それでは仮説検定の枠組みを適用してみましょう。帰無仮説は「Aの母集団とBの母集団で継続化率は等しい」、対立仮説は「Aの母集団とBの母集団で継続化率に差がある」、有意水準は0.05とします。標本は今までの流入ユーザー、母集団は未来の流入も含めたユーザーの集合です。前述のコインの例とは異なり、カイ二乗検定で分割表の独立性検定を行います（表6-4）。

表6-4 検定の設定の分割表

流入元	継続利用人数	離脱人数
A	40	165
B	62	228

```
# カイ二乗検定
_, p_value, _, _ = scipy.stats.chi2_contingency([a, b])
print(p_value)

0.694254736449
```

p 値は 0.69 となり、帰無仮説が棄却できない事がわかりました。この時は帰無仮説が正しいとも間違っているとも言えません。信頼区間の幅がほぼ重複していることから、差があると言えないのは違和感の無い結果でしょう。



p 値という明確な判断基準が得られるのが仮説検定のメリットです。プログラムで数多くの検定を行ない機械的に有意な仮説のみを抽出する事も可能になります。



この例では2つの広告配信サービスの配信対象ユーザーが独立していると仮定しましたが、独立していない場合は互いに影響を及ぼしてしまうため注意が必要です。インターネット広告、特にターゲティング広告では異なる配信サービスが同一の広告取引場で配信権利の買い付けを行なうことがあります。この状況で配信サービス間のコスト比較を行なうと、配信権利のオークションで価格がつり上がるため、それぞれ単独で利用した時のコストがわからなくなります。

6.2.3 偽陽性と偽陰性

2つの例で、p 値が事前に定めた有意水準を下回った場合に帰無仮説を棄却するのが仮説検定の枠組みだと説明しました。しかし稀にしか起らないことが起こったから帰無仮説を棄却する判断をする以上、帰無仮説が真であったとしても有意水準の確率で誤って帰無仮説を棄却してしまいます。例えば有意水準5%で検定を行った場合、2群の間に差が無かったとしても5%の確率で、差があると判断することになります。この誤った発見を偽陽性 (False Positive) と呼びます。逆に本当は有意差があるのに帰無仮説を棄却しないケースを偽陰性 (False Negative) と呼びます^{†2}。病気の検査で陽性が出たものの、後の精密検査で異常なしと診断されることがあります、これが偽陽性です。検出すべき疾患を見逃した場合は偽陰性です。

^{†2} 偽陽性を第一種の過誤 (Type I Error) や α エラー (α Error)、偽陰性を第二種の過誤 (Type II Error) や β エラー (β Error) とも言います

関連する項目として分類器の性能評価について3章で解説していますが、仮説検定では検定力(Power)を検定結果の評価に利用します。検定力は

$1 - \text{有意差があるのに有意差がないと判断する確率}$

の値で、正しく有意差を検出できる能力を表します。

6.3 仮説検定の注意点

仮説検定は今も昔も幾度となく批判の対象となっています。2017年7月にp値の閾値を0.05から0.005にすべきだという声明も発表されました、これはp値を利用した研究の再現性の低さから来ています[Benjamin_17]。仮説検定の誤った使用は偽陽性の確率を上げ、p値のみによる意思決定は本質を見逃します。ここでは検定の誤った使用パターンとしてありがちなものを紹介します。

6.3.1 繰り返し検定をしてしまう

仮説検定で注意が必要な点は、検定対象の標本を固定することです。有意差が出なかったからといって標本を変えて再試験したのでは有意水準の意味が失くなってしまいます。前述のコイン投げの例で、表が出る確率が50%、つまり帰無仮説が真の状況で1回投げる毎に検定をしたらどうなるでしょうか。

```
mu = 0.5 # 表が出る確率50%
init_sample = list(scipy.stats.bernoulli.rvs(mu, size=20))

sample = init_sample
p_value_history = []
for i in range(200):
    # 直近20回の結果を使って検定
    _, p_value = scipy.stats.ttest_1samp(sample[-20:], 0.5)
    p_value_history.append(p_value)
    # 新たにコインを投げて結果を保持
    sample.append(scipy.stats.bernoulli.rvs(mu))

plt.figure(figsize=(10, 4))
plt.plot(p_value_history)
plt.xlabel('Test Epoch')
plt.ylabel('p値')
```

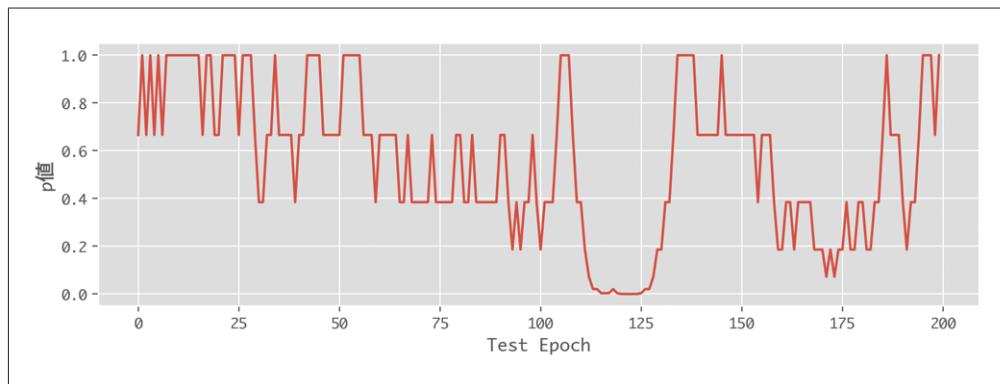


図6-7 コイン投げ1回毎に検定をした場合のp値の推移

120回目付近のp値が0.05を下回っています(図6-7)。表が出る確率は50%なので偶然起きたに過ぎません。検定を繰り返し行うと、いつか有意差が出てします。良い結果を出したいとの試行錯誤が偽陽性の確率を増やしてしまう点は検定を行う上で留意すべきでしょう。しかし、WebサービスでA/Bテストを実施している時の状況はこれに近く、p値を継続監視して有意になった所でテストを止めることができてしまいます。新たな観測データが逐次得られる状況における判断方法はA/Bテストの節で解説します。

6.3.2 有意差とビジネスインパクト

平均に差がある事の検定において、標本サイズによって結果がどう変わるか見てみましょう。平均に0.1%の差のある2群の比較をします。

```
max_sample = 3000000
# 標本A 平均:45.1%
a = scipy.stats.bernoulli.rvs(0.451, size=max_sample)
# 標本B 平均:45.2%
b = scipy.stats.bernoulli.rvs(0.452, size=max_sample)
p_values = []
# 5000づつ標本サイズを増やして検定を行う
sample_sizes = np.arange(1000, max_sample, 5000)
for sample_size in sample_sizes:
    _, p_value = scipy.stats.ttest_ind(a[:sample_size], b[:sample_size], equal_var=False)
    p_values.append(p_value)

plt.figure(figsize=(10, 3))
plt.plot(sample_sizes, p_values)
plt.xlabel('標本サイズ')
plt.ylabel('p値')
```

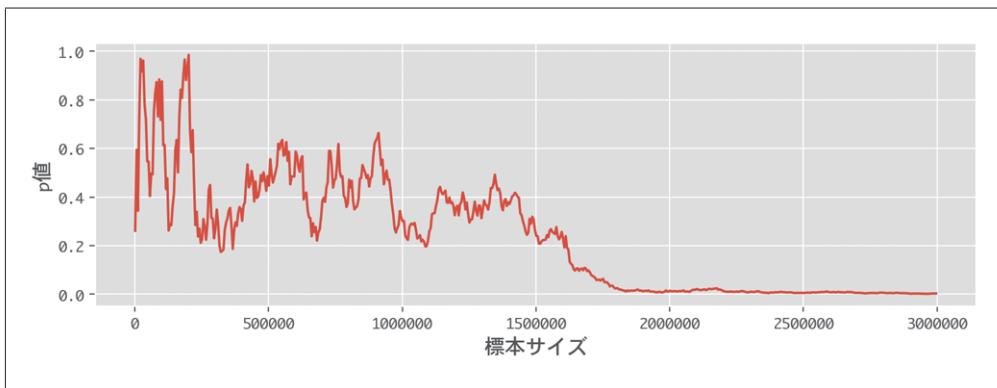


図6-8 標本サイズを増やしていく時のp値の変化

標本サイズが170万を越えたあたりでp値がゼロに近づき、有意差を示しています(図6-8)。これは推定量のばらつきを表わす標準誤差が、標本サイズの増大に従って小さくなるためです。標本サイズを増やしていくけば、わずかな差でも有意差となります。数百万の標本はWebサービスを運用していれば多くの数です。重要なのは有意差があるという事とビジネス上のインパクトは別である点です。0.1%の違いが大きな意味を持つのか、持たないのかはサービスの規模や事業の内容に依存します。リリースした機能を残すか取り下げるかの判断は有意差があるという点に留まらず、章の後半で紹介する効果量にも着目しましょう。



簡単に母平均の違いを見るのには信頼区間をプロットする方法がおすすめです^{†3}。

```
from statsmodels.stats.proportion import proportion_confint

# Wilson Score Intervalを利用して95%信頼区間を求める
a_lower, a_upper = proportion_confint(sum(a), len(a), alpha=0.05, method='wilson')
b_lower, b_upper = proportion_confint(sum(b), len(b), alpha=0.05, method='wilson')

plt.plot(1, np.mean(a), 'ro')
plt.plot(2, np.mean(b), 'bo')
plt.plot([1, 1], [a_lower, a_upper], 'r-')
plt.plot([2, 2], [b_lower, b_upper], 'b-')
plt.ylim(0.448, 0.454)
plt.xlim(0, 3)
```

^{†3} ベルヌーイ分布の信頼区間を求める方法はいくつかありますがサンプルコードではWilson Score Intervalを利用しています。平均がゼロ付近の場合でも正確な値が求まる点が特徴です。

```
plt.xticks([1, 2], ['A','B'], fontsize=20)
plt.xlabel('標本')
plt.ylabel('母平均')
```

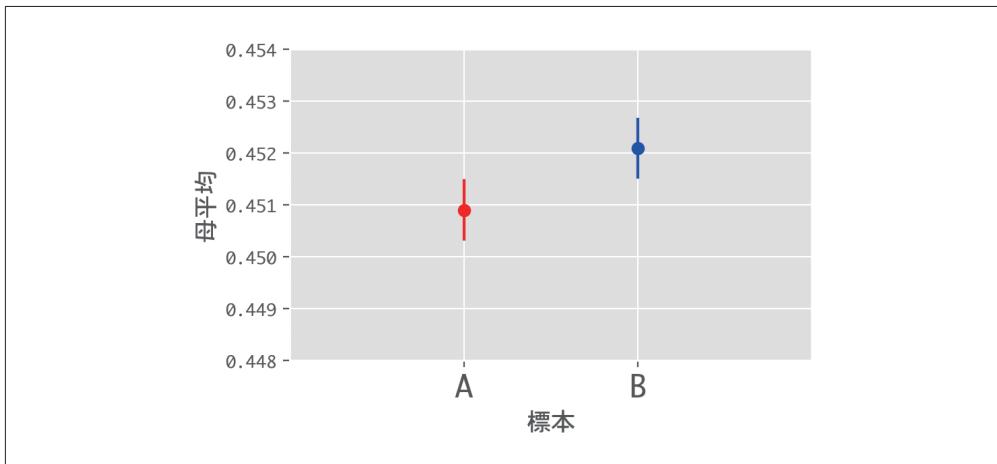


図6-9 95%信頼区間のプロット。信頼区間が分離しているため違いがある事は分かるが、差は0.001程度である。

6.3.3 複数の検定を同時に行う

くりかえし検定してしまうパターンに近いのが複数の仮説を検定するケースで、**多重検定 (Multiple Testing)** と呼びます。例えば M 個の説明変数の候補 $X \in X_0, X_1 \dots X_M$ から目的変数に有意に相関のある物を抽出したいとなると、 M 回の独立な検定が必要になります。 $(X_i$ が目的変数と独立であるという帰無仮説を棄却できれば有意に相関があるとできる) M 個の仮説を有意水準 α で検定をすると、誤って一つでも帰無仮説を棄却する確率は $1 - (1 - \alpha)^M$ です。 $\alpha = 0.05$ の時に M を増やしていく時のプロットを見ると $M=80$ ではほぼ 100% になることから多重検定は偽陽性を著しく上昇させるのが分かります。(図6-10)。このことから、多重検定において偽陽性を抑制する様々な方法が存在します。

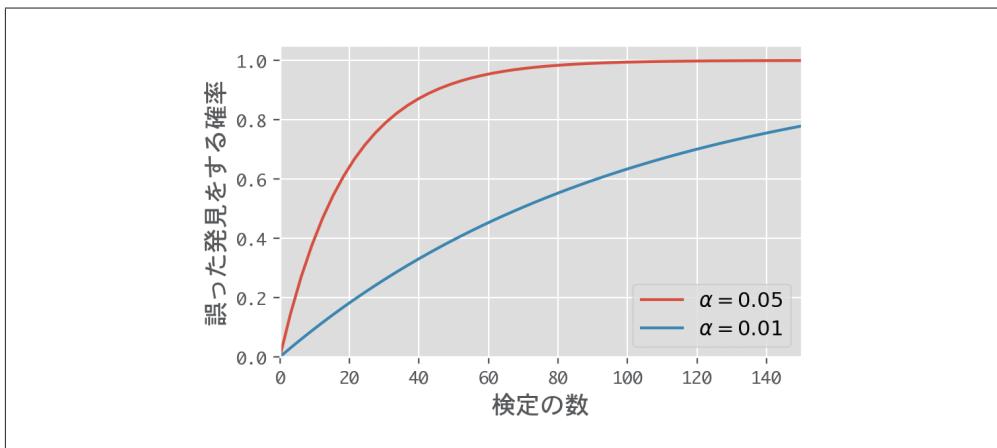


図6.10 誤った発見をしてしまう確率

多重検定において偽陽性を抑制するアプローチには、1つでも誤った発見をする確率 (Family Wise Error Rate、FWER) を抑制するものと、誤った発見の割合 (False Discovery Rate、FDR) を抑制するものの2つが存在します。前者の最もナイーブな方法はBonferroni法で、有意水準を α から α/M に変更します。ただしFWERの制御は検定力が著しく落ちます、検定力が欲しいケースはFDRを制御する手法が実用的でしょう。多重検定については文献[瀬々 15]が詳しいので、さらに知りたい方は参考にしてください。



「6.2.2 二群の母比率の差の検定」の例題の関連で、統計学の教科書に登場する「2群の平均差の検定は、等分散検定をした後に等分散であればStudentのt検定、不等分散であればWelchのt検定を行う」に対して、多重検定にあたるため最初から不等分散を前提とする検定を行うべきという主張があります。

6.4 因果効果の推定

仮説検定は標本から母集団の性質を推定する手法でした。次は母集団に対する効果を推定します。冒頭の「Yという事象が施策Xによってどれだけ影響を受けたか」を明らかにするために、因果推論における因果効果の考え方を見ていきます。

6.4.1 ルービンの因果モデル

インターネット広告の効果について考えてみましょう。因果推論では広告を見せるなどを介入 (Cause)、購買行動を結果変数 (Outcome)、介入した標本を介入群もしくは実験群 (Treatment Group)、介入していない標本を対象群もしくは統制群 (Control Group) と呼びます。

広告の効果は広告を見た時と見なかつた時の購買行動の差とすることができます。しかし個人の単位では観測可能な結果変数は介入を行つた場合か否かのどちらかに限定されます。広告に接触したAさんが広告に接触しなかつたケースは反事実 (Counter Factual) となり観測できません (図6-11)。

ここでルービンの因果モデルでは観測できないが潜在的に存在し得る結果変数を考えて、これを潜在的結果変数と呼びます。

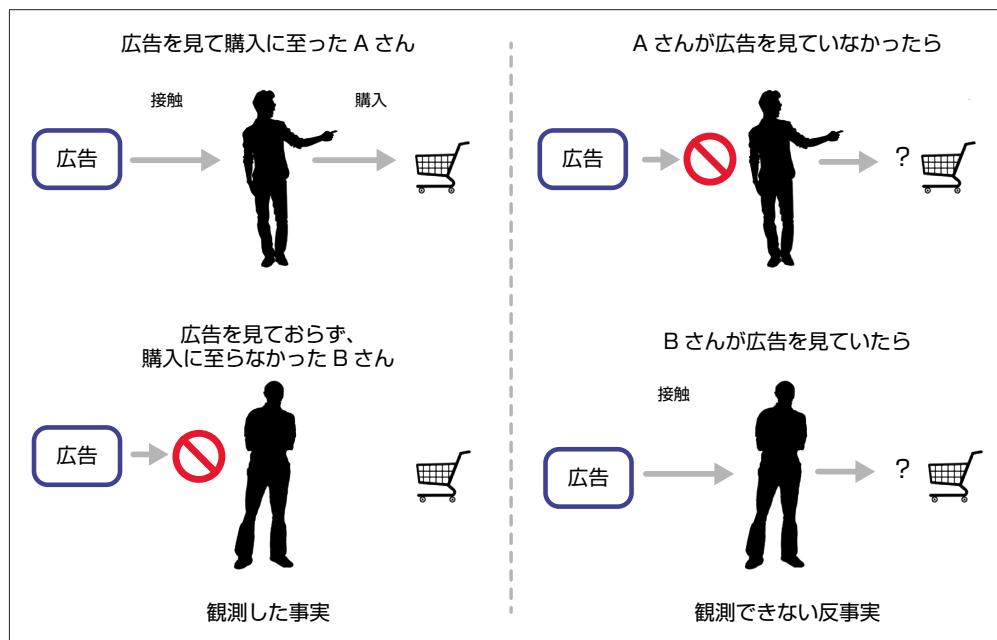


図6-11 個人の単位では片方のケースしか観測できない

購買に至つたかどうかの結果変数を $Y \in \{0, 1\}$ 、介入時と介入していない時の結果をそれぞれ Y_1, Y_0 として観測結果を表にすると次の通りになります。

表6-5 観測結果、ハイフンは欠損値を表わす

ユーザー	介入有無	Y_0	Y_1
1	1	-	1
2	1	-	0
3	1	-	1
4	1	-	0
5	1	-	1
6	0	1	-
7	0	0	-
8	0	0	-
...			
n	0	0	-

個人単位では $Y_1 - Y_0$ を測ることはできません。しかし我々が知りたいのは標本への効果ではなく、母集団への効果です。母集団への効果は個人単位の購買結果の差の期待値 $E(Y_1 - Y_0)$ と考えられます。これを平均処置効果 (Average Treatment Effect、ATE) と呼びます。

$$ATE = E(Y_1 - Y_0) = E(Y_1) - E(Y_0) \quad (\text{式}2)$$

以降はこの平均処置効果をいかに求めるかという話になります。

6.4.2 セレクションバイアス

求めたい平均処置効果の計算について、なんとなく「介入群の結果変数の平均」と「統制群の結果変数の平均」の差を取れば良いような気がします。観測できた値のみを使うので簡単です。しかしこの差は

$$E(Y_1 | \text{介入あり}) - E(Y_0 | \text{介入なし}) \quad (\text{式}3)$$

であり、介入有無と結果変数に相関があると式2と一致しません。インターネット広告の例に戻ると、購買行動に繋がりそうなユーザーを狙って介入をするのが一般的です。このため通常の観測結果では介入群は元々購買意欲が高い集団となり、介入群と統制群に差があることになります。この現象はセレクションバイアスと呼び、インターネット広告配信に限らず様々なデータに現われます^{†4}。

6.4.3 ランダム化比較試験

式2と式3が一致しない話をしましたが、一致するケースもあります。それは介入群と統制群に差がない時です。この状態を作り出して比較する手法がランダム化比較試験 (Randomized Controlled Trial、RCT) です。

標本に対してランダムに介入有無を決定することで、性質の等しい2群の片方に介入した状態を作ります。社会実験や臨床試験ではRTCはコストの高い手法とされます^{†5}が、Webサービスの効果検証には適用しやすいためA/Bテストのベースになっています。RCTを利用した広告効果計測サービスとしてGoogleのブランド効果測定^{†6}、Facebook ブランドリフト調査^{†7}が挙げられます。これは介入群にのみ広告を表示後、介入群と統制群にアンケートを実施する物です。PV数やクリック数といった指標と比較すると計測が難しいブランド認知ですが、RCTを使えば上手く捉えられます。

本章では因果推論について深掘りしませんが、興味のある方は文献[岩波データサイエンス Vol3]等

^{†4} がん検診の受診者と非受診者で発病リスクが異なるような、標本自身の持つ意思によって生じるバイアスはセルフセレクションバイアスと言います。

^{†5} 例えば救急搬送された患者に対してランダムに治療するしないを決めるのは倫理面で問題があるとされます

^{†6} <https://www.google.co.jp/ads/experts/blog/brand-lift.html>

^{†7} <https://www.facebook.com/business/help/1693381447650068>

に進んでもらえればと思います。RCTが実施不可能なケースにおける因果効果の推定など、実務で役に立つトピックが見つかるはずです。

6.4.4 過去との比較は難しい

Webサービスの売上向上施策Xをリリースした時のサービスのユーザー1人あたりの売上の変化を考えます。

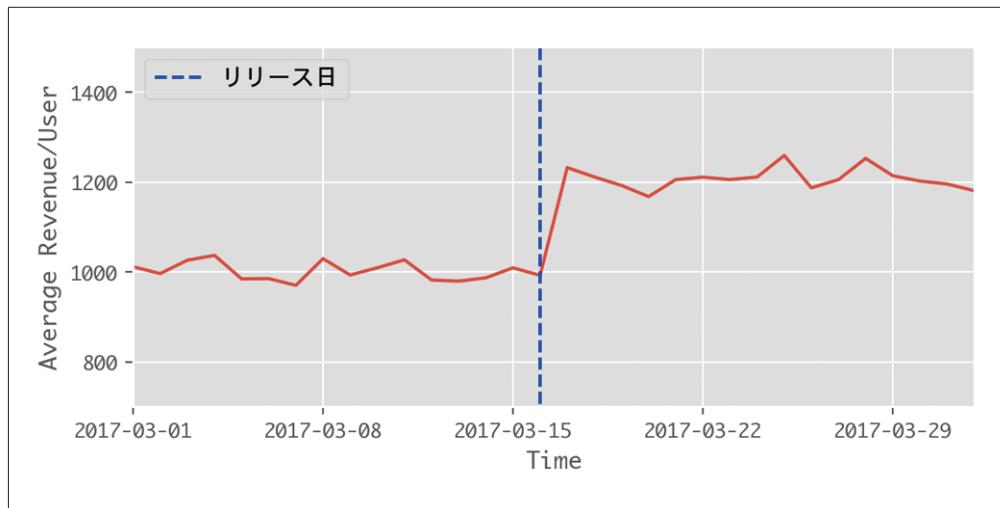


図6-12 売上向上施策は成功？

開発者としてはリリースしたタイミングで非連続な変化を観測したいと願うでしょう（図6-12）。しかしリリースタイミングで非連続な変化を観測しても、それが施策による効果とは言えません。筆者はインターネット広告業界の人間ですが、売上/粗利の時系列は強い季節性を持っており広告主の予算消化時期にピークがあります。また広告主の広告予算や配信先メディアの状況に強く影響を受けるため、時間経過によって刻々と非連続な変化をします。

こういった状況においては、過去との比較による因果効果の推定は非常に困難です。時系列モデルを構築してトレンド・季節性を除去した上で比較することも考えられますが、RCTで同じタイムスパンにおける2群を比較する方が簡潔で、施策による介入効果以外の要因を揃えることが可能（図6-13）。施策が無かった時の世界と比較して差が出ていれば施策の効果があったと判断できます。

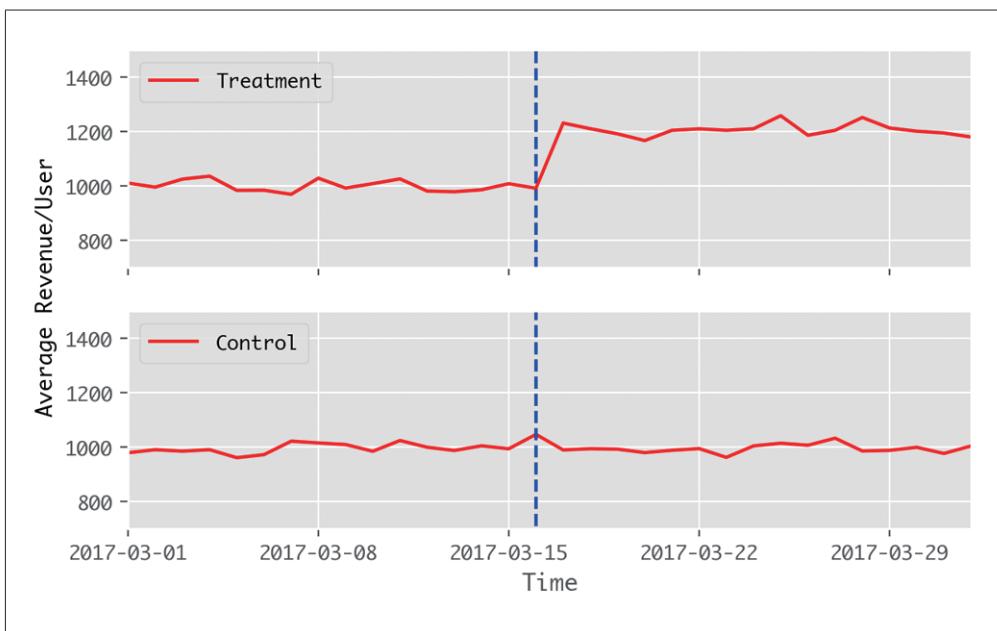


図6-13 施策が無かった時の世界との比較

章の冒頭の例「新機能をリリースしたら先週と比較して売上が20%上昇した」は、売上が20%上がった現象と施策の因果関係が不明なので施策の効果は何とも言えません。9章では発展的な手法であるUplift Modelingの紹介をします。

6.5 A/B テスト

前節でRCTによるセレクションバイアスの除去と、2群の同時比較による時間変化影響の除去について説明しました。A/Bテストは本番環境で前述2つの条件のもとで行うテストで、Webサービスの施策効果検証に広く使われています。A/Bテストの流れは次の通りです。

1. 2群の抽出
2. A/A テスト
3. 片方に介入
4. 結果の確認
5. テスト終了

いくつかのポイントを解説します。

6.5.1 2群の抽出と標本サイズ

仮説検定の節では標本サイズとp値の関係について説明しました。小さな差を検出したければ多くの標本が必要になります。適切な標本サイズの決定は文献【[自然科学の統計学](#)】等を参照していただくとして、A/Bテストでは事前に標本サイズを決定しなければならないパターンがあります。例を2つ示します。

表6-6 標本サイズの違い

ケース	検証方法	標本サイズ
A	ユーザーを2群に割り振り、介入後に1人あたりの売上の平均に違いが出るか検証する	2群を作った時点で固定
B	ユーザー新規登録画面の新デザインが既存の物と比較して登録完了率に違いがあるか検証する。新規ユーザーが訪れる時に一定の確率で新デザインを表示する	新規ユーザーの訪問がある限り増え続ける

ケースAで標本サイズが足りない場合、テストのやり直しになるため標本サイズの見積りは必須です。

一方で標本サイズが大きいとテストの影響を受けるユーザーの数が増えます。テスト施策の効果がマイナスになることも考えられるため、多すぎるのも問題です。別の問題として全ユーザーを半々にしてテストをすると、他のテストを同時に実施できなくなってしまいます。開発効率の観点からも部分抽出が良いでしょう。また、介入群・統制群の使い回しをすると前のテストの影響を引き継いでしまうため、テスト実施ごとに抽出を行うべきです。

6.5.2 A/Aテストによる均質さの確認

セレクションバイアスを避けるために介入するかどうかをランダム抽出で決めるのがRCTでした。Webサービスでユーザー単位に効果を与える施策であればランダムにユーザーを抽出して介入有無を決めます。ランダム抽出により等質な2群が得られるはずですが、それを確認するのがA/Aテストです。2群を抽出後に時間において2群に差がなければ片方に介入します。もしくは過去のデータを使って差がないことを確認できれば、即座にテストを開始できます。

6.5.3 A/Bテストの仕組み作り

A/Bテストを実施するにはサービスがA/Bテストに対応している必要があります。ランダム抽出や介入有無の設定はどんな施策にも必要となります。機械学習を使った施策特有の注意点としては学習データの分離が挙げられます。「ログを経由した副作用」で予測器が学習データに与える影響について説明ましたが、複数の予測器をA/Bテストで比較している時に学習データ経由で互いの影響を受けると本来の動作になりません。予測器が学習データに影響を与える場合においては学習データの分離

が必要になります。

他にも Microsoft の A/B テストチームは次の機能を活用しているとのことです [Microsoft_17]。

- 効果の悪いテストを早期に止めるためのアラート・自動停止
- テスト同士で相互作用がある物を自動で検出

6.5.4 テストの終了

テスト結果の判断を早く行うことには価値があります、良い施策をより早く全体に適用できます。また、効果の悪い施策は早くテストを止めるべきです。A/B テストで起きるのが、終了時期を決められずテストを放置してしまうことです。テスト実施期間にリミットを設定するのは良いアイデアでしょう。新たな観測データが逐次得られるテストの終了タイミングは自明ではありませんが、母平均が異なる事のテストであれば平均値の時系列プロットに信頼区間を重ねる事で判断ができます。信頼区間が分離したら差があるとし、なかなか信頼区間が分離しない場合はいずれ有意差が出たとしても差は小さく効果は見込めないと判断して中止できます。

A/B テストプラットフォームを運営する Optimizely は、p 値の継続監視による早期判定手法を提案しています [David_17]。

6.6 この章のまとめ

母集団に差がある事の確認方法、施策と結果の因果関係の捉え方、A/B テストについて解説しました。工場の抜き取り検査と違い、Web サービスの場合は低成本で多くの標本が利用できます。そのため限られた標本を使って母集団の性質を予測する統計のメリットはない様にも感じますが、副作用を共なう A/B テストにおいてはその影響を最小化できるのです。

第II部

7章から9章は、実際に手を動かして学べるケーススタディを中心にしています。これまでの章で説明した内容を含め、知識を読者が実際に直面する問題へ役立てるためのヒントとなるような内容を目指しています。

それぞれの章では、第I部の中で紹介した内容を含んでいますので、これまでの内容を思い出しながら読み進めると良いでしょう。

7章

映画の推薦システムをつくる

本章では、映画のデータを用いてレコメンデーションを行うことで、推薦とはなにか、推薦システムはどのように設計すればいいのかを学びます。

7.1 シナリオ

映画の推薦システムを作るために、MovieLens^{†1}という映画のレーティングサイトのデータを用いて、ユーザーが評価した星の数を予測します。

映画の推薦というテーマに関しては2006年に開催されたNetflix Prizeというコンテストがとても有名です。映画のレーティングに関する巨大なデータを用意し、当時使われていた推薦システムの精度を10%向上させれば賞金がもらえるという内容でした。MovieLensは、GroupLensというミネソタ大学の研究所が、Netflixによく似た情報をアカデミック用のデータとして公開しているものです。

今回は推薦のアルゴリズム、データの取得方法、評価尺度を学んだあとで、あるユーザーが見たことのない映画の評価値を予測してみましょう。システム構成としては、今回利用するアルゴリズムの制約から予測結果をDBに格納してWebアプリケーションから参照する形をとります。

7.1.1 推薦システムとは

推薦システムとは、何をするものかを考えてみましょう。AmazonなどのECサイトで表示される「この商品を買った人はこれも買っています」のような、関連する商品のオススメを思い浮かべる人もいるでしょう。また、5段階の星で示される評価値に基づいて楽曲がお薦めされる音楽アプリなどもあります。推薦システムは、何かしらのユーザー行動やアイテムの情報から、ユーザーが好むであろう関連アイテムを提示するのが目的です。ある一定の規模を超えた動画や音楽などのストリーミング配信サービスや大量の商品が存在するECサイトでは、ユーザーが大量の情報から検索だけで目的的コンテンツ

^{†1} <https://movielens.org/>

に到達するのは非常に難しく、推薦を使うことで自分の好みのコンテンツに到達しやすくなります。^{†2}特に自分が知らなかっただけで好みである意外性のあるコンテンツとの遭遇は、一意な答えが決まっていることが多い能動的な検索とは異なる大きなメリットです。一方で、サービス提供者にとっても良い推薦システムを提供することは、商品であれば購入のコンバージョン率を上げたり、動画や音楽のストリーミングサービスのデイリーアクティビティユーザーを増やします。

次節からは、文献 [kamishima] を参考にしながら推薦システムの特徴を整理していきます。この文献では、推薦システムについて学術的にどのような取り組みがなされてきたかを整理しています。

7.1.2 応用シーン

文献 [schafer] では、Eコマースにおける推薦システムの応用シーンは、運用目的に応じて以下の5つに整理されるとしています。これはサービスを作る上でも参考になるので紹介します。

- **概要推薦 (Broad Recommendation)**
- **利用者評価 (User Comments and Rating)**
- **通知サービス (Notification Service)**
- **関連アイテム推薦 (Item-associated Recommendation)**
- **パーソナライゼーション (Deep Personalization)**

「概要推薦」は、今週の人気商品のような統計情報ベースのおすすめや、編集者のセレクトしたアイテムを薦める場合など、人によらず大まかに行う推薦です。システムを利用し始めたユーザー や、たまたまにしか利用しないユーザーに特に効果を発揮します。

「利用者評価」は、ユーザーが☆を付けた評価やコメントを他のユーザーに見せたり、評価値の平均など統計情報として見せたりします。あまり推薦という感じはしませんが、他者の情報をもとに判断する基準を与えることができます。他のユーザーがおすすめしている情報は信頼できる根拠の1つと考えられるため、ユーザーは納得の行く選択ができるようになります。

「通知サービス」は、プッシュ通知やメールでユーザーが興味をもつアイテムを推薦することによりサイトへの再訪を促します。今となっては当たり前ですが、これも1つの応用シーンです。

「関連アイテム推薦」は、関連アイテムやその情報を元のアイテムと同時に提示することで、アイテムを同時に購入したり別のアイテムと比較したりすることを助けます。AmazonなどのECサイトで定番の方法ですね。

「パーソナライゼーション」では、人気のアイテムリストや編集者のおすすめリストと並べて、ユーザーが好みそうなアイテムリストを表示することで、そのユーザーが気に入ったアイテムに出会えるよ

^{†2} 動画ストリーミングサービスのNetflixは「Everything is a Recommendation」と言って推薦なしではサービスが成立し得ないと言っています。 <https://medium.com/netflix-techblog/55838468f429> を参照。

う促します。また、検索結果を個人によってカスタマイズするという方法もあるでしょう。

7.2 推荐システムをもっと知ろう

本節では、推薦システムに特有のデータの設計や取得方法、アルゴリズム、評価尺度について紹介します。

7.2.1 データの設計と取得

推薦システムの入力データとなりうる情報には、いくつかの種類があります。

- 嗜好データ (Preference Data)
- 検索クエリ (Query)
- 批評 (Critique)
- アイテム特徴 (Item Feature)
- デモグラフィック特徴 (Demographic Feature)
- コンテキスト特徴 (Context Feature)

嗜好データは、ユーザーのアイテムに対する「好き」という感情や5段階で「2」という評価など、その好みを表すものです。検索クエリは、例えばレストラン検索時に「5000円以下の和食の店」と指定する検索キーワードの情報で、批評は商品やお店に対する口コミなどを指します。アイテム特徴は商品説明文中の単語などの情報で、デモグラフィック特徴はユーザー自身の性別や年齢などの情報、コンテキスト情報は推薦されたアイテムを使った日付や位置の情報、アイテムの在庫状況など、推薦に関連する文脈の情報です。

推薦システムの難しさは、データが非常に疎 (Sparse) であるという点に原因の1つがあります。例えば、これまで見た映画は人によってまちまちです。そのため、人気の映画については評価情報が集まりますが、そうでない映画は情報が集まらず、一向に推薦されないということがよく起きます。また、ユーザが一生のうちに見られる映画に比べて、世の中に出て回っている映画の方がはるかに多いため、行レベルで見ると多くの箇所が欠損になります。そのため、評価される情報は一部のアイテムに偏り、評価値行列のはとんどの要素は評価情報のない行列になります。ここで評価値行列とは、例えば表7-1のようなユーザー × アイテムの評価の表のことを指します。

表7-1 映画の評価値行列の例

	映画A	映画B	映画C
ユーザー 1	5		2
ユーザー 2	4		1
ユーザー 3		4	5

また、推薦システムの良いところは、嗜好データなどの評価情報をそのまま推薦の正解として利用できることです。正解データを多く集めることは重要ですから、なるべく正しい情報を多く集める工夫が必要です。そのためには、ユーザーがアイテムを評価するコストを下げるか、ユーザーの評価回数が少なくても良いような何らかの工夫が必要です。例えば音楽の評価では、1曲を聞く事自体はおよそ数分程度で終わるため、1人の人が複数の曲の評価することはそこまで大きなコストになりません。このように、アイテムの評価に対するコストが低い場合は「好き」といった嗜好データが容易に手に入ります。一方で結婚式場や家の購入など、人生に何回もないようなイベントでは、アイテムの評価に対するコストが高く、嗜好データを大量には手入するのは難しいため、式場のWebページのPVのような別の指標を使うなど、少ない嗜好データや批評を補うための工夫が必要です。

この嗜好データの取得は推薦システムを構築する上で重要なことです。嗜好データの獲得についてもう少し詳しく見てみましょう。

7.2.2 明示的データと暗黙的データ

嗜好データを獲得する方法は大きく2つあります。1つは、ユーザーに直接好き嫌いや関心のあるなしを質問して回答してもらう明示的データ (Explicit Data) を集める方法です。もう1つは、利用者が商品を購入したり閲覧したりといったアイテムは関心があるとみなす暗黙的データ (Implicit Data) を集める方法です。表7-2にそれぞれの長所・短所を示します。

表7-2 嗜好データの獲得法による長所・短所

種類	明示的	暗黙的
データ量	x	○
データの正確さ	○	x
未評価と不支持の区別	○	x
利用者の認知	○	x

まず、データ量は明示的データよりも暗黙的データの方が圧倒的に多くなります。データ量が多いほうが統計的な方法を用いることができ、予測モデルを構築した場合にも予測性能の向上が期待できます。また、多くのユーザーはアンケートに答えるなどの嗜好データを提供するインセンティブがない場合も多く、明示的データの数を増やすのは難しいでしょう。

一方、データの正確さについてですが、明示的データはユーザーが明確に自分の意志を表明しているため、正確なことが多いです。暗黙的データは、例えばページの閲覧情報を暗黙的データとして利用している時に、誤クリックでページ遷移をしてすぐに戻った時にもポジティブな評価情報として取得されてしまいます。このように、暗黙的データの正確性は低い場合が多く、質を担保するために、例えばページの閲覧情報であれば滞在時間でフィルタリングするなど、データの前処理が重要になります。

また、未評価と不支持の区別ができるないのは暗黙的データの弱点です。暗黙的データの場合、「嫌い」に相当するネガティブな評価を取得できません。例えば、閲覧した映画はポジティブな評価と考えても

良いのに対し、閲覧しなかった映画は未定義なのかネガティブな評価なのか区別がつきません。場合によっては、見ていないけれど好みの映画を「嫌い」だとシステムが判定しやすくなってしまいます。

利用者の認知は、ユーザーに対して明示的にデータを獲得することで、どういった情報をシステムに与えたのかをユーザーが理解しているかどうかです。例えば、音楽アプリで最初に好きなアーティストやジャンルの情報を与えることで、「あのアーティストに関連する楽曲」というように、システムが推薦する情報に何らかの理由を考えられるようになります。システムが根拠のある出力を行っていると思ってもらえることから、ユーザーに好印象を与えやすくなります。

7.2.3 推荐システムのアルゴリズム

推荐システムのアルゴリズムは、映画の趣味が似ている人にオススメを聞くように、評価の傾向が似ている人を探したり、似たような評価をされる映画を探したりする**協調フィルタリング (Collaborative Filtering)**と、映画の監督名やジャンル、タイトル中の単語など内容が似た映画を探す**内容ベースフィルタリング (Content-based Filtering)**の大きく2つに分けられます。

協調フィルタリングの中でも特に、似た人を探すアプローチを**ユーザー間型協調フィルタリング (User-based Collaborative Filtering)**、似たアイテムを探すアプローチを**アイテム間型協調フィルタリング (Item-based Collaborative Filtering)**と呼びます。これらはシステムが持っているデータを元に、直接受けたユーザー/アイテムを提示するため、**メモリベース協調フィルタリング (Memory-based Collaborative Filtering)**の一種であるとも言われます。

また、メモリベースと違うアプローチとして、回帰や分類などの予測モデルを学習する**モデルベース協調フィルタリング (Model-based Collaborative Filtering)**もあります。

それでは、それぞれのアルゴリズムについて詳しく見てみましょう。

7.2.4 ユーザー間型協調フィルタリング

ユーザー間型協調フィルタリングは、「あなたと同じ商品を買った人はこんな商品を買っています」というような推薦をするための方法です。

協調フィルタリングは、ユーザー × アイテムの評価値行列があるときに、データに欠落のある箇所の評価値を予測します。ユーザー間型協調フィルタリングは、以下の流れで行います。

1. ユーザーの情報をベクトルで表現する
2. ユーザー間がどれくらい似ているか(類似度)を決める
3. 類似度にもとづいて評価値を算出する

例えば、ユーザーが映画を評価した評価ベクトルを考えます。

```
user[i] = [rating[i][1], rating[i][2], ..., rating[i][m]]
```

これは、表7-1において、あるユーザーの評価の行を1つ抜き出してきたことになります。変数 ratingはk人が、全m個の映画に対して付与した評価値のデータだとします。このとき、i番目のユーザーUの評価値ベクトルを $u = \text{user}[i]$ と、j番目のユーザーVの評価値ベクトルを $v = \text{user}[j]$ として類似度について考えてみましょう。

類似度は値が大きくなれば似ており、小さくなれば似ていません。代表的な類似度としては、ピアソンの相関係数 (Pearson Product-moment Correlation Coefficient)、コサイン類似度 (Cosine Similarity)、ジャッカード係数 (Jaccard Index, Jaccard Similarity Coefficient) が挙げられます。

ピアソンの相関係数はGroupLensの論文[grouplens]でも用いられており、いわゆる相関係数と呼ぶ時はこれを指すことが多いです。ピアソンの相関係数は-1から1の値を取ります。ピアソンの相関係数をコードで書くと、

```
import numpy as np
def pearson_coefficient(u, v):
    u_diff = u - np.mean(u)
    v_diff = v - np.mean(v)
    numerator = np.dot(u_diff, v_diff)
    denominator = np.sqrt(sum(u_diff **2)) * np.sqrt(sum(v_diff **2))
    return numerator / denominator
```

と表現できます。これは、SciPyを使うと以下のように算出できます。

```
from scipy.spatial.distance import correlation
1 - correlation(u, v)
```

コサイン類似度はテキストの文章間の距離を算出するときによく利用し、0から1の値を取ります。以下のように求められます。

```
np.dot(u, v) / (np.sqrt(sum(u **2)) * np.sqrt(sum(v **2)))
```

SciPyを用いるとこのように求められます。

```
from scipy.spatial.distance import cosine
1 - cosine(u, v)
```

ジャッカード係数は集合間の距離を計算することができ0から1までの値をとります。0、1の2値で表現するようなベクトルの距離を以下のように求められます。

```
np.dot(u, v) / (sum(np.absolute(u)) + sum(np.absolute(v)) - np.dot(u, v))
```

SciPyを用いるとこのように求められます。

```
from scipy.spatial.distance import jaccard
1 - jaccard(u, v)
```

どの類似度が良いかは問題によって変わってくるので、色々な方法をためしてみると良いでしょう。

ユーザー間型協調フィルタリングでは、類似するユーザーの好みのアイテムを提示するのが一番シンプルな使い方です。評価値を予測する場合、一番単純な方法に似ているユーザー上位k人の評価を平均する方法があります。あるユーザーUの映画Mに対する予測した評価値は以下のように求めることができます。

```
np.mean(nearest_user_ratings) / k
```

ここでnearest_user_ratingsは、似ているユーザー上位k人の映画Mに対する評価値のベクトルです。

また、ユーザー同士の類似度で重みをかけた以下のような予測値の求め方もあります。

```
np.dot(nearest_user_rating, nearest_user_similarity) / \
np.sum(nearest_user_ratings)
```

nearest_user_similarityは似ているユーザー上位k人のユーザーUとの間の類似度ベクトルです。よく似ている人の評価値には大きな重みをかけ、あまり似ていない人の評価値には小さな重みをかけて足し合わせることで、似ている人の評価値を重要視しようということです。割り算をしているのは、5段階評価のレンジを超えないように正規化しているためです。

7.2.5 アイテム間型協調フィルタリング

アイテム間型協調フィルタリングもメモリベースの手法で、ピアソンの相関係数やコサイン類似度などで類似度が高いアイテムを見つけるという解き方をします。計算方法としてはユーザー間型協調フィルタリングと大きく変わりません。また、コサイン類似度を改良した調整済みコサイン類似度 (Adjusted Cosine Similarity) という類似度もあります。[sarwar2001]

ユーザーUの評価ベクトルをu、ユーザーVの評価ベクトルをv、映画Mの評価ベクトルをm（評価行列を縦に一列切り取ったもの）、映画Nの評価ベクトルをnとすると、アイテム間型協調フィルタリングのピアソンの相関係数は、以下のように書けます。

```
def item_pearson_coefficient(u, v, m, n):
    u_diff = u - np.mean(m)
    v_diff = v - np.mean(n)
    numerator = np.dot(u_diff, v_diff)
    denominator = np.sqrt(sum(u_diff **2)) * np.sqrt(sum(v_diff **2))
    return numerator / denominator
```

アイテム間型協調フィルタリングでは、分母で映画Mの評価値の平均を引くのですが、その代わりにユーザーの評価値の平均で引いたものが、調整済みコサイン類似度です。調整済みコサイン類似度は以下のように書けます。

```
def adjusted_pearson_coefficient(u, v):
    u_diff = u - np.mean(u)
```

```

v_diff = v - np.mean(v)
numerator = np.dot(u_diff, v_diff)
denominator = np.sqrt(sum(u_diff **2)) * np.sqrt(sum(v_diff **2))
return numerator / denominator

```

これはちょうどユーザー間型協調フィルタリングのピアソンの相関係数を求めるのと同じことになります。ユーザーの評価値の平均を割り引くことで、高めの点数に評価をするユーザーと低めに評価をするユーザーの違いを除去することができ、予測性能が向上します。

ユーザーベースとアイテムベースでは、どちらのデータの増加スピードが早いかが更新頻度の鍵になります。また、アイテムに対する類似アイテムを表示するのは、まだアクティビティが少ないユーザーに対しても提示できるため、登録してすぐのユーザーにもアプローチしやすい方法です。反面、人気のアイテムばかりが推薦されるという課題も顕著なため、意外性を出すには工夫が必要です。

7.2.6 モデルベース協調フィルタリング

モデルベース協調フィルタリングは、教師あり学習や教師なし学習のモデルを作ることで、既知のデータの規則性をもとに予測をする方法です。ここでいうモデルには、クラスタリングを使ったモデルや、評価値を回帰などで予測するモデル、トピックモデル (Topic Model) を使った方法、行列分解 (Matrix Decomposition) を使った方法などがあります。

クラスタリングを使ったモデルは嗜好が似ているユーザーのグループを作成して、あるユーザーが所属するグループに好まれるもの推荐する方法です。

回帰を使った予測モデルは、線形回帰などの回帰モデルを学習して評価値を予測します。

トピックモデルは、PLSA (Probabilistic Latent Semantic Analysis) や LDA (Latent Dirichlet Allocation) など、評価行列を次元削減することで「アクションが好き」などの潜在的な意味を表現できるとされている手法です。暗黙的データでも予測ができるのが特徴です。

行列分解はもともと未評価と不支持の差を考慮しなければならないで、明示的データでなければなりませんでした。しかし近年では、暗黙的なデータへの拡張が行われています [implicitfm]。有名なのは Matrix Factorization と呼ばれる方法で、レーティングの評価行列をユーザーの行列とアイテムの行列で表現します。m人のユーザーとn個のアイテムの評価行列Rがあるときに、図7-1のようにユーザー行列Uとアイテム行列Iに分解します。ここで、dはハイパーパラメータ、評価行列Rはほとんどの要素で値が存在しない疎行列であり、ユーザー行列Uとアイテム行列Iは密行列であることに注意して下さい。

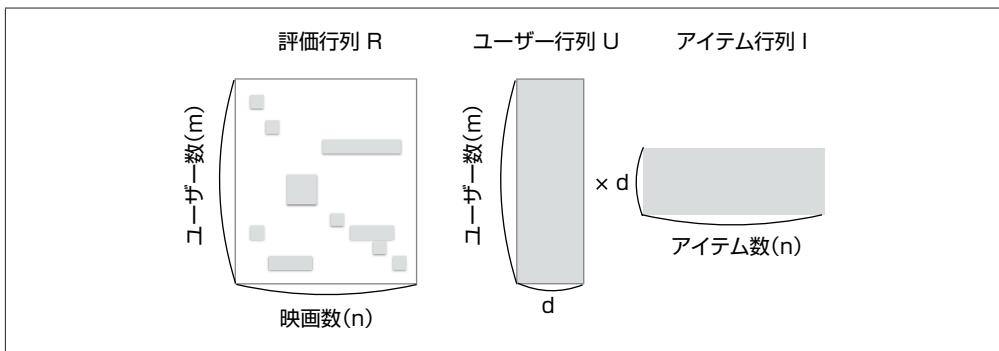


図7-1 Matrix Factorizationのイメージ

Matrix Factorizationは、確率的勾配法(SGD)を最適化手法として使うことで大規模データに対しても学習ができます。

7.2.7 内容ベースフィルタリング

内容ベースフィルタリングは、映画のタイトルや監督、ジャンルや俳優、口コミなどアイテムを表現する情報に着目し、それらの過去のデータから推薦を行います。ユーザーの嗜好を表現する単語が獲得できれば、それをベースに嗜好にあう映画を提示することが出来ます。

7.2.8 協調フィルタリングと内容ベースフィルタリングの得手・不得手

協調フィルタリングは、ジャンルやテキストに含まれている単語が似ていなくても良いため、多様性のある推薦結果を得られる確率が高いでしょう。また、ドメイン知識を管理する必要がありません。一方で、協調フィルタリングは新規ユーザーや新アイテムに対する推薦がデータが少なくなりがちで推薦が特に困難です(コールドスタート問題)。また、システムの利用者が少ないと良い推薦ができず、推薦が使われないと利用者が増えないという負のループにはまるリスクがあります。

内容ベースフィルタリングは、新しいサービスで行動データが蓄積されていなくても、比較的適切な推薦をしやすい手法です。しかし、特に日本語だとデータを形態素解析するための辞書をメンテナンスする必要があるなど、ドメインに特化した情報をどのように扱うかが課題となります。

7.2.9 評価尺度

推薦にはいくつかの評価尺度があります。正解率(Accuracy)は、例えば5段階評価で4以上の評価値が付与されていた場合に適合するとみなして、予測結果とユーザーの評価値を比較します。適合率(Precision)、再現率(Recall)は予測結果に対する正解の割合である適合率、真の正解に対するカバー率である再現率を計算します。これらは主に分類問題の評価に使われます。

推薦の結果として、回帰で星の数などの点数を予測する場合は回帰用の評価尺度を用います。平均絶対誤差 (Mean Average Error、MAE) は回帰で利用する評価尺度で、予測値と実測値の差の絶対値を評価用データで算出します。ズレが大きいと平均絶対誤差の値は大きくなります。平均二乗誤差 (Root Mean Squared Error、RMSE) も回帰で利用する評価尺度で、予測値と実測値の二乗した差を元にしたデータです。ズレが大きいと平均二乗誤差の値は大きくなります。MAEに比べて外れ値に弱いと言われています。

順位相関 (Rank Correlation) は推薦するアイテムの並びを評価する指標です。こちらは順位を学習する場合に使うランキング学習 (Learning to Rank) を行った際によく利用されます。

これ以外にも、多様性 (Diversity) を評価指標に取り入れる場合もあれば、全アイテムのうち評価値の予測が可能なアイテムの割合として被覆率 (Coverage) を評価指標として使う場合もあります。

7.3 MovieLensのデータの傾向を見る

それでは、MovieLensのデータがどのようなデータなのかを調べてみましょう。まず、データをダウンロードします。

以降のコードは、レポジトリ (<https://github.com/oreilly-japan/ml-at-work>) にあります。レポジトリの chap07/download.sh にダウンロード用のスクリプトがあるので、Linux系の環境の人は利用して下さい。

```
wget http://files.grouplens.org/papers/ml-100k.zip  
unzip ml-100k.zip
```

実際にデータを見ていきましょう。ノートブックはレポジトリの chap07/Movie recommendation.ipynb にあります。まずは、ユーザー情報、評価値情報、映画情報と一緒にデータの様子を見てていきましょう。ユーザー情報を読み込んで pandas の DataFrame に格納します。

```
import pandas as pd  
  
u_cols = ['user_id', 'age', 'sex', 'occupation', 'zip_code']  
users = pd.read_csv('ml-100k/u.user', sep='|', names=u_cols)  
users.head()
```

	user_id	age	sex	occupation	zip_code
0	1	24	M	technician	85711
1	2	53	F	other	94043
2	3	23	M	writer	32067
3	4	24	M	technician	43537
4	5	33	F	other	15213

図 7-2 ユーザー情報（一部）

ユーザー ID、年齢、性別、職業、郵便番号が含まれているのが分かります。同様に評価値情報を読み込みます。

```
r_cols = ['user_id', 'movie_id', 'rating', 'unix_timestamp']
ratings = pd.read_csv('ml-100k/u.data', sep='\t', names=r_cols)
ratings['date'] = pd.to_datetime(ratings['unix_timestamp'], unit='s')
ratings.head()
```

	user_id	movie_id	rating	unix_timestamp	date
0	196	242	3	881250949	1997-12-04 15:55:49
1	186	302	3	891717742	1998-04-04 19:22:22
2	22	377	1	878887116	1997-11-07 07:18:36
3	244	51	2	880606923	1997-11-27 05:02:03
4	166	346	1	886397596	1998-02-02 05:33:16

図 7-3 評価値情報（一部）

ユーザー ID、映画ID、5段階の評価値、評価したUNIX時間が入っています。UNIX時間は人には分かりづらいので、dateフィールドにパースした日時を入れています。

最後に、映画情報を読み込みます。

```
m_cols = ['movie_id', 'title', 'release_date',
          'video_release_date', 'imdb_url']
movies = pd.read_csv('ml-100k/u.item', sep='|',
                     names=m_cols, usecols=range(5), encoding = "latin1")
movies.head()
```

	movie_id	title	release_date	video_release_date	imdb_url
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(1995)
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)

図 7-4 映画情報（一部）

映画ID、映画名、公開日、ビデオ発売日、IMDb^{†3}というインターネット・ムービー・データベースのURLが書かれています。評価値の予測に使うため、すべての情報をマージします。

```
movie_rating = pd.merge(movies, ratings)
lens = pd.merge(movie_rating, users)
```

全データの中で、最も評価された25作品のタイトルを見てみましょう。

```
lens.title.value_counts()[:25]
# Star Wars (1977)           583
# Contact (1997)            509
# Fargo (1996)              508
# Return of the Jedi (1983)  507
# ...
# Name: title, dtype: int64
```

1位はスター・ウォーズの583件です。上位の映画を見ると2000年以前の映画が多いですね。これは、古い映画の方が評価する人が多くなるという理由が考えられます。

次に、評価の数と平均を集計し、平均値の高い順に並べ替えをします。

```
movie_stats = lens.groupby('title').agg({'rating': [np.size, np.mean]})
movie_stats.sort_values(by=[('rating', 'mean')], ascending=False).head()
```

するとどうでしょう、評価件数が1件と少ないため評価の平均が高くなる映画が上位に来てしましました。これを防ぐためにはどうすればよいでしょうか。

†3 <http://www.imdb.com/>

	rating	
	size	mean
title		
They Made Me a Criminal (1939)	1	5
Marlene Dietrich: Shadow and Light (1996)	1	5
Saint of Fort Washington, The (1993)	2	5
Someone Else's America (1995)	1	5
Star Kid (1997)	3	5

図 7-5 平均評価値上位の映画

1つの方法として、評価数が少ないと平均にノイズが乗りやすいので、評価数が多いもののみで平均することを考えてみましょう。以下のコードでは100件以上の評価が付いている映画の中で、評価値の平均が高い順に並べ替えていきます。

```
atleast_100 = movie_stats['rating']['size'] >= 100
movie_stats[atleast_100].sort_values(by=['rating', 'mean'],
                                   ascending=False)[:15]
```

	rating	
	size	mean
title		
Close Shave, A (1995)	112	4.491071
Schindler's List (1993)	298	4.466443
Wrong Trousers, The (1993)	118	4.466102
Casablanca (1942)	243	4.456790
Shawshank Redemption, The (1994)	283	4.445230
Rear Window (1954)	209	4.387560
Usual Suspects, The (1995)	267	4.385768
Star Wars (1977)	583	4.358491
12 Angry Men (1957)	125	4.344000
Citizen Kane (1941)	198	4.292929
To Kill a Mockingbird (1962)	219	4.292237
One Flew Over the Cuckoo's Nest (1975)	264	4.291667
Silence of the Lambs, The (1991)	390	4.289744
North by Northwest (1959)	179	4.284916
Godfather, The (1972)	413	4.283293

図 7-6 100件以上評価数のある平均評価値上位の映画

今度は想像していたランキングになりました。評価回数の分布はどうなっているでしょうか。ヒストグラムを見てみましょう。

```
from matplotlib import pyplot as plt
plt.style.use('ggplot')

lens.groupby('user_id').size().sort_values(ascending=False).hist()

plt.xlabel('rating size')
plt.ylabel('count of rating')
```

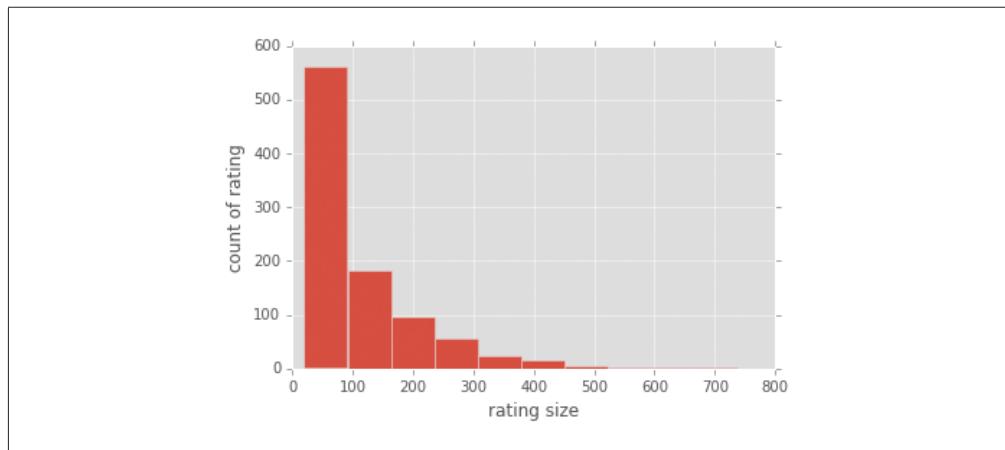


図7-7 評価値のヒストグラム

テールにしたがって頻度が低くなるいわゆるロングテールな分布になっています。これをジップの法則 (Zipf's law) に従っていると言います。つぎに、ユーザーごとの評価数と評価値の平均について調べてみましょう。

```
user_stats = lens.groupby('user_id').agg({'rating': [np.size, np.mean]})

user_stats['rating'].describe()
```

	size	mean
count	943.000000	943.000000
mean	106.044539	3.588191
std	100.931743	0.445233
min	20.000000	1.491954
25%	33.000000	3.323054
50%	65.000000	3.620690
75%	148.000000	3.869565
max	737.000000	4.869565

図7-8 ユーザーごとの評価数と評価値

特に評価値の平均に着目をすると、最低が平均1.49点の辛口のユーザーから4.87点の甘めのユーザーまで、ユーザーごとにバイアスがあることが分かります。

7.4 推荐システムの実装

それでは、MovieLensのデータを使って、映画の評価値を予測しましょう。

7.4.1 Factorization Machineを使った推薦

今回は、Matrix Factorizationを一般化したアルゴリズムであるFactorization Machineを使って推薦します。(図7-9、[fm2012]) Factorization Machineには以下のような特徴があります。

- Matrix Factorizationではユーザーとアイテムの情報しか扱えなかったが、Factorization Machineはそれ以外の特徴量も扱うことができる
- ロジスティック回帰などと異なり、Matrix Factorizationと同じく疎な行列を扱うことができる
- 特徴量の間で影響を与え合う交互作用(Interaction)を考慮することができるので、相関関係がある特徴量も適切に扱うことができる

カテゴリカルなデータをダミー変数として表現することで、カテゴリ同士の交互作用の影響を扱います。

Factorization Machineの各種ライブラリでは、回帰、分類、順位を学習するランク学習(Learning to Rank)を行えます。

Feature vector \mathbf{x}										Target y													
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$	
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$	
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(3)}$	
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(4)}$	
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(5)}$	
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(6)}$	
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$	
User			A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Time	TI	NH	SW	ST	...	Last Movie rated

図7-9 Factorization Machinesのイメージ図 (図は[fm2012]より引用)

Factorization Machineの実装は提案者の実装したlibFM^{†4}というライブラリが有名ですが、libFMはC++で書かれているため今回はそのPython実装であるfastFM^{†5}を使います。fastFMのinstallはpipでできます。

```
$ pip install fastFM
```

もし、pipでのインストールがうまくいかない場合はGitHubのレポジトリをcloneして、READMEに従って手動でインストールします。

オリジナルのlibFMと同様に、fastFMは評価値の回帰、二値分類、ランキング学習ができます。fastFMの利用するアルゴリズムにはAlternated Least Squared (ALS) か確率的勾配法 (SGD)、MCMC (Markov Chain Monte Carlo Methods、マルコフチェインモンテカルロ法) があります。

それぞれのアルゴリズムの長所と短所は以下のとおりです。

- ALS

- 長所：予測が速い、パラメータがSGDより少ない
- 短所：正則化が必須

- SGD

- 長所：予測が速い、大きなデータに対しても高速に学習ができる
- 短所：正則化が必須、ハイパーパラメータの数が多い

^{†4} <http://www.libfm.org/>

^{†5} <https://github.com/ibayer/fastFM>

● MCMC

- 長所：ハイパーパラメータの数が少ない、自動で正則化できる
- 短所：学習に一定の時間が必要

まず、手始めにALSを使って学習と予測をしてみましょう。

fastFMの入力は、ユーザーIDとアイテムIDについてはカテゴリカル変数として考え、ダミー変数に変換する必要があります。この変換のためにscikit-learnのDictVectorizerクラスを使います。

```
from sklearn.feature_extraction import DictVectorizer

# 入力データの生データ（ラベルである点数は含まれない）
# ユーザーID、評価したアイテムID、ユーザーの年齢の含まれる辞書
train = [
    {"user": "1", "item": "5", "age": 19},
    {"user": "2", "item": "43", "age": 33},
    {"user": "3", "item": "20", "age": 55},
    {"user": "4", "item": "10", "age": 20},
]

# DictVectorizer() を使い、age以外のフィールドをダミー変数へ変換する
v = DictVectorizer()
X = v.fit_transform(train)
print(X.toarray())

# user, itemはstringとして渡したため、ダミー変数に変換された
# [[ 19.  0.  0.  0.  1.  1.  0.  0.  0.]
#  [ 33.  0.  0.  1.  0.  0.  1.  0.  0.]
#  [ 55.  0.  1.  0.  0.  0.  0.  1.  0.]
#  [ 20.  1.  0.  0.  0.  0.  0.  0.  1.]]
```

このように、ユーザーIDとアイテムIDを文字列で表現することで、カテゴリカル変数として扱います。

それでは、試しにこのダミーデータに対してALSを使って回帰をしてみましょう。

```
from fastFM import als
import numpy as np

# 先程のダミーデータに対して、19歳のユーザーは5点、33歳のユーザーは1点、
# 55歳のユーザーは2点、20歳のユーザーは4点を付けたとする
y = np.array([5.0, 1.0, 2.0, 4.0])
# ALSで回帰のFMモデルを初期化、学習する
fm = als.FMRegression(n_iter=1000, init_stdev=0.1, rank=2,
                      l2_reg_w=0.1, l2_reg_V=0.5)
fm.fit(X, y)
# 24歳のユーザーID 5がアイテムID 10を評価した場合のrateを予測する
fm.predict(v.transform({"user": "5", "item": "10", "age": 24}))
# array([ 3.60775939])
```

このように24歳のユーザーID 5がアイテムID 10の評価値を予測した場合、年齢も近く同じ商品を

評価しているユーザー4の評価値に近い5段階評価のうち、3.6という評価を予測することが出来ました。

7.4.2 いよいよFactorization Machineで学習する

全体の傾向が分かったので、実際に映画の評価値を予測しましょう。実はMovieLensのデータには、ユーザーをうまく分けた開発データとテストデータが提供されています。まずは、それを使って評価値の予測をしてみましょう。

データを読み込む関数を定義して、開発データua.baseとテストデータua.testを読み込みます。

```
def load_data(filename, path="ml-100k/"):
    data = []
    y = []
    with open(path+filename) as f:
        for line in f:
            (user, movieid, rating, ts) = line.split('\t')
            data.append({"user_id": str(user), "movie_id": str(movieid)})
            y.append(float(rating))

    return (data, np.array(y))

(dev_data, y_dev) = load_data("ua.base")
(test_data, y_test) = load_data("ua.test")
```

このとき、Factorization MachineではユーザーIDと映画のIDをカテゴリカル変数として扱うため文字列型に変換し、DictVectorizerクラスを使ってダミー変数にしています。

次に開発データを訓練データと検証データに分けます。パラメータの調整には訓練データと検証データを使い、最終的な評価にはテストデータを使います。

```
from sklearn.feature_selection import train_test_split

v = DictVectorizer()
X_dev = v.fit_transform(dev_data)
X_test = v.transform(test_data)
np.std(y_test)
X_train, X_dev_test, y_train, y_dev_test =
    train_test_split(X_dev, y_dev, test_size=0.1, random_state=42)
```

X_trainが訓練データ、y_trainがその評価値、X_dev_testが検証データ、y_dev_testがその評価値になります。train_test_split()関数を使って9:1にデータを分割しています。

データが準備できたので、Factorization Machineを使って学習してみましょう。ここでは、ハイパーパラメータの少なさからMCMCを使って学習をします。fastFMでMCMCを使って学習するときには、その特性上学習と予測を同時にしなければいけない制約があります。つまり、fit()関数とpredict()関数が分離できず、fit_predict()という関数でしか学習と予測ができません。そのため、

オンラインで予測したいときなど予測時間がネックになる場合は、最適化手法としてSGDを使うほうが良いでしょう。

MCMCで学習をして、イテレーションの回数に対する平均二乗誤差とMCMCのハイパーパラメタ(α, λ_w, μ_w)の推移を見てみましょう。

```
from sklearn.metrics import mean_squared_error
from fastFM import mcmc

# fastFMのパラメータの指定
n_iter = 300
step_size = 1
seed = 123
rank = 4

# MCMCを使った回帰のFMモデルを初期化する
fm = mcmc.FMRegression(n_iter=0, rank=rank, random_state=seed)
fm.fit_predict(X_train, y_train, X_dev_test)

rmse_dev_test = []
rmse_test = []
hyper_param = np.zeros((n_iter - 1, 3 + 2 * rank), dtype=np.float64)

# イテレーション回数を変化させて、予測結果の性能とハイパーパラメータを得る
for nr, i in enumerate(range(1, n_iter)):
    fm.random_state = i * seed
    y_pred = fm.fit_predict(X_train, y_train, X_dev_test, n_more_iter=step_size)
    rmse_test.append(np.sqrt(mean_squared_error(y_pred, y_dev_test)))
    hyper_param[nr, :] = fm.hyper_param_

# 最初の5回は値が落ちていいないので無視する
values = np.arange(1, n_iter)
x = values * step_size
burn_in = 5
x = x[burn_in:]

# RMSEとハイパーパラメータをプロットする
from matplotlib import pyplot as plt
fig, axes = plt.subplots(nrows=2, ncols=2, sharex=True, figsize=(15, 8))

axes[0, 0].plot(x, rmse_test[burn_in:], label='dev test rmse', color="r")
axes[0, 0].legend()
axes[0, 1].plot(x, hyper_param[burn_in:, 0], label='alpha', color="b")
axes[0, 1].legend()
axes[1, 0].plot(x, hyper_param[burn_in:, 1], label='lambda_w', color="g")
axes[1, 0].legend()
axes[1, 1].plot(x, hyper_param[burn_in:, 3], label='mu_w', color="g")
axes[1, 1].legend()
```

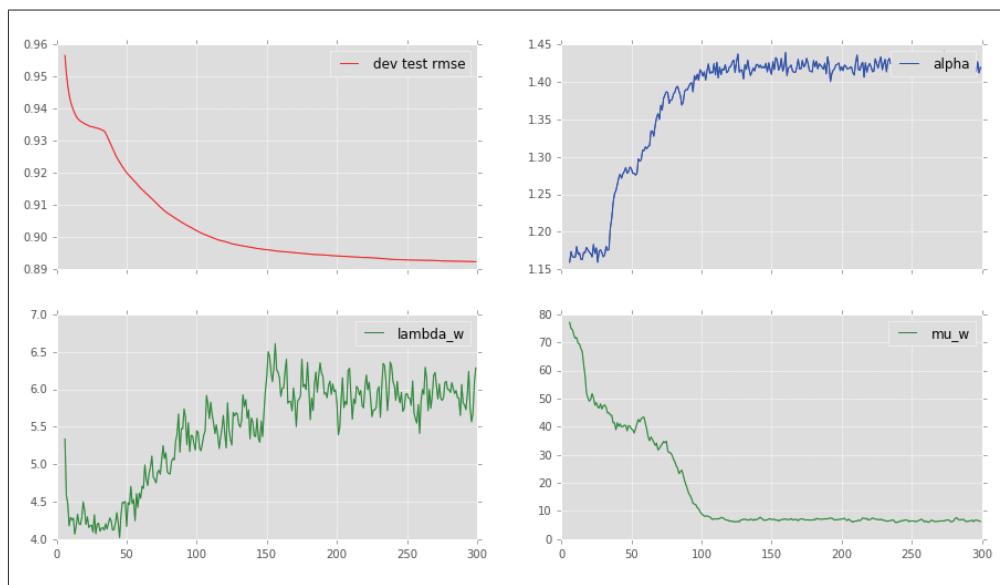


図 7-10 MCMC のイテレーションに対する平均二乗誤差とパラメータ

およそ 100 回イテレーションを繰り返すと、どのパラメータも収束して来ているのが分かります。標準偏差は、回帰問題における平均二乗誤差のベンチマークとなります。常に平均値を出力する予測モデルがあったとき、その平均二乗誤差は標準偏差に等しくなることが知られています。検証データの標準偏差は 1.12 なのに対し、今回の予測した評価値の平均二乗誤差は 0.89 と低い値になっています。

次に、行列を圧縮する次数である rank (Matrix Factorization における d と同じようなハイパーパラメータ) を大きくした時に性能がどう変わっていくかを見てみましょう。

```
n_iter = 100
seed = 333

rmse_test = []
# rank を 4, 8, 16, 32, 64 で探索する
ranks = [4, 8, 16, 32, 64]

# rank を変えて学習・予測をし、dev test データに対する RMSE を獲得する
for rank in ranks:
    fm = mcmc.FMRegression(n_iter=n_iter, rank=rank, random_state=seed)
    y_pred = fm.fit_predict(X_train, y_train, X_dev_test)
    rmse = np.sqrt(mean_squared_error(y_pred, y_dev_test))
    rmse_test.append(rmse)
    print('rank:{}\trmse:{:.3f}'.format(rank, rmse))

# 各 rank 毎の RMSE をプロットする
plt.plot(ranks, rmse_test, label='dev test rmse', color="r")
plt.legend()
```

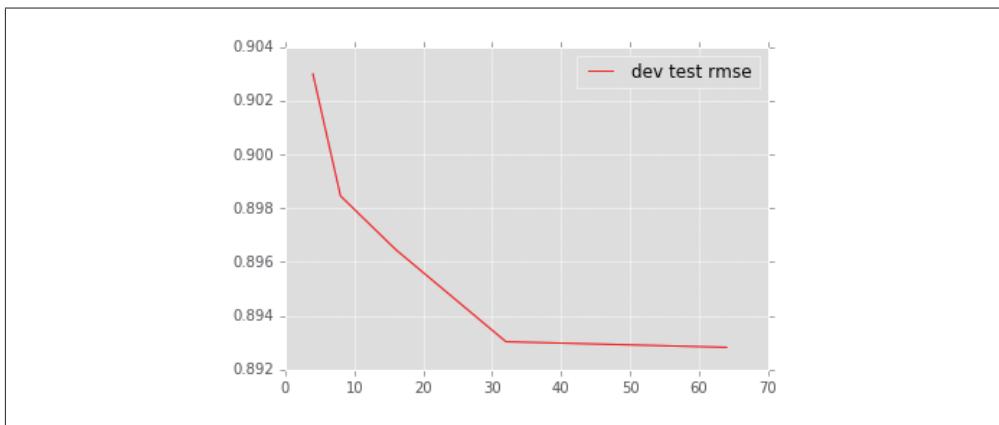


図 7-11 Rank 毎の平均二乗誤差

rank は 32になると平均二乗誤差の変化が落ち着いてくることが分かりました。rank が大きくなればなるほど、考慮すべき交互作用が増えて学習対象となる重みの個数が増えるため、学習時間も長くなります。適当なサイズを見極めるのが重要です。

それでは、テストデータで評価してみましょう。

```
fm = mcmc.FMRegression(n_iter=300, rank=32, random_state=seed)
y_pred = fm.fit_predict(X_train, y_train, X_test)
np.sqrt(mean_squared_error(y_pred, y_test))
```

平均二乗誤差が 0.921となりました。ここで、データの傾向を調べた時、ユーザーごとに評価値のバラつきがあったのを思い出してください。これに対処をするためには、評価値の標準化をすると良さそうです。今回はシンプルに StandardScaler クラスを使ってみましょう。StandardScaler クラスは平均を 0 に、標準偏差を 1 に合わせるように標準化します。

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
y_train_norm = scaler.fit_transform(y_train.reshape(-1, 1)).ravel()
fm = mcmc.FMRegression(n_iter=300, rank=32, random_state=seed)
y_pred = fm.fit_predict(X_train, y_train_norm, X_test)
np.sqrt(mean_squared_error(scaler.inverse_transform(y_pred), y_test))
```

標準化の結果、平均二乗誤差が 0.920と僅かですが小さくなりました。

7.4.3 ユーザーと映画以外のコンテキストも加える

最後に、ユーザー ID と映画 ID 以外の情報を加えてみましょう。用意するデータとしては、映画の公開年、ユーザーの年齢、性別、評価年を加えてみます。まずは、各特徴量の前処理をします。

```

lens['user_id'] = lens['user_id'].astype(str)
lens['movie_id'] = lens['movie_id'].astype(str)
lens['year'] = lens['date'].apply(str).str.split('-').str.get(0)
lens['release_year'] = \
    lens['release_date'].apply(str).str.split('-').str.get(2)
lens['year'] = lens['date'].apply(str).str.split('-').str.get(0)
lens['release_year'] = \
    lens['release_date'].apply(str).str.split('-').str.get(2)

```

ユーザーID、映画IDはダミー変数に変換をするため文字列に変換しています。映画の公開年と評価年をそれぞれリリース日と評価日から抽出しています。これらの特徴量の組み合わせ候補を作っています。

```

candidate_columns = [
    ['user_id', 'movie_id', 'release_year', 'age', 'sex', 'year', 'rating'], #A
    ['user_id', 'movie_id', 'age', 'sex', 'year', 'rating'], #B
    ['user_id', 'movie_id', 'sex', 'year', 'rating'], #C
    ['user_id', 'movie_id', 'age', 'sex', 'rating'], #D
    ['user_id', 'movie_id', 'rating'], #E
]

```

それぞれの組み合わせをAからEと名付けた時、どの組み合わせが良いか実際に学習してみましょう。

```

rmse_test = []

# カラム候補毎ごとに評価を行う
for column in candidate_columns:
    # 欠損値を落とす
    filtered_lens = lens[column].dropna()
    # 入力データをダミー変数に変換する
    v = DictVectorizer()
    X_more_feature = v.fit_transform(
        list(filtered_lens.drop('rating', axis=1).T.to_dict().values()))
    # 教師となるレーティングを代入する
    y_more_feature = filtered_lens['rating'].tolist()

    # 教師データの学習用と評価用の分割
    X_mf_train, X_mf_test, y_mf_train, y_mf_test = train_test_split(
        X_more_feature, y_more_feature, test_size=0.1, random_state=42)

    # ratingの正規化をする
    scaler = StandardScaler()
    y_mf_train_norm = scaler.fit_transform(np.array(y_mf_train)).ravel()

    # MCMCを使ったモデルの学習
    fm = mcmc.FMRegression(n_iter=500, rank=8, random_state=123)
    fm.fit_predict(X_mf_train, y_mf_train_norm, X_mf_test)

    # テストデータでの予測結果のRMSEの取得
    y_pred = fm.fit_predict(X_mf_train, y_mf_train_norm, X_mf_test)

```

```

rmse = np.sqrt(
    mean_squared_error(scaler.inverse_transform(y_pred), y_mf_test))
rmse_test.append(rmse)

# RMSE をプロットする
ind = np.arange(len(rmse_test))
bar = plt.bar(ind, height=rmse_test)
plt.xticks(ind, ('A', 'B', 'C', 'D', 'E'))
plt.ylim((0.88, 0.90))

```

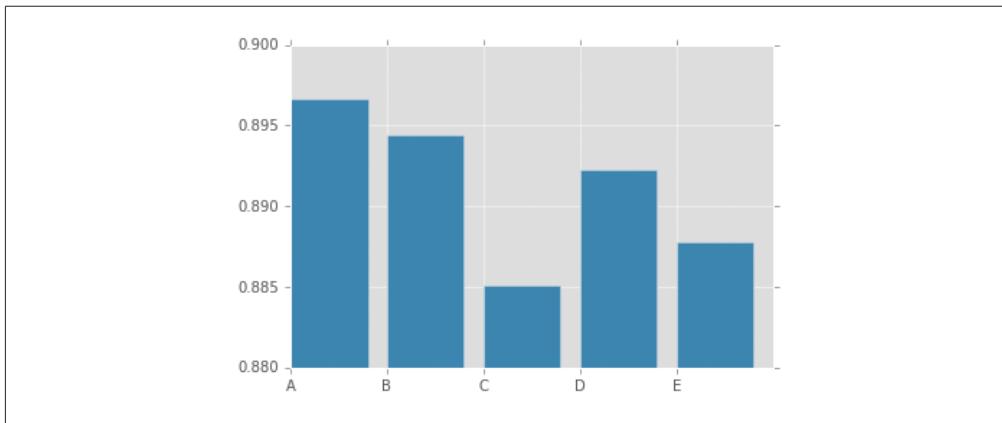


図7-12 各特微量の組み合わせ毎の平均二乗誤差

予測結果の平均二乗誤差を図7-12に図示します。結果より、CのユーザーID、映画ID、性別、評価年などの組み合わせが良いことが分かりました。平均二乗誤差も0.885と低めにおさえられています。

このように、Factorization Machineは気軽に特微量の組み合わせを様々なものに変えることができ、どのような文脈を加えればより良い性能になるかをとても簡単に検証できます。こうしたユーザーとアイテム以外の情報を用了レコメンドを**Context-Aware Recommendation**と呼びます [[contextawarerecom](#)]。

今回の予測結果をDBに格納することで、Webアプリケーションから参照して利用しましょう。なお、MCMCの制約としてリアルタイム処理で予測するのは難しいですが、リアルタイムに予測したい場合はSGDを使ったモデルを構築すれば、リアルタイム処理でも評価値を得られます。

7.5 この章のまとめ

本章では、映画の評価値の予測を通じて推薦システムの概要を学び、実際に Factorization Machine を用いた推薦を行いました。Factorization Machineを使った評価値の予測では、ユーザー ID や映画 ID 以外にも適切な特徴量の組み合わせを用いて予測をすることで性能が改善されることが分かりました。

実際にシステムとして使う場合は、評価値がないため苦労することが多いでしょう。評価値を取得するためのインタラクションはシステムとしてユーザーメリットがある形で組み込まなければデータとして集まりません。また、予測対象とするデータはユーザーの嗜好が現れるデータを適切に利用するように心がけましょう。特に、その対象が最終的に予測したい値なのか、それとも特徴量として使える途中的データなのかを切り分けることが重要です。

8章

Kickstarterの分析、機械学習を使わないという選択肢

本章では、実際にデータ分析を行いながら、データ分析を進めていく際の思考と、その結果どのようなレポートが出てくるのか、という過程を実際にお見せします。

本章では、Kickstarterをスクレイピングして得たデータをもとに、Excelを活用したデータ分析、上司に提出する用のレポート作成までをひと通り実務に即した形で体験してみるものです。



この分析は筆者がサイドワークで携わっていたプロジェクトにおいて、Kickstarterでの資金調達を検討した際に実際に行ったものです。結局、Kickstarterでの資金調達は実現しませんでした。しかし、Kickstarterの分析でいくつか面白い知見が得られたので、その内容を踏まえてご紹介します。

本章で利用したコードは <https://github.com/oreilly-japan/ml-at-work/tree/master/chap08> に置いてあります。合わせてご覧ください。

8.1 KickstarterのAPIを調査する

はじめに Kickstarter の API について調べてみましょう。まず「kickstarter api」で検索してみると、Does Kickstarter have a public API? - Stack Overflow^{†1} という記事が見つかります。

どうやら、Kickstarter の検索ページの URL に .json を指定すると、そのまま JSON 形式でデータが返ってくるという非公開 API があるようです。今回はこれを利用させてもらいましょう。

<https://www.kickstarter.com/projects/search?term=3d+printer>

<https://www.kickstarter.com/projects/search.json?term=3d+printer>

^{†1} <http://stackoverflow.com/questions/12907133/does-kickstarter-have-a-public-api>

8.2 Kickstarterのクローラを作成する

実際に、このAPIを使ったクローラを書いてみましょう。

今回の分析はKickstarterのTechnologyカテゴリに限定しています。これはTechnologyカテゴリ以外では分析の勘ドメイン知識が働かないで、ひとまずTechnologyに限定しています。

このAPIの問題点としては、Kickstarterは検索結果を、1リクエストあたり20件^{†2}返し、200ページまで検索でき、合計4000件しか得ることができないことにあります。そのため、検索時にTechnologyカテゴリ（カテゴリID=16）の下にあるサブカテゴリを指定することで検索結果を絞り込み、この制約を回避しようとしています。

今回のクローラはresultフォルダを作成し、この中に収集してきたプロジェクトのJSONを保存します。これにより、クローラと分析のコードを切り離すことができ、分析を行う際はローカルファイルに対して、高速にアクセスできるようになります。

```
import urllib.request
import json
import os
import time

os.makedirs('result', exist_ok=True)

search_term = ""
sort_key = 'newest'
category_list = [16, 331, 332, 333, 334, 335, 336,
                 337, 52, 362, 338, 51, 339, 340,
                 341, 342] # technology category
query_base = "https://www.kickstarter.com/projects/search.json?term=%s&category_"
id=%d&page=%d&sort=%s"

for category_id in category_list:
    for page_id in range(1, 201):
        try:
            query = query_base % (
                search_term, category_id, page_id, sort_key)
            print(query)
            data = urllib.request.urlopen(query).read().decode("utf-8")
            response_json = json.loads(data)
        except:
            break

        # 1ページあたり、20件の結果が帰ってくるので、1件ずつ保存する
        for project in response_json["projects"]:
            filepath = "result/%d.json" % project["id"]
            fp = open(filepath, "w")
```

^{†2} 1ページあたり20件は、本分析を行った2017年3月頃の話で、2017年10月現在は1ページあたり12件に減っています、ご注意ください。

```

fp.write(json.dumps(project, sort_keys=True, indent=2))
fp.close()

# 1アクセスごとに1秒のウェイトを入れることで、過剰アクセスを防止
time.sleep(1)

```

このクローラーを実行すると、result フォルダに JSON ファイルが outputされるので、軽く覗いてみましょう。ちなみに紙面の都合上、ここに掲載しているのは必要そうな項目を取り出したものです。

```
{
    "backers_count": 0,
    "country": "GB",
    "created_at": 1452453394,
    "currency": "GBP",
    "deadline": 1457820238,
    "disable_communication": true,
    "goal": 30000.0,
    "id": 1629226758,
    "launched_at": 1452636238,
    "name": "Ordering web page for small fast food businesses.",
    "pledged": 0.0,
    "slug": "ordering-web-page-for-small-fast-food-businesses",
    "state": "suspended",
    "state_changed_at": 1455831674,
    "static_usd_rate": 1.45221346,
    "urls": {
        "web": {
            "project": "https://www.kickstarter.com/projects/189332819/ordering-web-page-for-small-fast-food-businesses?ref=category_newest",
            "rewards": "https://www.kickstarter.com/projects/189332819/ordering-web-page-for-small-fast-food-businesses/rewards"
        }
    },
    "usd_pledged": "0.0"
}
```

8.3 JSON データを CSV に変換する

JSON を眺めていると、それらしいパラメータがちらほらあるので抽出してみましょう。フォルダに保存したファイルを読み込むには Python の標準ライブラリである glob を使います。ワイルドカードを使って指定することでフォルダ内のファイル名をリストとして簡単に取り出せます。

今後の分析は基本的に Excel に任せるため、Excel で分析のしやすい CSV 形式に変換します。CSV 形式への変換は、pandas の pandas.io.json.json_normalize(json_data)^{†3} という関数を利用します。

^{†3} http://pandas.pydata.org/pandas-docs/version/0.20.3/generated/pandas.io.json.json_normalize.html

Excelで読み込ませるときのコツとしては、CP932（Windowsで使われているShift_JISの拡張文字コード）でエンコードしてあげることや、Unix Timestamp型のカラムをDateTime型に変換するなどが挙げられます。

```
import glob
import json
import pandas
import pandas.io.json

project_list = []

# globでresultフォルダにあるファイルを走査して読み込む
for filename in glob.glob("result/*.json"):
    project = json.loads(open(filename).read())
    project_list.append(project)

# json_normalizeを使ってDataFrameに変換する
df = pandas.io.json.json_normalize(project_list)

# 末尾が"_at"で終わるunixtimeのカラムをdatetimeに変換する
datetime_columns = filter(lambda a: a[-3:] == "_at", df.columns)
for column in datetime_columns:
    df[column] = pandas.to_datetime(df[column], unit='s')

# DataFrameからCSV形式のstrに変換する
csv_data = df.to_csv()

# WindowsのExcelに読み込ませるので、CP932にする
csv_data = csv_data.encode("cp932", "ignore")

# 結果を書き込む
fp = open("kickstarter_result.csv", "wb")
fp.write(csv_data)
fp.close()
```

8.4 Excelで軽く眺めてみる

さて、抽出されたデータをExcelで眺めてみます。散布図行列などをつかってグラフの概形をつかむ方法もありますが、生データに対する雰囲気をつかむため、ランダムサンプリングデータでも良いので、Excel等で開いて眺めてみることをお勧めします。

生データの肌感覚があるとないのでは、分析効率が全く違います。機械学習を行うにしても、肌感覚がないと実行結果の異常に気付くことが難しいため、生データを眺める時間は絶対に必要です。

category	A	B	C	D	E	F	G	H	I	J	K	L	cur
2	tech/technology/hardware	0	50000 CAD	currency	backers_count	0 failed	Safer Home	2015/6/3 12:52	2015/7/3 12:52	Placing furniture a https://www.kick	2116322866	https://www.kick	1
3	tech/technology/gadgets	2030	87000 USD		15 canceled	Gizbee Unlimited	2016/3/6 4:30	2016/3/26 3:30	Gizbee is the first https://www.kick	1067733003	https://www.kick	1	
4	tech/technology/web	141	100000 USD		3 failed	Diposta - liberat	2016/7/24 20:18	2016/8/23 2018	The problem is b https://www.kick	1053577177	https://www.kick	1	
5	tech/technology/gadgets	3	25000 CAD		3 failed	Bent Spray Bottl	2016/3/20 10:57	2016/8/30 0:57	Ever had a spray b https://www.kick	1101368447	https://www.kick	1	
6	tech/technology/hardware	11393	260000 USD		19 succeeded	Smart Home Ther	2015/5/4 2:04	2016/7/28 2:04	Smart Home Ther https://www.kick	106669899	https://www.kick	1	
7	tech/technology/software	282	4000 EUR		12 failed	Gantish - Onlin	2015/6/17 22:37	2015/9/16 22:37	Taking the novelty https://www.kick	1511932511	https://www.kick	1	
8	tech/technology/web	25	15000 USD		1 failed	Sellergo start up	2016/3/5 10:13	2016/3/21 3:26	a simple and user https://www.kick	967162140	https://www.kick	1	
9	tech/technology/hardware	20	6000 USD		2 failed	Musical Light S	2011/5/19 16:09	2011/6/22 16:09	A musical Light S https://www.kick	095094580	https://www.kick	1	
10	tech/technology/flight	1355	8500 USD		4 failed	Hanger 1 Found	2014/9/4 2:40	2014/10/19 2:40	Seeking funding fo https://www.kick	1619263064	https://www.kick	1	
11	tech/technology/hardware	2691	250000 USD		561 successful	ODIN: Andromeda	2014/5/25 10:57	2014/12/1 10:57	First Moon Mission https://www.kick	106669899	https://www.kick	1	
12	tech/technology/hardware	11601	120000 AUD		18 failed	Timeless	2013/6/2 8:15	2013/9/30 8:15	Timeless https://www.kick	2123957850	https://www.kick	1	
13	tech/technology/hardware	656	20000 CAD		18 failed	The Forever Re	2013/9/30 6:26	2014/3/10 6:26	This talkin the https://www.kick	172024344	https://www.kick	1	
14	tech/technology/software	470566	40000 GBP		3080 successful	VLC for the rev	2012/11/23 21:54	2012/12/28 21:54	VLC for Windows https://www.kick	1061646922	https://www.kick	1	
15	tech/technology/apps	56680	55000 MXN		24 successful	APP - CUSTO...	2012/6/11 19:12	2016/6/12 19:12	Propuesta de Ap https://www.kick	49283059	https://www.kick	1	
16	tech/technology/fabricator	5000	5000 USD		51 successful	Student Operat	2012/6/11 20:55	2012/7/11 20:55	Completing the La https://www.kick	09805533	https://www.kick	1	
17	tech/technology/hardware	107	3500 USD		7 failed	A New Omni-Wi	2012/6/15 23:19	2012/6/24 23:19	On the wheel th https://www.kick	203625547	https://www.kick	1	
18	tech/technology/software	7157	100000 USD		184 successful	Intelligent Mind	2012/6/15 23:20	2012/6/15 23:20	Intelligent Mind https://www.kick	203625547	https://www.kick	1	
19	tech/technology/hardware	227673	25000 USD		22780 successful	UPU - I UPU	2011/6/10 17:00	2011/6/10 17:00	Universal Postage https://www.kick	819716725	https://www.kick	1	
20	tech/technology/apps	0	8500 USD		0 failed	Virtual Addiction	2014/7/24 23:49	2015/6/28 23:49	Help the addict the https://www.kick	1067635737	https://www.kick	1	
21	tech/technology/web	0	8000 USD		0 failed	College Spring E	2015/2/27 2:35	2015/3/29 2:35	The goal is to cre https://www.kick	252764955	https://www.kick	1	
22	tech/technology/gadgets	5650	50000 USD		22 canceled	CLEX - 4 in 1 S	2016/6/14 6:55	2016/7/19 5:55	World's first - PC https://www.kick	2016117568	https://www.kick	1	
23	tech/technology/apps	0	500 USD		0 failed	Stamped	2015/11/23 23:36	2016/1/22 23:36	The idea is a new https://www.kick	774461693	https://www.kick	1	
24	tech/technology/apps	0	1000 USD		0 failed	Croc2Pop - Wh	2014/7/22 4:25	2014/8/21 4:25	Croc2Pop is a cro https://www.kick	1403764871	https://www.kick	1	
25	tech/technology/gadgets	970	170000 USD		4 canceled	Wear The Sun	2014/7/22 4:25	2014/8/21 4:25	Wear The Sun https://www.kick	1403764871	https://www.kick	1	
26	tech/technology/technology	1777	100000 USD		21 successful	LAZER GOLF	2013/6/3 9:15	2014/6/23 6:50	Lighter Laser GOLF - The https://www.kick	2116322866	https://www.kick	1	
27	tech/technology/hardware	3810	15000 USD		34 failed	Smart Dive Buoy	2014/6/16 24:50	2014/7/26 14:50	Smart Dive Buddy https://www.kick	1617013047	https://www.kick	1	
28	tech/technology/web	410	18000 USD		4 failed	Goalzilla Let's	2015/3/20 5:01	2015/4/24 5:01	What do you want https://www.kick	165380405	https://www.kick	1	
29	tech/technology/software	1	98000 USD		1 failed	FREE SUPERROC	2014/4/10 9:46	2014/4/11 16:44	HELP US Build Thittes https://www.kick	1390235412	https://www.kick	1	
30	tech/technology/diy_electric	23447	10000 USD		298 successful	Equilo: Simple I	2015/5/11 22:52	2015/6/22 22:52	A development plaftform https://www.kick	389610462	https://www.kick	1	
31	tech/technology/apps	85	45000 USD		71 failed	Domino Home Te	2014/9/22 2:26	2014/10/17 2:26	Change the way h https://www.kick	6769911	https://www.kick	1	
32	tech/technology/technology	411598	250000 USD		236 failed	Robotic Robotic	2014/9/22 2:26	2014/10/17 2:26	Robotics https://www.kick	1961117568	https://www.kick	1	
33	tech/technology/technology	590	150000 USD		7 failed	Alrock to the hel	2017/1/3 0:55	2017/1/20 0:55	Your own control h https://www.kick	203625547	https://www.kick	1	
34	tech/technology/diy_electrics	3095	150000 USD		88 successful	Turn your reuse	2015/1/21 2:48	2016/7/29 2:48	A clean and profess https://www.kick	1807337189	https://www.kick	1	
35	tech/technology/apps	0	10000 USD		0 failed	Write my Estima	2015/5/29 2:32	2015/11/28 2:32	This will allow conf https://www.kick	976313294	https://www.kick	1	
36	tech/technology/hardware	312756	25000 USD		426 successful	Spank the ei	2012/2/22 10:22	2012/3/24 13:00	a big rock ruff https://www.kick	107975578	https://www.kick	1	
37	tech/technology/camera_elec	16988	8500 USD		135 successful	Phochon XA FI	2016/1/10 9:54	2016/11/18 9:54	600 a quick and accur https://www.kick	870664678	https://www.kick	1	
38	tech/technology/software	5	7000 USD		1 live	How safe is yo	2017/1/21 1:19	2017/3/19 7:19	Ingenious https://www.kick	195890303	https://www.kick	1	
39	tech/technology/camera_elec	5043	20000 USD		1 failed	Smart Aerial V	2014/1/21 1:19	2014/3/20 1:19	Smart Aerial V https://www.kick	1948293394	https://www.kick	1	
40	tech/technology/camera_elec	50	150000 USD		1 failed	Quakeless Paper	2014/4/9 2:55	2014/4/13 2:55	Quakeless Paper is a d https://www.kick	597002994	https://www.kick	1	
41	tech/technology/gadgets	6672	10000 CAD		115 canceled	Scupys turn ym	2015/1/21 2:26	2015/12/20 1:26	Scupys turn ym https://www.kick	1726276143	https://www.kick	1	
42	tech/technology/diy_electrical	149905	40000 USD		43 live	Long Range MU	2017/2/15 1:03	2017/7/31 11:00	11 serious GPS track https://www.kick	1836255117	https://www.kick	1	
43	tech/technology/web	0	5000 GBP		0 failed	New Social Net	2014/1/10 6:17	2014/11/15 8:10	The project is sim https://www.kick	1487316163	https://www.kick	1	
44	tech/technology/diy_electrical	0	5000 USD		0 canceled	PS5 (P)Cancele	2017/1/28 14:54	2017/3/9 14:54	PS5 Pro Rilis https://www.kick	251386942	https://www.kick	1	
45	tech/technology/gadgets	396299	300000 CAD		2421 successful	RiseRise The Ri	2016/4/6 5:27	2016/4/16 2:27	RiseRise converts https://www.kick	6592024494	https://www.kick	1	
46	tech/technology/apps	0	1000 USD		0 failed	Smart Alert Ap	2016/4/16 5:27	2016/4/16 5:27	Smart Alert App https://www.kick	99969262	https://www.kick	1	
47	tech/technology/web	216134	20000 USD		292 successful	Altair - Portabl	2015/3/9 22:26	2015/3/20 22:26	Altair is a portabl https://www.kick	188145877	https://www.kick	1	
48	tech/technology/apps	7	4000 USD		3 failed	Pocket Parking	2015/2/21 1:26	2015/2/27 1:26	Pocket Parking Me https://www.kick	122809751	https://www.kick	1	
49	tech/technology/camera_elec	30126	9000 USD		164 successful	Underwater IP91	2016/10/14 5:24	2016/11/16 4:50	Vy390 4K VR 360 https://www.kick	230566680	https://www.kick	1	

図8-1 生データを眺める

`predged` (調達金額) と `goal` (目標金額) から達成率を、`backers_count` (支援者数) から一人当たりの Back (支援) 金額のカラムを足してみます。データを抽出する段階で計算しても良いのですが、筆者は Excel で計算しています。

まずは達成率を降順にソートして、グラフにしてみましょう。

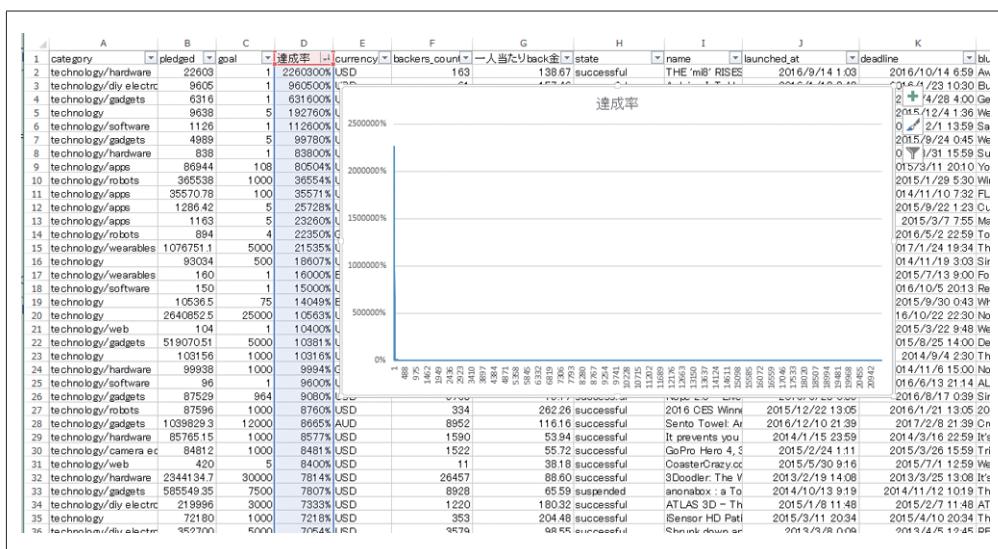


図8-2 達成率を降順にソートしてグラフ化

ゴール金額を1ドルに設定している人がいるため、グラフが壊れてしまっているようです。縦軸の上界を500%にしてみます。

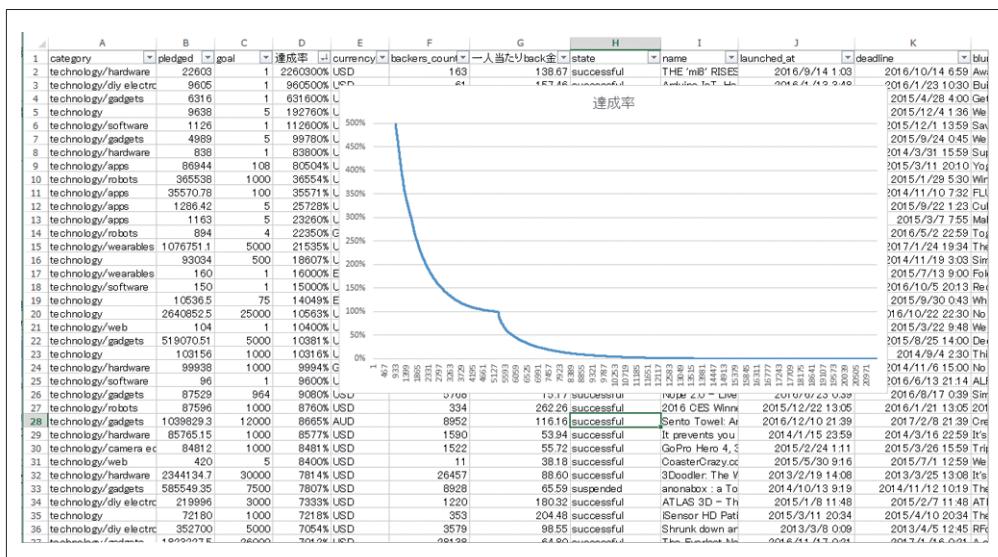


図8-3 縦軸の上界を500%にする

すると100%地点に謎の特異点が生まれました。これはKickstarterの効果でしょうか。達成率とい

う軸で見ると何か発見がありそうです。

次はstateで絞りこんで、終了したプロジェクトと終了前のプロジェクトを比較してみます。

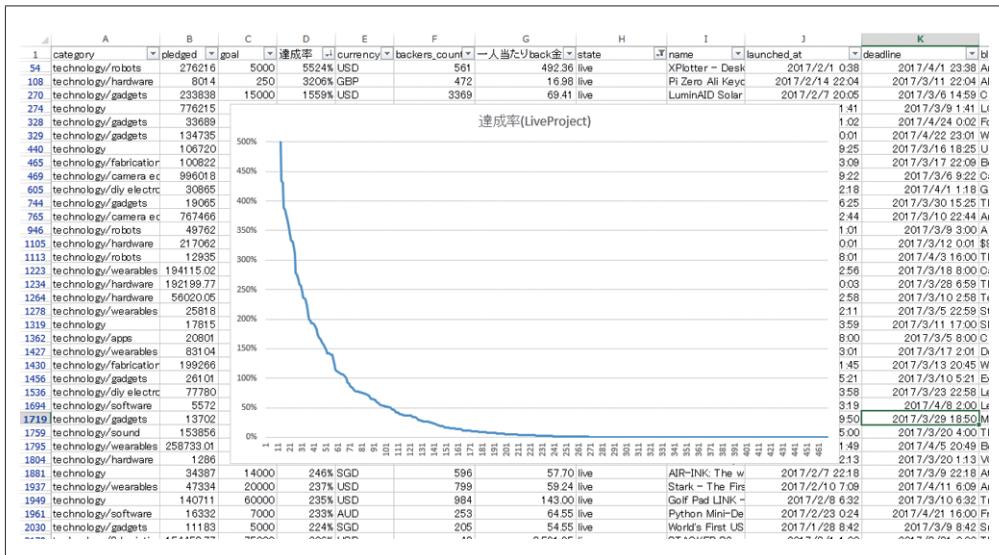


図8-4 終了前のプロジェクトの達成率分布

どうやらstateがLiveのまだ締め切りに到達していない案件に、この特異点は存在しないようです。ということは、この特異点はプロジェクト終了直前に生まれるらしいということが分かりました。理由としては次のようなものが考えられます。

- Kickstarterが終了間際のプロジェクトをトップページで紹介している
 - 終了間際にBackされる理由にはなるが、100%近辺に特異点が生まれる理由にはならない
- ギリギリの場合、当人たちが終わり際に頑張って宣伝する
- 身内が最後のひと押しのBackを行っている
- 勝ち馬に乗りたい人が、達成しそうなプロジェクトにBackしている
- 最後のひと押しの快感を味わいたい人がいる

8.5 ピボットテーブルでいろいろと眺めてみる

さて、達成率で軸を切るといろいろと見えてくるようなので、ここを基準にしてピボットテーブルを使いながらデータを眺めてみましょう。なお、ここから先はstateがLiveのデータを除き、終了したプロジェクトのみを対象にします。加えて、集計に金額が関連する場合、平均等を求める処理の都合上、

米ドルでのプロジェクトに絞っています。

筆者はExcelのカラースケール機能によるヒートマップを愛用しています。プリント・オン・デマンドの場合、紙面が白黒なので見づらい可能性があります、あらかじめご了承ください。

まずは縦軸に達成率を取り、件数、件数比率、平均Back金額、平均Back件数を出してみます。

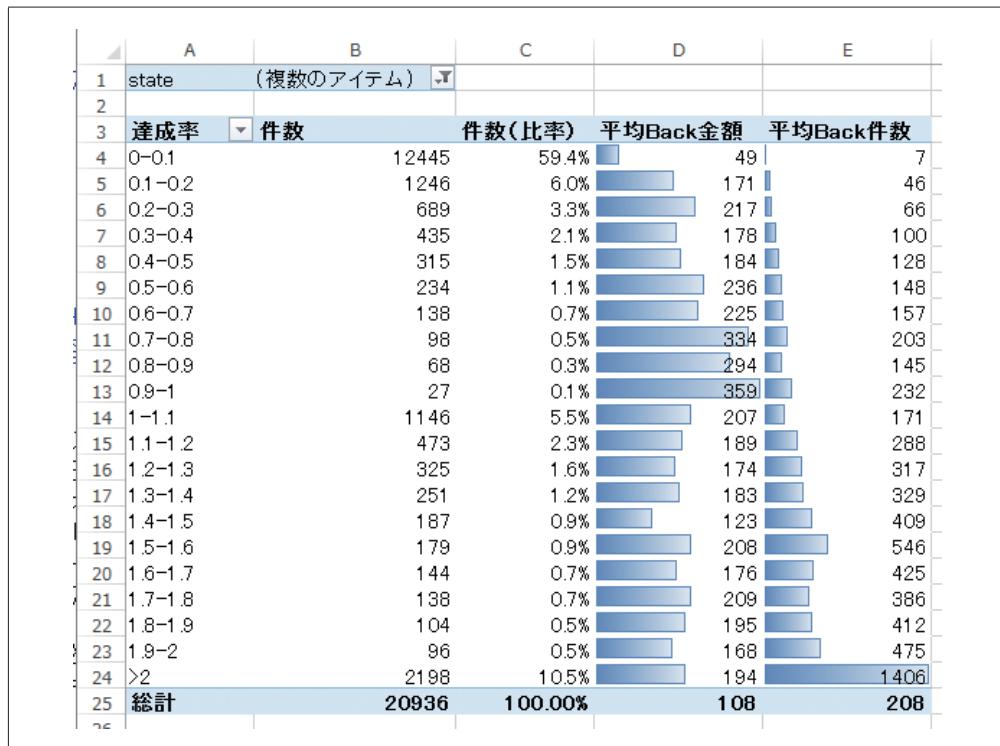


図8-5 縦軸に達成率を取って比較

客単価が高いと、ギリギリ失敗するケースが多そうです。平均Back金額が200ドル前後であれば成功確率は上がりそう…というわけで、平均Back金額の軸で比較してみましょう。

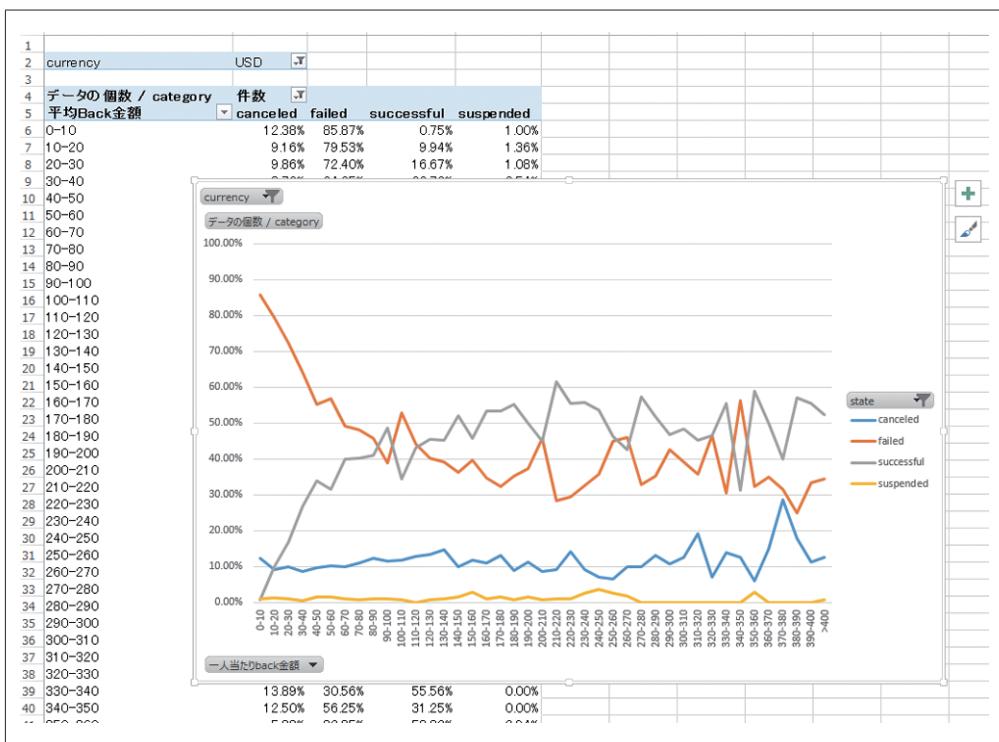


図8-6 平均Back金額で比較

平均Back金額の軸で比較してみたところ、平均Back金額が安いプロジェクトは失敗しやすいようです。しかし平均Back金額の軸では、平均Back金額が大きいからプロジェクトの成功率が下がるということはなさそうです。

次は達成率と年度の軸を比較してみましょう。年度ごとに集計することで、グラフの時系列変化を知ることができますし、年度による傾向の違いなども分かれます。

Excelのピボットテーブルで年度ごとに集計するには、日付型のフィールドを軸に設定した後でグループ化します。これにより年や月、日などのグループを作成できます。今回はこれらExcelの機能を活用して、年度等を変換しています。

A	B	C	D	E	F	G	H	I	J
1 state	(複数のアイテム)								
2 currency	USD								
3									
4 件数	列ラベル								
5 達成率	2009年	2010年	2011年	2012年	2013年	2014年	2015年	2016年	2017年
6 0~0.1		28	99	101	127	315	2167	3296	1884
7 0.1~0.2		5	16	16	38	85	225	254	212
8 0.2~0.3		1	4	13	20	49	134	162	95
9 0.3~0.4			4	6	15	33	96	91	71
10 0.4~0.5		1	3	3	14	24	47	69	37
11 0.5~0.6			1	4	8	15	32	52	39
12 0.6~0.7			2		3	10	26	30	20
13 0.7~0.8			2	1	1	5	18	26	14
14 0.8~0.9				1	1	4	9	11	15
15 0.9~1					1	1	6	4	8
16 1~1.1		9	18	22	48	95	186	257	201
17 1.1~1.2		2	11	17	24	39	77	102	78
18 1.2~1.3			4	9	18	36	56	70	45
19 1.3~1.4		1	2	8	11	35	41	48	39
20 1.4~1.5		1	4	5	9	23	24	33	30
21 1.5~1.6		1	3	2	11	19	28	39	30
22 1.6~1.7			3	8	9	15	23	28	21
23 1.7~1.8		1	1	3	9	18	24	30	20
24 1.8~1.9				1	7	11	23	23	14
25 1.9~2					2	5	15	17	15
26 >2		1	8	51	136	235	364	466	403
27 総計		51	185	274	514	1082	3623	5106	3291
28									
29 state	(複数のアイテム)								
30 currency	USD								
31									
32 件数	列ラベル								
33 達成率	2009年	2010年	2011年	2012年	2013年	2014年	2015年	2016年	2017年
34 0~0.1		55%	54%	37%	25%	29%	60%	65%	57%
35 0.1~0.2		9.8%	8.6%	5.8%	7.4%	7.9%	6.2%	5.0%	6.4%
36 0.2~0.3		2.0%	2.2%	4.7%	3.9%	4.5%	3.7%	3.2%	2.9%
37 0.3~0.4		0.0%	2.2%	2.2%	2.9%	3.0%	2.6%	1.8%	2.2%
38 0.4~0.5		2.0%	1.6%	1.1%	2.7%	2.2%	1.3%	1.4%	1.1%
39 0.5~0.6		0.0%	0.5%	1.5%	1.6%	1.4%	0.9%	1.0%	1.2%
40 0.6~0.7		0.0%	1.1%	0.0%	0.6%	0.9%	0.7%	0.6%	0.6%
41 0.7~0.8		0.0%	1.1%	0.4%	0.2%	0.5%	0.5%	0.5%	0.4%
42 0.8~0.9		0.0%	0.0%	0.4%	0.2%	0.4%	0.2%	0.2%	0.5%
43 0.9~1		0.0%	0.0%	0.4%	0.0%	0.1%	0.2%	0.1%	0.2%
44 1~1.1		17.6%	9.7%	8.0%	9.3%	8.8%	51%	50%	6.1%
45 1.1~1.2		3.9%	5.9%	6.2%	4.7%	3.6%	2.1%	2.0%	2.4%
46 1.2~1.3		0.0%	2.2%	3.3%	3.5%	3.3%	1.5%	1.4%	1.4%
47 1.3~1.4		2.0%	1.1%	2.9%	2.1%	3.2%	1.1%	0.9%	1.2%
48 1.4~1.5		2.0%	2.2%	1.8%	1.8%	2.1%	0.7%	0.6%	0.9%
49 1.5~1.6		2.0%	1.6%	0.7%	2.1%	1.8%	0.8%	0.8%	0.9%
50 1.6~1.7		0.0%	1.6%	2.9%	1.8%	1.4%	0.6%	0.5%	0.6%
51 1.7~1.8		2.0%	0.5%	1.1%	1.8%	1.7%	0.7%	0.6%	0.6%
52 1.8~1.9		0.0%	0.0%	0.4%	1.4%	1.0%	0.6%	0.5%	0.4%
53 1.9~2		0.0%	0.0%	0.7%	1.0%	1.4%	0.5%	0.3%	0.5%
54 >2		2.0%	4.3%	18.6%	26.5%	21.7%	10.0%	9.1%	12.2%
55 総計		100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
56									

図8-7 達成率と年度の軸を比較

どうやら、2011年以前、2011年から2013年、2014年以降で傾向が異なりそうです。件数と成功率が大きく変化しています。これはKickStarterの営業方針の変化などが原因だと推測できます。

また、達成率が50%から100%の案件が非常に少ないことが分かります。2014年以降、プロジェクトはぎりぎり達成か、200%以上の大成功かの二者択一という状況になってきています。

次はプロジェクトの目標金額と達成率を出してみましょう。プロジェクトの目標金額の分布は対数スケールなので、対数軸にして集計をとります。Excelのピボットテーブルでは対数分布の表に対して集計を適用できないため、手動で新しい項目を作ります。Excelで計算するなら $=10^{\text{int}(\log_{10}(\text{c2}))}$ のようなカラムを追加することで実現できます。当然のことですが、目標金額の低い方が目標到達率が良くなります。とはいっても特に変わった傾向があるわけではありません。

A	B	C	D	E	F	G	H	I
1	currency	USD						
2								
3	件数	列ラベル						
4	行ラベル	1	10	100	1,000	10,000	100,000	1,000,000
5	0~0.1	9.09%	32.56%	36.83%	51.46%	57.38%	70.52%	96.15%
6	0.1~0.2	0.00%	4.65%	5.33%	5.91%	6.57%	4.55%	0.00%
7	0.2~0.3	0.00%	6.98%	3.39%	3.89%	3.37%	2.41%	0.00%
8	0.3~0.4	0.00%	2.33%	2.75%	2.20%	2.37%	1.77%	0.96%
9	0.4~0.5	0.00%	0.00%	1.62%	1.51%	1.51%	0.86%	0.96%
10	0.5~0.6	0.00%	0.00%	1.94%	1.00%	1.07%	1.23%	0.00%
11	0.6~0.7	0.00%	0.00%	0.81%	0.56%	0.71%	0.59%	0.00%
12	0.7~0.8	0.00%	0.00%	0.16%	0.59%	0.55%	0.48%	0.00%
13	0.8~0.9	0.00%	0.00%	0.32%	0.46%	0.29%	0.21%	0.00%
14	0.9~1	0.00%	0.00%	0.00%	0.08%	0.20%	0.16%	0.00%
15	1~1.1	0.00%	2.33%	7.75%	8.06%	5.35%	3.37%	0.00%
16	1.1~1.2	0.00%	9.30%	2.75%	2.69%	2.44%	1.98%	0.00%
17	1.2~1.3	0.00%	2.33%	3.07%	1.41%	1.84%	1.12%	0.00%
18	1.3~1.4	0.00%	2.33%	1.29%	1.51%	1.31%	0.80%	0.00%
19	1.4~1.5	0.00%	2.33%	1.62%	1.18%	0.89%	0.54%	0.00%
20	1.5~1.6	0.00%	2.33%	1.29%	0.87%	0.94%	0.86%	0.96%
21	1.6~1.7	0.00%	0.00%	0.81%	1.02%	0.69%	0.54%	0.00%
22	1.7~1.8	0.00%	2.33%	0.97%	0.77%	0.72%	0.59%	0.00%
23	1.8~1.9	0.00%	0.00%	1.29%	0.67%	0.54%	0.37%	0.00%
24	1.9~2	0.00%	0.00%	1.62%	0.84%	0.30%	0.32%	0.00%
25	>2	90.91%	30.23%	24.39%	13.31%	10.96%	6.74%	0.96%
26	総計	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
27								
28	失敗率	9.09%	46.51%	53.15%	67.67%	74.03%	82.77%	98.08%
29	成功率	90.91%	53.49%	46.85%	32.33%	25.97%	17.23%	27.45%
30								

図8-8 目標金額と達成率

次は目標金額とプロジェクトのステータスを軸にして見てみましょう。

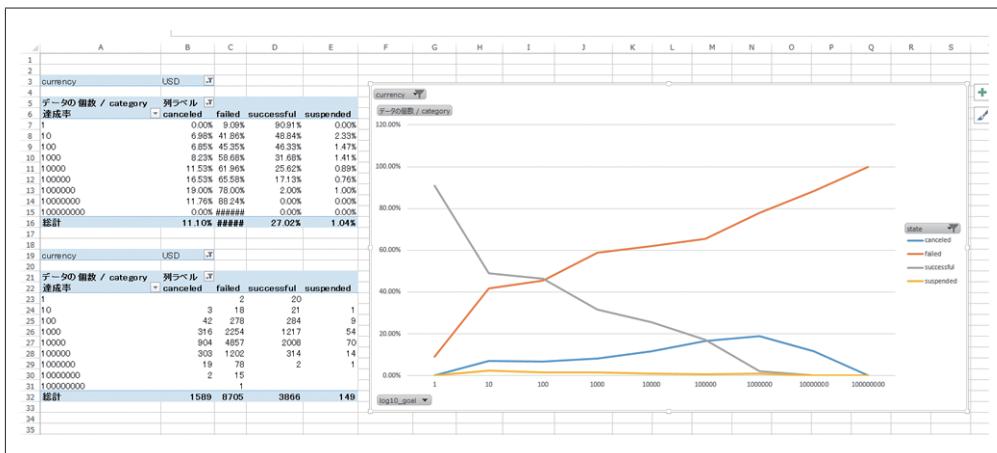


図8-9 目標金額とステータス

目標金額が上がるにつれて失敗率は上がっています。こちらのグラフのほうがキレイですね。データを見ると、目標金額に到達しているのにプロジェクトをキャンセルしている事例がちらほらあります。キャンセルしたデータが面白そうなので、詳しく調査してみましょう。

ひとまず、年度とキャンセル件数を可視化してみます。興味深いことに2014年以降、目標金額を達成したのにキャンセルされた事例が増えています。次は実際にキャンセルされたプロジェクトを見てみましょう。

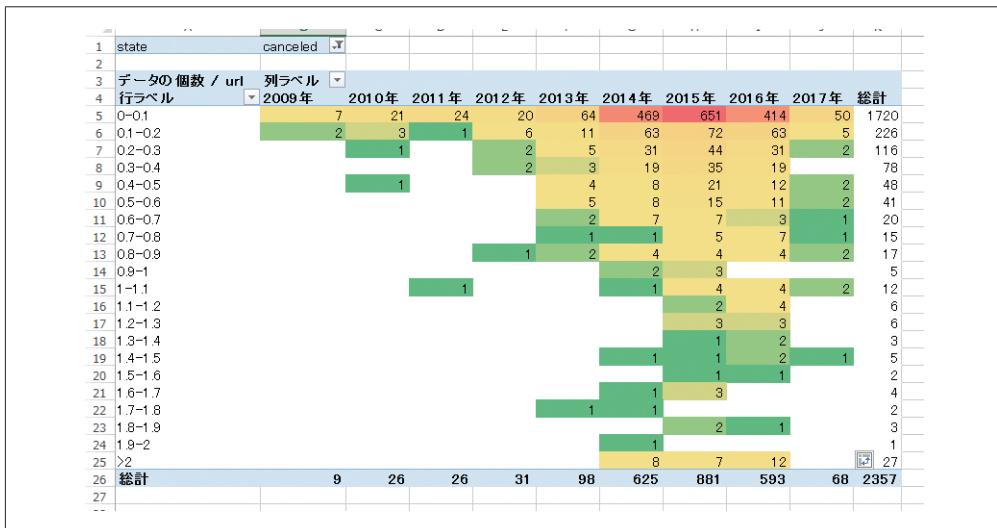


図8-10 キャンセルされたプロジェクト

8.6 達成したのにキャンセルされたプロジェクトを見てみる

AnyTouch Blue^{†4}は、USBポートに刺すと、スマホとBluetooth接続して、スマホが仮想キーボード、仮想マウスになるという製品です。目標金額20,000ドルに対して、3万ドル弱を集めたものの、プロジェクトをキャンセルして目標金額5,000ドルで再スタートしています(2017年3月現在)。

再スタートしたプロジェクト^{†5}では最低Back金額が18ドルから16ドルに下げられています。本稿の執筆時点で、目標金額5,000ドルに対して2万ドル強を集めており、プロジェクトの成功は確定しています。

この2つのプロジェクトを比較すると、一番分かりやすいのが、顧客単価です。まず、キャンセルされた方のプロジェクトは、 $29,763[\text{ドル}]/235[\text{Backer}] = 126[\text{ドル}/\text{Backer}]$ ですが、新たに立てた方のプロジェクトでは $21,189[\text{ドル}]/449[\text{Backer}] = 47[\text{ドル}/\text{Backer}]$ となります。

The screenshot shows the Kickstarter project page for 'AnyTouch Blue - Smart Keyboard & Mouse USB Dongle (Canceled)'.

- Creator:** Sung Mi Kim (2 created)
- Title:** AnyTouch Blue - Smart Keyboard & Mouse USB Dongle (Canceled)
- Description:** \$18 only. The AnyTouch Blue is a smart keyboard & mouse for wireless presentation, smart TV, smart STB, PC, MAC using a usb dongle.
- Thumbnail:** An image showing the white USB dongle and a smartphone connected to it, both displaying a presentation slide on a screen.
- Statistics:**
 - \$29,763 pledged of \$20,000 goal
 - 235 backers
- Status:** Funding Canceled. A note states: "Funding for this project was canceled by the project creator on February 20."
- Links:** Download on the App Store, GET IT ON Google Play.
- Location:** Auburn, AL

図8-11 キャンセルされたAnyTouch Blueのプロジェクトページ

†4 <https://www.kickstarter.com/projects/2094324441/anytouch-blue-smart-keyboard-and-mouse-usb-dongle/>

†5 <https://www.kickstarter.com/projects/2094324441/anytouch-blue-smart-keyboard-and-mouse-usb-dongler>

キャンセルされた方のプロジェクトでは、500個の商品が受け取れる9000ドルのプランに対して申し込んでいる人が2人もいます。新たに立てられたプロジェクトでは同様の8000ドルのプランに対して申し込んでいる人が1人に減っています。

おそらく、Kickstarterは8%程度の手数料^{†6}が差し引かれるため、大口顧客については裏で直接取引を持ち掛けていることが予想されます。そのため、二回目の募集では大口顧客がいなくなつたため、目標金額が5000ドルに引き下げられたのだと考えられます。

Kickstarterには、大きい金額のプランを作つておくことで、おそらくは卸業者であろう大口顧客とのパイプを作れるという特徴があるようです。加えて、対卸業者の場合、お互いに利益率を良くしたいという動機があるため、Kickstarterの手数料を回避する目的で、直接取引を持ち掛けている、ということが分かりました。

なお、このプロジェクトはindiegogoで資金を募り^{†7}、そちらで成功して商品出荷が完了したのちに、より大きい目標を立てて Kickstarterで募集をかけています。indiegogoで出資を募った際は、目標金額500ドルに対して1,705ドルを獲得しています。Kickstarterで募集をしたときにはindiegogoで成功したことを完全に隠しており、クラウドファンディングが一種の商流と化していることが伺えます。

今回は大口顧客がいたためにプロジェクトを再スタートしていますが、達成率100%付近の特異点を利用したプロジェクトの再スタートも考えられます。たとえば、一回目に低い目標金額で募集を開始し、どの程度 Backer が集まるかを計測し、「想定以上の Backer が集まつたので量産効果により値下げしました」と、目標金額を引き上げてリスタートすることで、目標金額付近での特異点を利用したギリギリでの集金効果をうまく利用することができるようになり、より大きな支援を得られると考えられます。

^{†6} <https://www.kickstarter.com/help/fees?country=US>

^{†7} <https://www.indiegogo.com/projects/anytouch-blue-android-bluetooth#/>

8.7 国別に見てみる

データの個数 / url	列ラベル	canceled	failed	live	successful	suspended	総計
行ラベル							
US		1589	8705	276	3866	149	14585
GB		255	1324	35	423	26	2063
CA		142	813	33	237	16	1241
AU		88	561	20	124	19	812
DE		44	276	21	96	4	441
NL		39	291	4	79	6	419
FR		51	243	13	83	2	392
IT		36	241	12	31	2	322
ES		15	169	8	21	4	217
DK		16	102	4	22		144
NZ		12	90	4	27	3	136
SE		14	82	1	13		110
CH		12	66	3	22	1	104
IE		8	51	2	18	1	80
NO		11	58		7	1	77
AT		9	51	4	12		76
BE		7	51	2	5	2	67
HK		3	11	7	20	3	44
MX		5	17	14	7		43
SG		1	14	4	6		25
LU			4	1	1		6
総計		2357	13220	468	5120	239	21404

図8-12 国別でみたプロジェクトステータス(件数)

2	3 データの個数 / url	列ラベル	canceled	failed	live	successful	suspended	総計
4 行ラベル								
5 US			10.89%	59.68%	1.89%	26.51%	1.02%	100.00%
6 GB			12.36%	64.18%	1.70%	20.50%	1.26%	100.00%
7 CA			11.44%	65.51%	2.66%	19.10%	1.29%	100.00%
8 AU			10.84%	69.09%	2.46%	15.27%	2.34%	100.00%
9 DE			9.98%	62.59%	4.76%	21.77%	0.91%	100.00%
10 NL			9.31%	69.45%	0.95%	18.85%	1.43%	100.00%
11 FR			13.01%	61.99%	3.32%	21.17%	0.51%	100.00%
12 IT			11.18%	74.84%	3.73%	9.63%	0.62%	100.00%
13 ES			6.91%	77.88%	3.69%	9.68%	1.84%	100.00%
14 DK			11.11%	70.83%	2.78%	15.28%	0.00%	100.00%
15 NZ			8.82%	66.18%	2.94%	19.85%	2.21%	100.00%
16 SE			12.73%	74.55%	0.91%	11.82%	0.00%	100.00%
17 CH			11.54%	63.46%	2.88%	21.15%	0.96%	100.00%
18 IE			10.00%	63.75%	2.50%	22.50%	1.25%	100.00%
19 NO			14.29%	75.32%	0.00%	9.09%	1.30%	100.00%
20 AT			11.84%	67.11%	5.26%	15.79%	0.00%	100.00%
21 BE			10.45%	76.12%	2.99%	7.46%	2.99%	100.00%
22 HK			6.82%	25.00%	15.91%	45.45%	6.82%	100.00%
23 MX			11.63%	39.53%	32.56%	16.28%	0.00%	100.00%
24 SG			4.00%	56.00%	16.00%	24.00%	0.00%	100.00%
25 LU			0.00%	66.67%	16.67%	16.67%	0.00%	100.00%
26 総計			11.01%	61.76%	2.19%	23.92%	1.12%	100.00%

図8-13 国別でみたプロジェクトステータスの(比率)

多くのプロジェクトが出されている国ほど成功しやすい。英語圏ほど成功しやすい。という傾向がありそうです。これは、Kickstarterが英語のサイトなので仕方ない側面があります。逆に言うと、各々の言語圏にそれぞれのクラウドファンディングサイトがある、というのが正しいのかもしれません。

次は国別のプロジェクト件数を年度を軸にして見てみます。

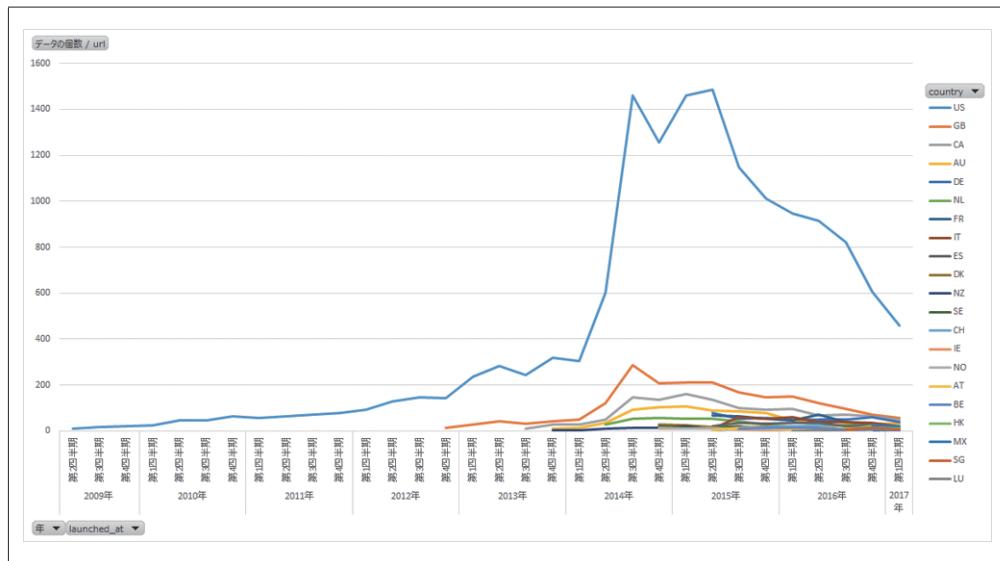


図8-14 国別のプロジェクト件数

2015年から国際化を始め、まずは英語圏、次にヨーロッパに進出しているようです。また、2015年の第3四半期から、プロジェクト数が減少しているのが分かります。これは他のクラウドファンディングサイトが立ち上がった影響でしょうか？とりあえず、Kickstarterが（Technologyカテゴリに限っては）ダウントレンドというのは面白い事実です。



実際には他のクラウドファンディングサービスのデータを眺めてみないと何とも言えません。他のクラウドファンディングサービスに顧客が流れた、他のカテゴリにプロジェクトを立てる人が増えた、等の理由が考えられます。

8.8 レポートを作る

さて、一通りのデータが集まつたので、ここからレポートを作つてみましょう。

今回は「KickStaterの分析を上司から依頼されたので、とりあえずデータをかき集めて分析して上

司に説明する」という場面を想定しています。また一部の図表は資料作成中に作成しています。

また、機械学習を利用したデータマイニングについては、このレポートの分析を元にして、今後どうするかを考えるという想定です。

Kickstarterの統計分析

中山ところてん

データ収集

- KickstarterのAPIを利用して収集したデータ
 - Technologyカテゴリのプロジェクト、21404件
 - データ収集日、2017/03/04
- データ収集方法
 - Kickstarterの非公開APIを利用して、jsonデータを収集
 - 通常検索
 - <https://www.kickstarter.com/projects/search?term=3d+printer>
 - 非公開API
 - <https://www.kickstarter.com/projects/search.json?term=3d+printer>
 - ソースコード
 - <https://github.com/oreilly-japan/ml-at-work/tree/master/chap08>
 - 備考、APIは4000件までしか結果を返さないので、Technologyカテゴリの下にあるサブカテゴリを指定して検索件数を抑制し、新しいプロジェクト順でデータ収集を実施
 - データ欠損の可能性があり
 - 次のクエリでTechnologyタグ全体を検索した際のhit数は27000件であり、データ欠損の可能性がある
 - https://www.kickstarter.com/projects/search.json?term=&category_id=16

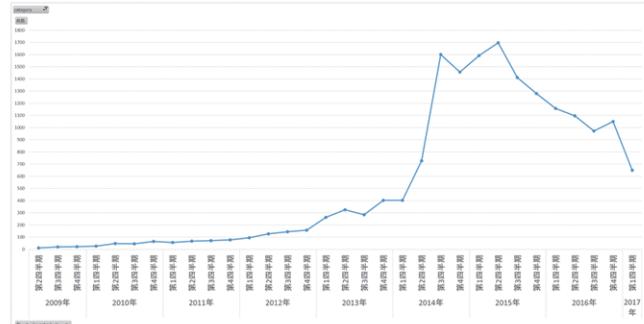
データ欠損の可能性

- カテゴリと年度のデータから、appsカテゴリがデータ欠損の可能性あり
- 検索オーダーがNewestであるため、時系列を伴わない統計分析については問題ないと判断する。以後、時系列を利用した分析を行う際にはappsカテゴリを除外して実施する

データの個数 / category	年度	年別									総計
		2009年	2010年	2011年	2012年	2013年	2014年	2015年	2016年	2017年	
technology		2	6	44	102	172	201	221	224	45	1017
technology/3d printing						35	148	243	142	12	580
technology/apps		2	3	5	5	968	1842	1066	121	4012	
technology/camera equipment			2	3	15	70	114	112	22	338	
technology/diy electronics		1	4	15	24	195	256	218	33	746	
technology/fabrication tools		1	5	2	2	45	77	64	11	207	
technology/flight			1	6	7	99	166	87	10	376	
technology/gadgets				6	10	480	977	773	126	2372	
technology/hardware		9	49	101	199	697	829	742	559	79	3264
technology/makerspaces				1	2	3	38	94	55	11	204
technology/robots		1	8	21	15	118	179	116	24	482	
technology/software		40	124	95	142	260	633	771	512	76	2653
technology/sound				5	11	11	103	188	192	33	543
technology/space exploration		1	5	15	10	65	94	72	12	274	
technology/wearables					6	217	350	359	49	981	
technology/web					7	944	1505	792	107	3355	
総計		51	185	274	529	1279	5153	7819	5343	771	21404

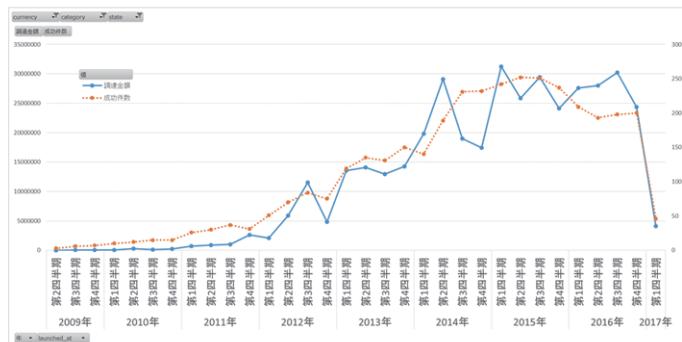
プロジェクト数のトレンド

- プロジェクト数は2015年第二四半期をピークに減少トレンドである
 - 他のクラウドファンディングサイトをクリエイターが使うようになった可能性
 - Kickstarter内の他のタグにプロジェクトが流れている可能性がある
 - 決済通貨をUSDに限定しても同様の傾向であるため、海外案件が増えたわけではない



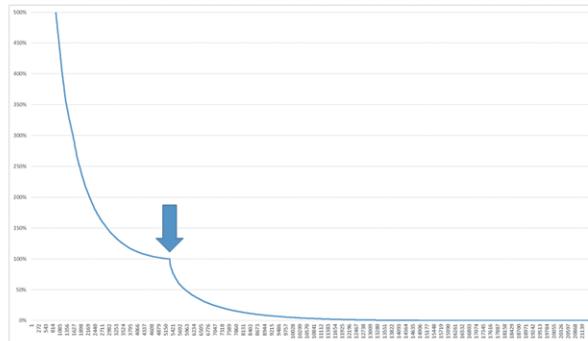
累計調達金額、成功プロジェクト数のトレンド

- 成功プロジェクト、調達金額(USDのプロジェクトに限定)は横ばい
 - Kickstarterが縮小傾向にあるとは言えない
 - 成功率の低いプロジェクトが2014年に多かつただけの模様



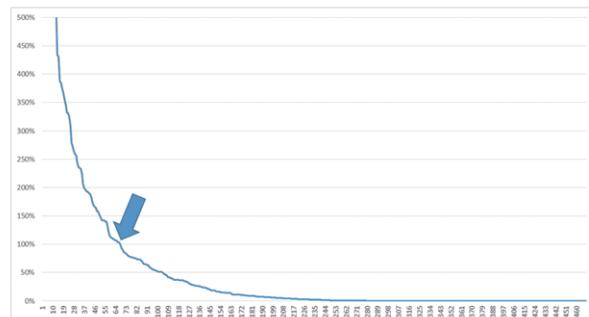
達成率に現れる特異点

- 達成率=調達金額/目標金額
 - 達成率順に並べて可視化を行う(縦軸:達成率)
 - 達成率100%近辺に特異点が発生している



特異点は終了したプロジェクトに現れる

- まだ募集中のプロジェクトに限定して達成率を算出
 - 100%付近での特異点は消える
 - プロジェクト終了間際に特異点が生まれると考えられる



年度ごとの達成率の分布

- フィルター条件
 - 現在募集中のプロジェクトは除外
 - 年度ごとの分析であるためappsカテゴリは除外
- 達成率の状況から、3つの異なる傾向が存在
 - 2010年まで、2011～2013年、2014年以降

category (件数のアーティクル)	2009年										2010年										2011年										2012年										2013年										2014年										2015年										2016年										2017年										総計																																																																																																																																																																																																																													
0~0.1	0.1~0.2	0.2~0.3	0.3~0.4	0.4~0.5	0.5~0.6	0.6~0.7	0.7~0.8	0.8~0.9	0.9~1.0	1~1.1	1.1~1.2	1.2~1.3	1.3~1.4	1.4~1.5	1.5~1.6	1.6~1.7	1.7~1.8	1.8~1.9	1.9~2.0	2~2.1	2.1~2.2	2.2~2.3	2.3~2.4	2.4~2.5	2.5~2.6	2.6~2.7	2.7~2.8	2.8~2.9	2.9~3.0	3~3.1	3.1~3.2	3.2~3.3	3.3~3.4	3.4~3.5	3.5~3.6	3.6~3.7	3.7~3.8	3.8~3.9	3.9~4.0	4~4.1	4.1~4.2	4.2~4.3	4.3~4.4	4.4~4.5	4.5~4.6	4.6~4.7	4.7~4.8	4.8~4.9	4.9~5.0	5~5.1	5.1~5.2	5.2~5.3	5.3~5.4	5.4~5.5	5.5~5.6	5.6~5.7	5.7~5.8	5.8~5.9	5.9~6.0	6~6.1	6.1~6.2	6.2~6.3	6.3~6.4	6.4~6.5	6.5~6.6	6.6~6.7	6.7~6.8	6.8~6.9	6.9~7.0	7~7.1	7.1~7.2	7.2~7.3	7.3~7.4	7.4~7.5	7.5~7.6	7.6~7.7	7.7~7.8	7.8~7.9	7.9~8.0	8~8.1	8.1~8.2	8.2~8.3	8.3~8.4	8.4~8.5	8.5~8.6	8.6~8.7	8.7~8.8	8.8~8.9	8.9~9.0	9~9.1	9.1~9.2	9.2~9.3	9.3~9.4	9.4~9.5	9.5~9.6	9.6~9.7	9.7~9.8	9.8~9.9	9.9~10.0	10~10.1	10.1~10.2	10.2~10.3	10.3~10.4	10.4~10.5	10.5~10.6	10.6~10.7	10.7~10.8	10.8~10.9	10.9~11.0	11~11.1	11.1~11.2	11.2~11.3	11.3~11.4	11.4~11.5	11.5~11.6	11.6~11.7	11.7~11.8	11.8~11.9	11.9~12.0	12~12.1	12.1~12.2	12.2~12.3	12.3~12.4	12.4~12.5	12.5~12.6	12.6~12.7	12.7~12.8	12.8~12.9	12.9~13.0	13~13.1	13.1~13.2	13.2~13.3	13.3~13.4	13.4~13.5	13.5~13.6	13.6~13.7	13.7~13.8	13.8~13.9	13.9~14.0	14~14.1	14.1~14.2	14.2~14.3	14.3~14.4	14.4~14.5	14.5~14.6	14.6~14.7	14.7~14.8	14.8~14.9	14.9~15.0	15~15.1	15.1~15.2	15.2~15.3	15.3~15.4	15.4~15.5	15.5~15.6	15.6~15.7	15.7~15.8	15.8~15.9	15.9~16.0	16~16.1	16.1~16.2	16.2~16.3	16.3~16.4	16.4~16.5	16.5~16.6	16.6~16.7	16.7~16.8	16.8~16.9	16.9~17.0	17~17.1	17.1~17.2	17.2~17.3	17.3~17.4	17.4~17.5	17.5~17.6	17.6~17.7	17.7~17.8	17.8~17.9	17.9~18.0	18~18.1	18.1~18.2	18.2~18.3	18.3~18.4	18.4~18.5	18.5~18.6	18.6~18.7	18.7~18.8	18.8~18.9	18.9~19.0	19~19.1	19.1~19.2	19.2~19.3	19.3~19.4	19.4~19.5	19.5~19.6	19.6~19.7	19.7~19.8	19.8~19.9	19.9~20.0	20~20.1	20.1~20.2	20.2~20.3	20.3~20.4	20.4~20.5	20.5~20.6	20.6~20.7	20.7~20.8	20.8~20.9	20.9~21.0	21~21.1	21.1~21.2	21.2~21.3	21.3~21.4	21.4~21.5	21.5~21.6	21.6~21.7	21.7~21.8	21.8~21.9	21.9~22.0	22~22.1	22.1~22.2	22.2~22.3	22.3~22.4	22.4~22.5	22.5~22.6	22.6~22.7	22.7~22.8	22.8~22.9	22.9~23.0	23~23.1	23.1~23.2	23.2~23.3	23.3~23.4	23.4~23.5	23.5~23.6	23.6~23.7	23.7~23.8	23.8~23.9	23.9~24.0	24~24.1	24.1~24.2	24.2~24.3	24.3~24.4	24.4~24.5	24.5~24.6	24.6~24.7	24.7~24.8	24.8~24.9	24.9~25.0	25~25.1	25.1~25.2	25.2~25.3	25.3~25.4	25.4~25.5	25.5~25.6	25.6~25.7	25.7~25.8	25.8~25.9	25.9~26.0	26~26.1	26.1~26.2	26.2~26.3	26.3~26.4	26.4~26.5	26.5~26.6	26.6~26.7	26.7~26.8	26.8~26.9	26.9~27.0	27~27.1	27.1~27.2	27.2~27.3	27.3~27.4	27.4~27.5	27.5~27.6	27.6~27.7	27.7~27.8	27.8~27.9	27.9~28.0	28~28.1	28.1~28.2	28.2~28.3	28.3~28.4	28.4~28.5	28.5~28.6	28.6~28.7	28.7~28.8	28.8~28.9	28.9~29.0	29~29.1	29.1~29.2	29.2~29.3	29.3~29.4	29.4~29.5	29.5~29.6	29.6~29.7	29.7~29.8	29.8~29.9	29.9~30.0	30~30.1	30.1~30.2	30.2~30.3	30.3~30.4	30.4~30.5	30.5~30.6	30.6~30.7	30.7~30.8	30.8~30.9	30.9~31.0	31~31.1	31.1~31.2	3

達成率の分布からわかること

- ・観測された事実
 - ・達成率10%未満で終わるプロジェクトが54%存在する
 - ・達成率50%から100%で終わるプロジェクトは3%程度と少ない
 - ・達成率100%～110%で終わるプロジェクトは5.7%と非常に多い
 - ・達成率が200%超で終わるプロジェクトは12%以上ある
 - ・達成率の変曲点は、募集中のプロジェクトには存在しない
 - ・年度ごとに見ても、達成率に変曲点
 - ・達成率に応じて、プロジェクトは3種類に大別することができる
 - ・達成率50%未満 : 典型的な失敗プロジェクト
 - ・達成率50%～200% : 終了間際にBackされるプロジェクト
 - ・達成率200%超 : 大成功プロジェクト

典型的な失敗プロジェクト

- ・達成率10%未満で終わるプロジェクトが54%存在する
 - ・プロジェクトの16.2%はBackerが0人、52%が10人以下
 - ・身内のご祝儀Backもできないような準備不足プロジェクトが多い
 - ・プロジェクトを成功させたいのであれば、Backer100人をどうやって集めるかを考える必要がある

終了間際にBackされるプロジェクト

- ・プロジェクト終了間際の広報活動によるBackerの増加
 - ・プロジェクトメンバーが最後のお願いをして回る
 - ・Kickstarterが終了間際のプロジェクトをトップページで紹介している
 - ・ただし、これは達成率100%付近に変曲点が生まれる理由にはならない
- ・最後のひと押しを積極的に行っている人がいる可能性
 - ・プロジェクトメンバーが自腹を切ってプロジェクトを成功させる
 - ・「俺が成功させた」感を味わいたい人
 - ・プロジェクトを達成させた、という体感を楽しみたい人
 - ・キャスティングポートを握るという稀有な体験ができる
 - ・クラウドファンディングをゲームとして楽しんでいる人
 - ・クラウドファンディングを「成功するかどうか分からぬゲーム」として考えた場合、勝率の高いゲームである、達成率90%付近のプロジェクトにBackするのは合理的
 - ・自分がBackしたプロジェクトが失敗するのは嫌なので、達成率が低いプロジェクトにはBackしない
 - ・達成率が100%を超えて成功確実の場合、ゲームにならないため、Backしない

ギリギリで失敗したプロジェクトの分析

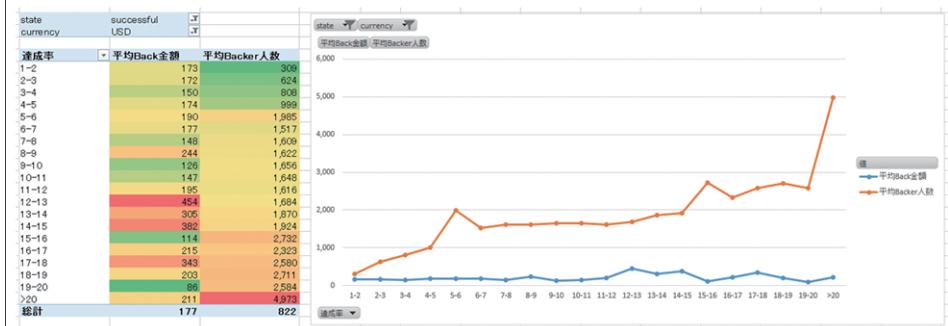
- ・達成率が70%から100%のプロジェクトは平均Back金額が250ドル超
- ・ギリギリで達成したプロジェクトは平均Back金額が200ドル未満
- ・Backプランの設計が成否を分けると考えられる

達成率	件数	性別(比率)		
		性別(比率)	平均Back金額	平均Backer
0-0.1	8118	56.7%	48	8
0.1-0.2	858	6.0%	135	47
0.2-0.3	482	3.3%	181	64
0.3-0.4	318	2.2%	159	110
0.4-0.5	202	1.4%	160	140
0.5-0.6	155	1.0%	193	175
0.6-0.7	92	0.6%	182	169
0.7-0.8	68	0.4%	279	200
0.8-0.9	45	0.3%	254	144
0.9-1	20	0.1%	404	241
1-1.1	849	5.9%	192	175
1.1-1.2	357	2.4%	165	287
1.2-1.3	242	1.6%	169	334
1.3-1.4	188	1.3%	149	363
1.4-1.5	133	0.9%	115	422
1.5-1.6	133	0.9%	178	652
1.6-1.7	108	0.7%	166	480
1.7-1.8	106	0.7%	197	412
1.8-1.9	82	0.5%	179	406
1.9-2	69	0.4%	142	406
>2	1686	11.7%	183	1514
総計	14309	100.0%	102	246

平均Back金額をそろえる
ために、決済通貨がUSDの
プロジェクトに限定

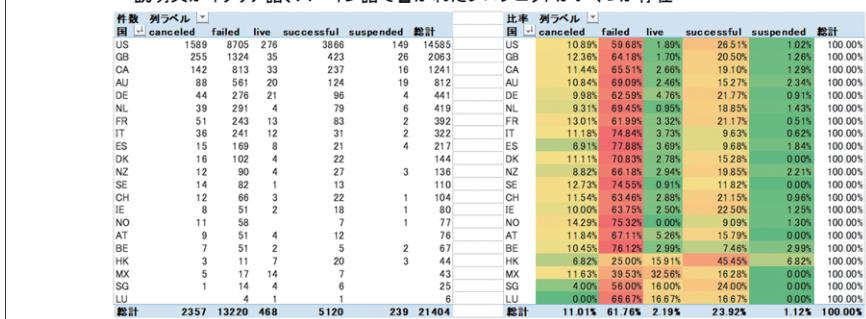
大成功プロジェクト

- 平均Back金額は大きく変動しない
- 達成率は基本的に平均Backer人数に依存する
- 高額なBackプランを払う人が多いから成功するというわけではない



国とプロジェクトの成功

- 英語が成功の力ぎを握る
 - アメリカが他国と比べて成功率が高い(件数の少ない香港を除く)
 - 英語圏と、ロマンス語の中でも英語に近いフランス語、英語がほぼ必須な小国は成功率が高い
- イタリア、スペインのロマンス語圏は失敗率が高い
 - 説明文がイタリア語、スペイン語で書かれたプロジェクトがいくつか存在



目標金額と成功率

- 基本的には目標金額が小さいほど成功確率は増大する

state currency	(複数のアイテム)											合計
件数	ゴール金額											
達成率	1	10	100	1,000	10,000	100,000	1,000,000	10,000,000	100,000,000	1,000,000,000	合計	
0-0.1	9.09%	32.56%	36.70%	51.29%	57.44%	70.38%	96.00%	100.00%	100.00%	100.00%	56.73%	
0.1-0.2	0.00%	4.65%	5.38%	5.96%	6.52%	4.53%	0.00%	0.00%	0.00%	0.00%	6.00%	
0.2-0.3	0.00%	6.98%	3.43%	3.93%	3.34%	2.45%	0.00%	0.00%	0.00%	0.00%	3.37%	
0.3-0.4	0.00%	2.33%	2.61%	2.21%	2.33%	1.75%	1.00%	0.00%	0.00%	0.00%	2.22%	
0.4-0.5	0.00%	0.00%	1.63%	1.48%	1.51%	0.87%	1.00%	0.00%	0.00%	0.00%	1.41%	
0.5-0.6	0.00%	0.00%	1.79%	1.02%	1.03%	1.20%	0.00%	0.00%	0.00%	0.00%	1.07%	
0.6-0.7	0.00%	0.00%	0.82%	0.55%	0.71%	0.55%	0.00%	0.00%	0.00%	0.00%	0.64%	
0.7-0.8	0.00%	0.00%	0.16%	0.52%	0.48%	0.49%	0.00%	0.00%	0.00%	0.00%	0.48%	
0.8-0.9	0.00%	0.00%	0.33%	0.44%	0.28%	0.22%	0.00%	0.00%	0.00%	0.00%	0.31%	
0.9-1	0.00%	0.00%	0.08%	0.18%	0.16%	0.00%	0.00%	0.00%	0.00%	0.00%	0.14%	
1-1.1	0.00%	2.33%	7.83%	8.12%	5.42%	3.44%	0.00%	0.00%	0.00%	0.00%	5.93%	
1.1-1.2	0.00%	9.30%	2.77%	2.73%	2.47%	2.02%	0.00%	0.00%	0.00%	0.00%	2.49%	
1.2-1.3	0.00%	2.33%	3.10%	1.43%	1.86%	1.15%	0.00%	0.00%	0.00%	0.00%	1.69%	
1.3-1.4	0.00%	2.33%	1.31%	1.54%	1.34%	0.92%	0.00%	0.00%	0.00%	0.00%	1.31%	
1.4-1.5	0.00%	2.33%	1.63%	1.15%	0.97%	0.55%	0.00%	0.00%	0.00%	0.00%	0.93%	
1.5-1.6	0.00%	2.33%	1.31%	0.86%	0.94%	0.87%	1.00%	0.00%	0.00%	0.00%	0.93%	
1.6-1.7	0.00%	0.00%	0.89%	1.02%	0.69%	0.55%	0.00%	0.00%	0.00%	0.00%	0.75%	
1.7-1.8	0.00%	2.33%	0.98%	0.78%	0.74%	0.60%	0.00%	0.00%	0.00%	0.00%	0.74%	
1.8-1.9	0.00%	0.00%	1.31%	0.68%	0.52%	0.38%	0.00%	0.00%	0.00%	0.00%	0.57%	
1.9-2	0.00%	0.00%	1.63%	0.78%	0.29%	0.33%	0.00%	0.00%	0.00%	0.00%	0.48%	
>2	99.91%	30.23%	24.47%	13.43%	11.01%	6.71%	1.00%	0.00%	0.00%	0.00%	11.78%	
総計	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

達成率とプロジェクトキャンセル

- 達成率が100%を超えていてもかかわらず、自らキャンセルしているプロジェクトがいくつか存在する

state category	2009年	2010年	2011年	2012年	2013年	2014年	2015年	2016年	2017年	合計	
データの個数 / category											
行ラベル	2009年	2010年	2011年	2012年	2013年	2014年	2015年	2016年	2017年	合計	
0-0.1	7	21	24	20	64	469	651	414	50	1720	
0.1-0.2	2	3	1	6	11	63	72	63	5	226	
0.2-0.3	1		2	5	31	44	31	2	116		
0.3-0.4	1		2	3	19	35	19		78		
0.4-0.5		4	8	21	12	2	2		48		
0.5-0.6		5	8	15	11	2	2		41		
0.6-0.7		2	7	7	3	1	1		20		
0.7-0.8	1	1	5	7	7	7	1		15		
0.8-0.9	1	2	4	4	4	4	2		17		
0.9-1		2	3						5		
1-1.1	1		1	4	4	4	2		12		
1.1-1.2			2	4	4	4	6		6		
1.2-1.3			3	3	3	3	6		6		
1.3-1.4			1	2	2	1	1		3		
1.4-1.5		1	1	2	1	1	1		5		
1.5-1.6		1	1	1	1	1	1		2		
1.6-1.7		1	3				4		4		
1.7-1.8	1	1	2	1	1	1	1		2		
1.8-1.9		8	7	12			3		3		
1.9-2									1		
>2									27		
総計	9	26	26	31	98	625	881	593	68	2357	

達成後キャンセル事例

- ・工場に発注してみたら、予算オーバーした
 - ・ロボット系、ドローン系
- ・価格改定のためにプロジェクトを立て直す
 - ・想定以上の注文により、量産効果により価格を下げられることが判明
 - ・Weather Point 2.0 スマホのイヤホンジャックに付ける気象センサー
 - ・AnyTouch Blue USBドングルをPCIに刺すと、スマホが仮想キーボード、マウスになる
 - ・このプロジェクトはindiegogoで成功した後にKickstarterに出品するという形で、クラウドファンディングを流通網の一つとして利用しており、大変興味深い
- ・事故でプロジェクトの継続が一時的に困難になった
 - ・Dotlens smartphone microscope スマホのカメラに付けるレンズ
 - ・テキサス州の洪水にあり、製造設備と在庫を喪失
 - ・その後プロジェクトを立て直し、無事成功

まとめ

- ・Kickstarterは衰退はしていないものの横ばい
 - ・他のクラウドファンディングサービスや、他国のローカルのクラウドファンディングサービスに顧客を奪われている可能性がある
 - ・Technologyカテゴリに限定しているので、他のカテゴリ次第では成長している可能性
- ・クラウドファンディングの効果を、達成率100%付近の歪みとして可視化
 - ・最後のひと押しをしてくれる人がいる
 - ・プロジェクト関係者が身銭を切っている可能性
 - ・「俺が育てた感」を味わうのが好きな人がいる
 - ・最後のひと押しをするには、安いBackプランが必要
- ・大成功プロジェクトは、基本的には大多数から支持されるもの
 - ・Kickstarterが販売チャネルになっているようなケースが多い

Kickstarterで成功させるコツ

- 英語でコンテンツを用意する
 - 追加情報を探しに行っても母国語でしか出てこないケースではBackしにくい
- 目標金額を小さくし、確実に成功できるようにする
 - プロジェクトが成功したことそのものをニュースにすることができ、広報活動につながる
 - 大成功したら、プロジェクトをいったんキャンセルして、立て直す裏ワザもある
- プロジェクトを始める前に、最低でも10人はBackerを確保する
 - 盛り上がってる感を正しく演出する
 - 目標金額の30%を超えると、「成功しそうだから」という理由でBackする人が増え始める可能性が高い
- 少額でリワードが得られるBackプランを用意しておく
 - 10ドル程度で何かがもらえる少額のBackプランを容易しておく
 - 平均Back金額が200ドル未満になるようにする
 - 最後のひと押しのための応援は、少額であれば気軽に使うことができる

8.9 今後行いたいこと

さて、資料はどうでしたか？紙面の都合上、今回それほど複雑な分析は行えませんでした。きちんと分析する時間があるのなら、次のような分析を行うと良いでしょう。

- プロジェクト紹介の説明文の文章と成功率の関係
- プロジェクト紹介の説明文の見出しと成功率の関係（紹介文がどのような構造になっていると通りやすいのか）
- チーム紹介の有無と成功率
 - チーム紹介はプロジェクト紹介を見ないと分からないので、自然言語処理が必要
- チーム人数と、成功率
 - チーム人数は、チーム紹介の中を解析しないと分からないので、自然言語処理が必要
- クリエイターのBack経験と成功率
 - ユーザとして Kickstarter を活用している人ほど、Kickstarter の文化を理解しており、どのようなプロジェクトが成功する可能性が高いのかを理解している
- プロジェクトの募集期間と成功率の関係
- プロジェクト開始日、終了日の曜日と成功率の関係

ソーシャルゲームの分析との関連

今回の分析は「達成率」という指標を導入することで、様々な目標金額、到達状況にあるプロジェクトを横断的に分析しました。実はこの手法はソーシャルゲームのデータ分析手法を応用したものなのです。例えば次のようなイベントがあると考えてみます。

- イベントポイントを貯めるとレアカードがもらえる
- イベントポイントが10位以上で超レアなカードが4枚もらえる
- イベントポイントが100位以上で超レアなカードが3枚もらえる
- イベントポイントが1000位以上で超レアなカードが2枚もらえる
- イベントポイントが5000位以上で超レアなカードが1枚もらえる

このようなイベントでは、ランキングと報酬が非連続に変化するため、非連続箇所においてイベントポイントの分布が歪むことになります。

自分が今5,005位にいると分かっているユーザは、ほんの少しの努力をするだけで自分が5000位以内に入れることが確実であるため、5000位に入るよう努力します。これによって5000位だったユーザは5001位に転落するため、彼も再び5000位以内に入れるようにゲームを遊ぶこととなります。

したがって、報酬がもらえる境界付近のユーザの競争が過熱化し、イベントポイントをより多くためるために、ゲームに課金するという行動が発生しやすくなります。ランキングシステムを採用しているソーシャルゲームの売上は、こういった報酬の境界線の決定などにより大きく変化します。

また、この分析手法では順位ごとの課金額を計算することで、レアカードの価値を算定できます。つまり、10位の人の課金額=超レアカード4枚の価値、5000位の人の課金額=超レアカード1枚分の価値ということができます（実際には50位刻み程度で課金額の平均値をとり、ノイズを低減させる）。これはある意味、ユーザを利用してレアカードをオークションさせていると言えます。

これにより、レアカード1枚当たりの価値を常に計測できます。ソーシャルゲームはインフレしていく宿命にあるため、カードの価値というものは常に下がり続けます。そのため、このようなデータを計測することで、レアカード1枚当たりの価値が一定よりも下がってきたら、より強いカードをリリースする、という運用が可能になります。

8.10 おわりに

今回の分析では、達成率という指標を用いて、Kickstarterのプロジェクトを分析しました。

Kickstarterのように、特定の場所に非連続な報酬があるタイプの問題では、非連続な報酬が与えられる箇所に着目して分析することが有効です。世の中はこのような歪みに満ち溢れているので、何かの歪みを見つけたら、どこで非連続になっているのか、非連続になっている箇所付近ではどのような現象が起こっているのかに着目すると面白いでしょう。

Kickstarterのように報酬の非連続性がトータルとしてプラスに働く場合もあれば、年収103万円の壁^{†8}、130万円^{†9}の壁などのマイナスに働く場合もあります。このような非連続性のある個所を分析することは、ビジネスにおいて強い武器になるので、覚えておいて損はないと思います。

†8 配偶者控除により、所得税を払わなくて済み、かつ扶養家族でいられる年収。これを超えると所得税の支払いが発生し、扶養家族の税制優遇がなくなります。2018年から150万円に改正されるようです。

†9 親族の健康保険の扶養から外れ、自ら健康保険に入る必要が生じます。

9章

Uplift Modelingによるマーケティング資源の効率化

本章ではUplift Modelingの紹介と実装を行います。

Uplift Modelingとは、疫学統計やダイレクトマーケティングにおいて活用される機械学習の手法です。この手法は、ランダム化比較試験 (Randomized Controlled Trial) のデータを分析することにより、どのような患者に対して薬が作用するのか、どのような顧客に対してダイレクトメールを送付すると成績に結びつくのか、といったことを予測できます。



Uplift ModelingのUpliftとは「持ち上げる」という意味です。

ランダム化比較試験とは、いわゆるA/Bテストのことです。母集団をランダムに実験群 (Treatment Group) と統制群 (Control Group) の2つに分け、実験群には試験したい介入行為を行い、統制群には何しません。たとえば新薬開発であれば、実験群には介入行為として新薬を投与し、統制群には偽薬を投与します。ウェブサービスであれば、実験群には介入行為として新しいバナー広告を表示し、統制群には従来のバナー広告を表示します。

Uplift Modelingが普通のA/Bテストと異なるのは、単に反応したかどうかを調べるのではなく、実験群と統制群においてどのような特徴量を持つ標本が反応したのか、あるいは反応しなかったのかを調べることで、ある標本が介入行為に対してどのように反応するかを予測します。これにより、効果が出ると見込める対象にのみ介入行為を行えるようになります。逆に、介入が逆効果になると予測される対象に対しては介入を控えることができます。

たとえば医療を例に考えます。患者の年齢性別、遺伝子や生活習慣を特徴量として学習を行うことで、薬が効くと分かっている患者にだけ投薬を行い、副作用が出ると分かっている患者や、自然に治ると分かっている患者には投薬を控えるといったことが可能になります。これにより、医療のパーソナライゼーションだけでなく、医療資源の効率的な利用につながります。

Uplift Modelingの詳しい解説や活用事例については、『ヤバい予測学』[ヤバい予測学]の第七章 数字による説得、および同書の参考文献を参照してください。

9.1 Uplift Modelingの四象限のセグメント

Uplift Modelingでは、介入行為が無かったらどのような行動をとるのか、介入があったらどのような行動をとるのかを軸として、対象を4つのセグメント（層）に分けて考えます。どのような行動をとるのかについては、便宜的にコンバージョン（Conversion、CV）する、コンバージョンしないの2値として考えます。

表9-1 Uplift Modelingの四象限

介入行為なし	介入行為あり	カテゴリ	るべきアクション
CVしない	CVしない	無関心	介入行為にコストがかかる場合、介入を控える
CVしない	CVする	説得可能	できる限りこここのセグメントに介入する
CVする	CVしない	天邪鬼	絶対に介入してはいけない
CVする	CVする	鉄板	介入行為にコストがかかる場合、介入を控える

無関心は、介入行為をしてもしなくてもコンバージョンしないセグメントです。例えば、過去3年間購買記録のない顧客にダイレクトメールを送っても、購入に結びつかない可能性が高いと推測されます。そこで、このような顧客には休眠顧客としてのフラグを付与して、ダイレクトメールの送付をやめることでコストを削減できます。

説得可能は、介入行為があつてはじめてコンバージョンに転じるセグメントであり、Uplift Modelingによって最も発見したいセグメントです。例えば「ウェブサイトを何度も訪問しており、他社のウェブサイトと価格比較をしていて、価格で迷っている」というような顧客に対しては、割引クーポンを送付して、購買に転換させることができるでしょう。

天邪鬼は、何もしなければコンバージョンするが、介入行為を行うとコンバージョンしなくなるセグメントです。例えば、衣料品店で声を掛けられると居づらくなってしまうお客様や、借金の返済状況について確認の連絡をすると早期返済してしまうお客様^{†1}などがこれに該当します。このようなセグメントに対しては、介入行為を行うと売上が下がってしまうため、介入行為を行わないようにします。

鉄板は、介入行為を行っても行わなくても、どちらにせよコンバージョンしてしまうセグメントです。例えば、スーパーのレジ前に並んでいるお客様割引クーポンを渡したところで、レジ前に並んでいるお客様はコンバージョンすることが確実であるため、売り上げは増えません。それどころか、クーポンによる割引によって売り上げは減少してしまいます。介入行為による反応率という点で非常に良い反

†1 銀行の収益源は利息であり、借入金の早期返済は収益の減少につながります。

応が得られるため、反応率をKPIにしている場合、往々にして実施対象にしてしまいますが、クーポンや広告提示のように介入行為にコストが伴う場合、介入は控えるべきです^{†2}。

9.2 A/B テストの拡張を通じた Uplift Modeling の概要

では、バナー広告のA/Bテストの事例を元にして、A/Bテストをどのように拡張し Uplift Modeling を実現するのかを考えましょう。

まず、ウェブサイト内におけるバナー広告のA/Bテストについて考えます。バナー広告AとBの反応率が、それぞれ4.0%と5.0%であったとします。普通のA/Bテストであれば、バナー広告Bを全員に出すのが良いという判断になります。

表9-2 A/Bテストの結果

表示内容	バナー広告A	バナー広告B
反応率	4.0%	5.0%

しかし、Uplift Modellingでは、個々の顧客が持つ特徴量を活用します。ここでは顧客の性別の軸でA/Bテストの結果を拡張してみましょう。この例では話を単純にするため、男女比率は1:1であると仮定します。

表9-3 性別の軸でA/Bテストを拡張する

反応率	バナー広告A	バナー広告B
男性	6.0%	2.0%
女性	2.0%	8.0%
平均	4.0%	5.0%

性別の軸でデータを眺めると、男性にはバナー広告A、女性にはバナー広告Bを出すことが有効であると分かりました。男性にはバナー広告A、女性にはバナー広告Bを出すことになると、平均で7.0%の反応率が期待でき、A/BテストによってBだけを出していった場合よりも、高い反応率を得ることができます。

分類する軸が性別のような名義尺度であれば、人力でも簡単に分析することができます。しかし、より多くの特徴量を元に、どちらのバナーを出せば良いかを判断するのは人の手では困難です。このようなときに機械学習を用いて解決します。

^{†2} 鉄板セグメントに対して広告出稿しないことも、マーケティング部門の仕事に含まれます。反応率がいいからといって、リターゲティング広告に頼ると、鉄板セグメントに対して広告を打つことになったりします。

9.3 Uplift Modelingのためのデータセット生成

Uplift Modelingには標準的な公開データセットが存在しないため、今回はデータセットの生成からはじめます。加えて、今回のように新しいアルゴリズムを実装する際、性質が未知のデータセットを用いてアルゴリズム開発を行うと、出力された結果がおかしかった場合、データセットの性質によるものなのか、アルゴリズムにバグがあるせいなのかが分からなくなつたのです。

Uplift Modelingでは、実験群と統制群の二種類の標本が必要です。今回は、標本サイズとランダムシードを与えることで、コンバージョンしたか否か(is_cv_list)、実験群か否か(is_treat_list)、8次元の特徴量(feature_vector_list)を返す関数を作成します。

この関数は、どの特徴量がどれくらいコンバージョンに影響を与えるかという重みを内部に持ち、各特徴量の値が、コンバージョンに影響を与えるようにしています。base_weightが統制群が持つ重み、lift_weightが介入により変化する重みです。

なお、このサンプルコードでは、lift_weightの合計値を0に設定しており、実験群と統制群のコンバージョンレートがほぼ同一になるように設定しています。つまり、介入行為により、あるセグメントの顧客はコンバージョンレートが改善したが、別セグメントの顧客のコンバージョンレートが悪化し、全体としては改善していないように見える、というシナリオです。

```
import random

def generate_sample_data(num, seed=1):
    # 返却するリストを確保
    is_cv_list = []
    is_treat_list = []
    feature_vector_list = []

    # 亂数を初期化
    random_instance = random.Random(seed)

    # 返す関数の特徴を設定
    feature_num = 8
    base_weight = \
        [0.02, 0.03, 0.05, -0.04, 0.00, 0.00, 0.00, 0.00]
    lift_weight = \
        [0.00, 0.00, 0.00, 0.05, -0.05, 0.00, 0.00, 0.00]

    for i in range(num):
        # 特徴ベクトルを乱数で生成
        feature_vector = \
            [random_instance.random()
             for n in range(feature_num)]
        # 実験群かどうかを乱数で決定
        is_treat = random_instance.choice((True, False))
        # 内部的なコンバージョンレートを算出
        cv_rate = \
```

```

        sum([feature_vector[n] * base_weight[n]
            for n in range(feature_num)])

    if is_treat:
        # 実験群であれば、lift_weight を加味する
        cv_rate += \
            sum([feature_vector[n] * lift_weight[n]
                for n in range(feature_num)])

    # 実際にコンバージョンしたかどうかを決定する
    is_cv = cv_rate > random_instance.random()

    # 生成した値を格納
    is_cv_list.append(is_cv)
    is_treat_list.append(is_treat)
    feature_vector_list.append(feature_vector)

# 値を返す
return is_cv_list, is_treat_list, feature_vector_list

```

この関数は、まず[0, 1]の乱数を8個もった特徴量 (feature_vector) を生成します。次に実験群か統制群であるか (is_treat) を乱数で決め、それぞれの場合について内部コンバージョンレート (cv_rate) を求めます。内部コンバージョンレートは、feature_vector と base_weight の内積で定義し、実験群であった場合 (is_treat == True)、feature_vector と lift_weight の内積を加算します。

式で表現すると次のようになります。なお「・」はベクトルの内積を表します。

$$cv_rate = \begin{cases} feature_vector \cdot base_weight & \dots(\text{統制群の場合}) \\ feature_vector \cdot (base_weight + lift_weight) & \dots(\text{実験群の場合}) \end{cases}$$

そして、cv_rateの値に基づいて、コンバージョンしたかどうか (is_cv) を決定します。例えば、cv_rateが0.3であればis_cvは30%の確率でTrueになります。これにより、feature_vector と is_cv、is_treatのみが観測でき、外部からbase_weight や lift_weightといった潜在的な変数が観測できないサンプルデータ生成器ができあがります。

また、base_weightには重みが0の変数が用意されており、これは、観測できているが、コンバージョンに寄与しない変数を意味しています。このような変数を用意することで、モデルの頑健性を評価できます。

この関数を実行すると、is_cv_list, is_treat_list, feature_vector_listのタプルが得られます。次節からは、この関数を使って、Uplift Modelingのアルゴリズムを作っていきます。

```
generate_sample_data(2)
```

```
([False, False],
 [True, False],
```

```
[[0.5692038748222122,
  0.8022650611681835,
  0.06310682188770933,
  0.11791870367106105,
  0.7609624449125756,
  0.47224524357611664,
  0.37961522332372777,
  0.20995480637147712],
 [0.43276706790505337,
  0.762280082457942,
  0.0021060533511106927,
  0.4453871940548014,
  0.7215400323407826,
  0.22876222127045265,
  0.9452706955539223,
  0.9014274576114836]])
```

9.4 2つの予測モデルを利用したUplift Modeling

原始的なUplift Modelingは、実験群と統制群でそれぞれで予測モデルを作ります。

ある特徴ベクトルを持った顧客について、それぞれの予測モデルでコンバージョンレートを予測します。統制群の予測モデルでは、介入行為を行わなかった場合のコンバージョンレートが予測されます。実験群の予測モデルでは、介入行為を行った場合のコンバージョンレートを予測します。したがって、統制群の予測モデルと、実験群の予測モデルを組み合わせることで、介入行為によるコンバージョンレートの変化を予測できるのです。

以下の表は、予測モデルの出力結果と、Uplift Modelingのセグメントを組み合わせたものです。

表9-4 予測モデルの出力と、Uplift Modelingのセグメント対応

統制群の予測モデルの結果	実験群の予測モデルの結果	対応するUplift Modelingのセグメント
低	低	無関心
低	高	説得可能
高	低	天邪鬼
高	高	鉄板

まずは学習用のサンプルデータを生成し、全体のコンバージョンレートを確認します。

```
# trainデータの生成
sample_num = 100000
train_is_cv_list, train_is_treat_list, train_feature_vector_list = \
    generate_sample_data(sample_num, seed=1)

# データをtreatmentとcontrolに分離
treat_is_cv_list = []
treat_feature_vector_list = []
control_is_cv_list = []
```

```

control_feature_vector_list = []

for i in range(sample_num):
    if train_is_treat_list[i]:
        treat_is_cv_list.append(train_is_cv_list[i])
        treat_feature_vector_list.append(
            train_feature_vector_list[i])
    else:
        control_is_cv_list.append(train_is_cv_list[i])
        control_feature_vector_list.append(
            train_feature_vector_list[i])

# コンバージョンレートを表示
print("treatment_cvr",
      treat_is_cv_list.count(True) / len(treat_is_cv_list))
print("control_cvr",
      control_is_cv_list.count(True) / len(control_is_cv_list))

treatment_cvr 0.0309636212163288
control_cvr 0.029544629532529343

```

若干、実験群のほうがコンバージョンレートが大きいのですが、3.10%と2.95%でほぼ差がありません。これがA/Bテストであれば、有意な差がないためこの実験は失敗であった、と判断することになります。

しかし今回はUplift Modelingです。顧客の持つ特微量とコンバージョンしたか否かの情報をもとに、どのようなセグメントが介入行為に反応したのかを特定し、最終的には介入行為により改善するセグメントにのみ介入行為を実施することを狙います。

次に、学習器を構築して、trainデータの学習を行います。今回はコンバージョン予測の問題であるため、このような問題でよく使われる、ロジスティック回帰によるクラス分類を利用します。

```

from sklearn.linear_model import LogisticRegression

# 学習器の生成
treat_model = LogisticRegression(C=0.01)
control_model = LogisticRegression(C=0.01)

# 学習器の構築
treat_model.fit(treat_feature_vector_list, treat_is_cv_list)
control_model.fit(control_feature_vector_list, control_is_cv_list)

```

続いて、Uplift Modelingのスコアを算出します。

2つの予測モデルを利用したUplift Modelingの場合、統制群の予測値と実験群の予測値の2つが得られます。このままだと扱いにくいため、一次元の値に変換します。説得可能な顧客と、天邪鬼な顧客はそれぞれ次のようになっています。

- 統制群の予測値が低く、実験群の予測値が高いとき、説得可能な顧客なので、高いスコアになっ

てほしい

- 統制群の予測値が高く、実験群の予測値が低いとき、天邪鬼な顧客なので、低いスコアになってほしい

したがって、予測値の比もしくは差を利用することで、説得可能な顧客は高いスコアに、天邪鬼な顧客は低いスコアに変換することができます。今回は予測値の比を利用します。

$$\text{UpliftModeling のスコア} = \frac{\text{実験群の予測値}}{\text{統制群の予測値}}$$

scikit-learnのクラス分類器は、`predict_proba`関数を持っており、特徴ベクトルを引数に与えると、`numpy.ndarray`型の配列でクラスの所属確率を得られます。今回は、クラスがTrueとFalseの2個であることが分かっているため、配列の1番目の値を参照しています。また`model.classes_`を参照することで、どのクラスが何番目に格納されているかが分かります。なお、`model.classes_`は辞書順でソートされています。

```
# seedを変えて、テストデータを生成
test_is_cv_list, test_is_treat_list, test_feature_vector_list = \
    generate_sample_data(sample_num, seed=42)

# それぞれの学習器でコンバージョンレートを予測
treat_score = treat_model.predict_proba(test_feature_vector_list)
control_score = control_model.predict_proba(test_feature_vector_list)

# スコアの算出、スコアは実験群の予測CVR / 統制群の予測CVR
# predict_probaはクラス所属確率のリストを返すため1番目を参照する
# numpy.ndarrayなので、そのまま割り算しても、要素の割り算になる
score_list = treat_score[:,1] / control_score[:,1]
```

これで、ある顧客が介入行為によってコンバージョンに転じるかどうかの指標ができました。続いて、この指標が正しく機能するかどうかを調べます。

まずはスコアの大きい順にソートし、10パーセンタイルごとにコンバージョンレートを算出して比較します。Uplift Modelingが正しく機能していれば、スコアが高いところでは統制群のコンバージョンレートが低く、実験群のコンバージョンレートが高くなっているはずです。またスコアの低いところでは、逆になっているはずです。

```
import pandas as pd
import matplotlib.pyplot as plt
from operator import itemgetter
plt.style.use('ggplot')
%matplotlib inline

# スコアが高い順に並べ替える
result = list(
```

```
zip(test_is_cv_list, test_is_treat_list, score_list))
result.sort(key=itemgetter(2), reverse=True)

qdf = pd.DataFrame(columns=('treat_cvr', 'control_cvr'))

for n in range(10):
    # 結果を10%ごとに切断
    start = int(n * len(result) / 10)
    end = int((n + 1) * len(result) / 10) - 1
    quantiled_result = result[start:end]

    # 実験群と統制群の数を数える
    treat_uu = list(
        map(lambda item: item[1], quantiled_result)
    ).count(True)
    control_uu = list(
        map(lambda item: item[1], quantiled_result)
    ).count(False)

    # 実験群と統制群のコンバージョン数を計測
    treat_cv = [item[0] for item in quantiled_result
                if item[1] is True].count(True)
    control_cv = [item[0] for item in quantiled_result
                  if item[1] is False].count(True)

    # コンバージョンレートに変換し、表示用のDataFrameに格納
    treat_cvr = treat_cv / treat_uu
    control_cvr = control_cv / control_uu

    label = "{}%~{}%".format(n * 10, (n + 1) * 10)
    qdf.loc[label] = [treat_cvr, control_cvr]

qdf.plot.bar()
plt.xlabel("percentile")
plt.ylabel("conversion rate")
```

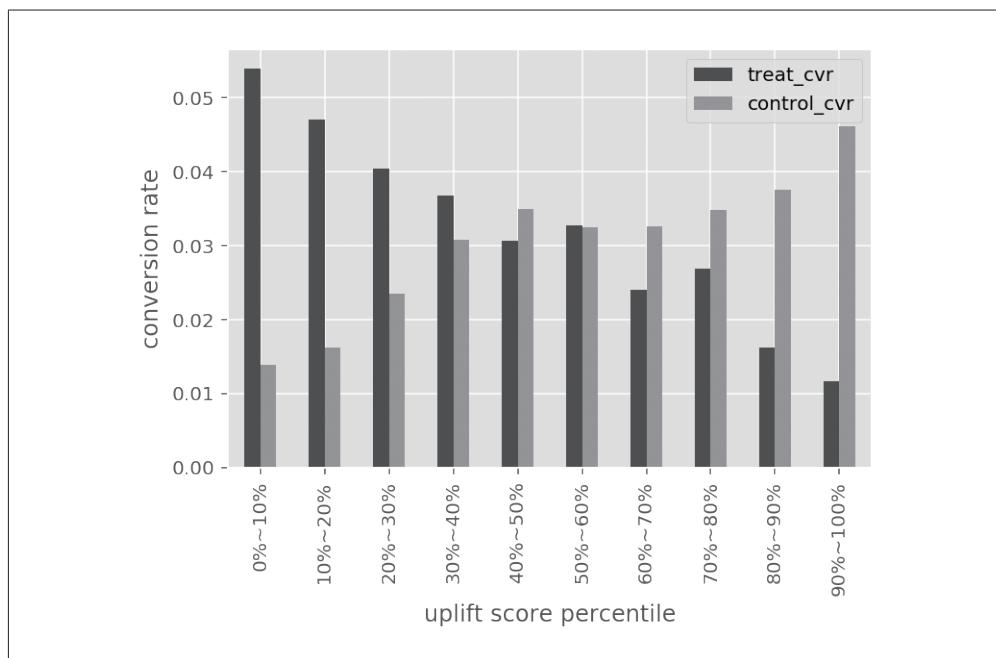


図9-1 10パーセンタイルごとにコンバージョン率を可視化

10パーセンタイルごとにコンバージョンレートを可視化した結果から、スコアが高いほど実験群のコンバージョンレートが高く、統制群のコンバージョンレートが低くなることが確認できました。これにより、Uplift Modelingがうまく動いていること分かります。また、グラフから、スコアの上位40%にだけ介入行為を行うことで、全体のコンバージョンレートが改善できそうであることが分かりました。

9.5 Uplift Modellingの評価方法、AUUC

続いて、Uplift Modelingの評価を行います。Uplift Modelingの評価には、Area Under the Uplift Curve (AUUC) という指標を使います。AUUCの値は大きければ大きいほど、Uplift Modelingの性能が高いと言えます。

AUUCの算出にはliftという指標を用います。liftは、あるスコア以上の顧客には介入行為を行い、あるスコア未満の顧客には介入行為を行わなかった場合、介入行為を行わなかった場合と比較して、どれくらいコンバージョン件数が増えたか、という値です。

AUUCは、liftをランダムに介入行為を行った場合と比べて、どれくらいコンバージョンが増えるのか、というのを正規化した値になります。

したがってAUUCを算出するためには、次の手順が必要です。

1. スコアの高い順に走査し、その時点までのコンバージョンレートを計測する。
2. コンバージョンレートの差から、介入行為によるコンバージョンの上昇数 (lift) を算出する。
3. ランダムに介入を行った場合の想定コンバージョン上昇数として、lift の原点と終点を結んだ直線をベースライン (base_line) とする。
4. lift と base_line に囲まれた領域の面積を算出し、正規化し、これを AUUC とする。

以下のコードは上記の手順を実装したものになります。

```
# スコア順に集計を行う
treat_uu = 0
control_uu = 0
treat_cv = 0
control_cv = 0
treat_cvr = 0.0
control_cvr = 0.0
lift = 0.0

stat_data = []

for is_cv, is_treat, score in result:
    if is_treat:
        treat_uu += 1
        if is_cv:
            treat_cv += 1
        treat_cvr = treat_cv / treat_uu
    else:
        control_uu += 1
        if is_cv:
            control_cv += 1
        control_cvr = control_cv / control_uu

    # コンバージョンレートの差に実験群の人数を掛けることで lift を算出
    # CVR の差なので、実験群と統制群の大きさが異なっていても算出可能
    lift = (treat_cvr - control_cvr) * treat_uu

    stat_data.append(
        [is_cv, is_treat, score, treat_uu, control_uu,
         treat_cv, control_cv, treat_cvr, control_cvr, lift])

# 統計データを、DataFrame に変換する
df = pd.DataFrame(stat_data)
df.columns = \
    ["is_cv", "is_treat", "score", "treat_uu",
     "control_uu", "treat_cv", "control_cv",
     "treat_cvr", "control_cvr", "lift"]

# ベースラインを書き加える
df["base_line"] = \
    df.index * df["lift"][len(df.index) - 1] / len(df.index)
```

```
# 可視化を行う
df.plot(y=["treat_cv", "control_cv"])
plt.xlabel("uplift score rank")
plt.ylabel("conversion count")

df.plot(y=["treat_cvr", "control_cvr"])
plt.xlabel("uplift score rank")
plt.ylabel("conversion rate")

df.plot(y=["lift", "base_line"])
plt.xlabel("uplift score rank")
plt.ylabel("conversion lift")
```

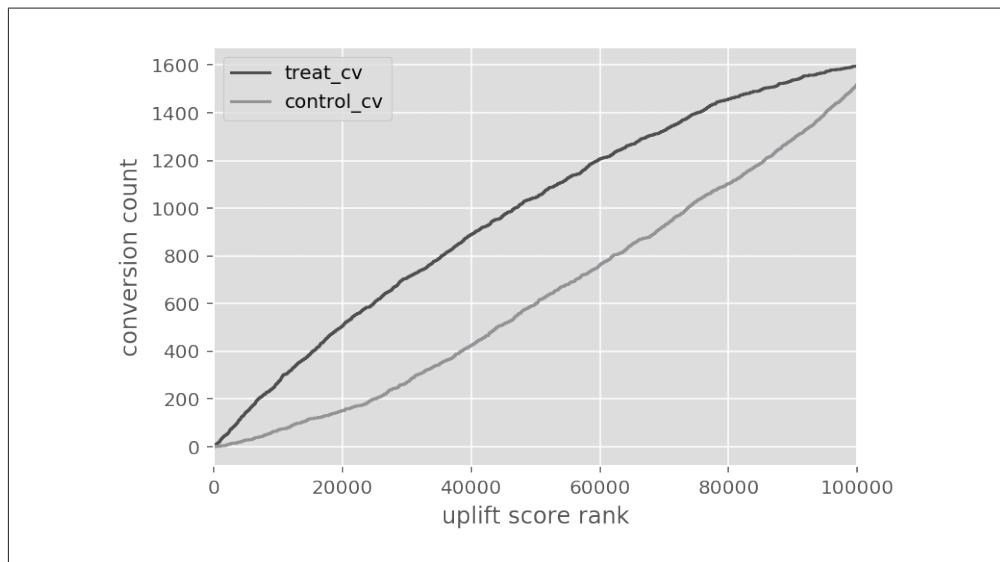


図9-2 実験群と介入群のコンバージョン件数の比較

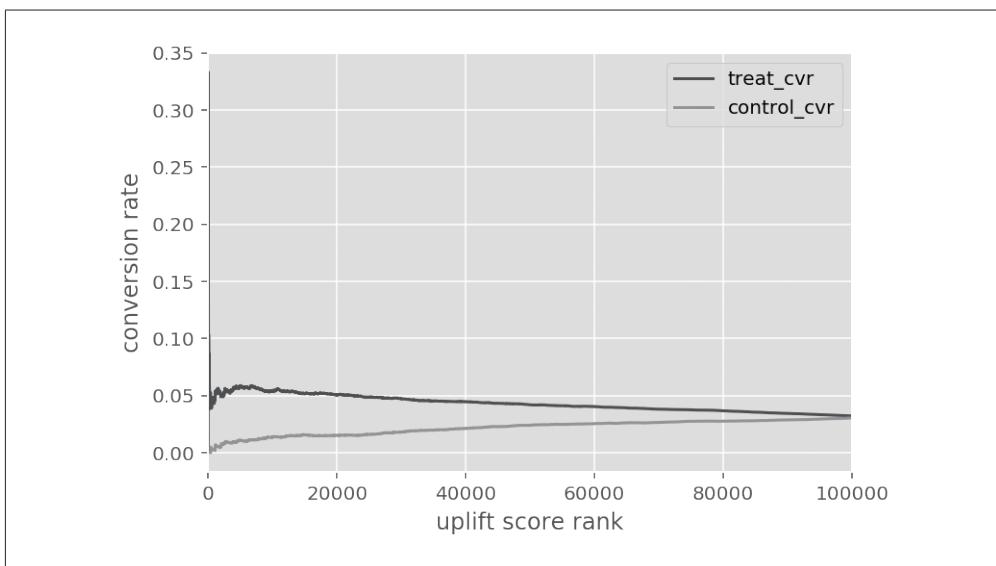


図9-3 実験群と加入軍のコンバージョンレートの比較

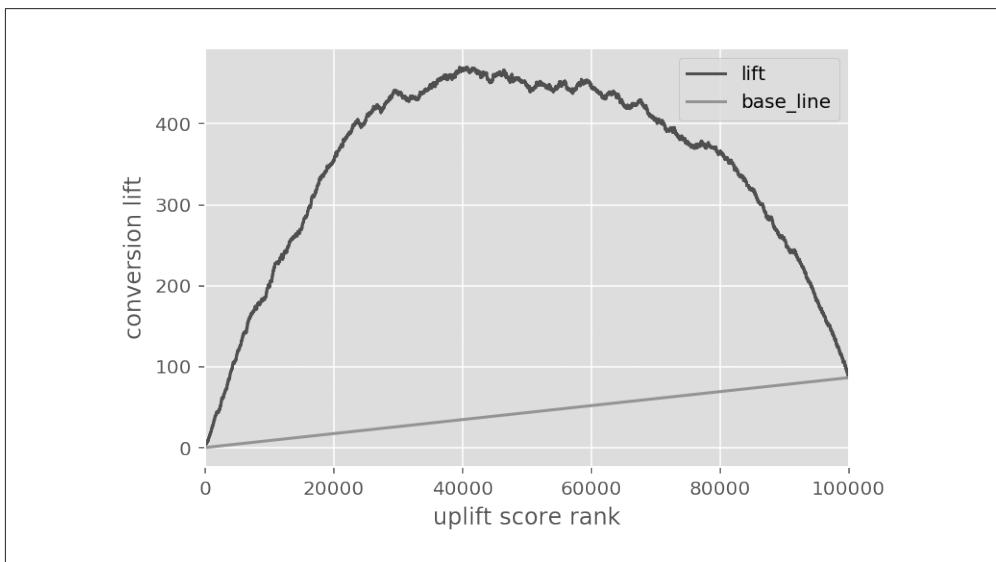


図9-4 コンバージョンレートの差から、コンバージョン上昇件数を推定

続いてAUUCを算出します。AUUCはliftとbase_lineの間に囲まれた領域を正規化したもののなので、次のように計算することができます。

```
auuc = (df["lift"] - df["base_line"]).sum() / len(df["lift"])
print("AUUC:", auuc)
```

AUUC: 302.246369848

Uplift Modelingの精度が高ければ高いほど、スコアの上位は実験群においてコンバージョンする顧客が集まり、統制群においてはコンバージョンしない顧客が集まります。スコアの下位についてはこの逆になります。そのため、liftの曲線は最初のうちは実験群のコンバージョンが集まるため正の傾きを持ち、精度が上がると傾きは急になります。逆に最後のほうでは統制群のコンバージョンが集まるため負の傾きを持ち、精度が上がると傾きは急になります。そのため、lift曲線は精度が上がれば上がるほど、より上に凸になり、liftとbase_lineに囲まれた面積が大きくなり、AUUCのスコアは大きくなります。

実運用を行う場合は、スコアに従って介入行為を実施するかどうかを決定する必要があります。そのため、どのスコア以上で介入行為を行うかを決定するために、横軸をスコアにしたLiftのグラフを可視化します。

```
df.plot(x="score", y=["lift", "base_line"])
plt.xlabel("uplift score")
plt.ylabel("conversion lift")
```

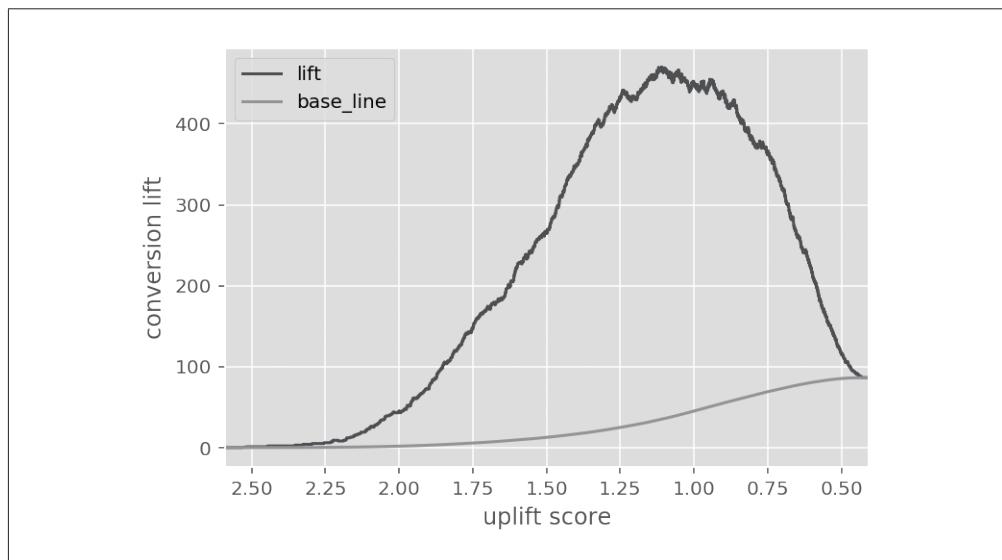


図9-5 横軸をスコアにしたLiftのグラフ

以上から、Uplift Modelingのスコアが1.2以上に対し介入行為を行うと、Liftが最大化できることが分かります。そのため、実運用を行う際には、今回作ったモデルを用いて予測を行い、Uplift

Modelingのスコアが1.2以上であれば、介入行為を行うようにします。

今回の例では2つの学習器を利用した簡易的なUplift Modelingを実装しました。Uplift Modelingには、決定木を活用したものや、SVMを拡張したアルゴリズムなども提案されています^{†3}。AUUCを用いることで、データセットが同一であれば、パラメータやアルゴリズムを差し替えることも評価することができます。実際の運用では、AUUCの値を元に複数のアルゴリズムを比較したり、パラメータのグリッドサーチを行って、最適な条件を探索します。

9.6 実践的な問題での活用

この節では、作成したアルゴリズムが実際の問題でどのように動くかを確かめます。

今回はThe MineThatData E-Mail Analytics And Data Mining Challenge^{†4}のデータセット^{†5}を利用します。このデータは、過去12か月に購買履歴のある顧客に対して、ランダムに「男性向けメールを送る」「女性向けメールを送る」「メールを送らない」という3種類の行動を実施し、その後、サイト訪問に結びついたか、商品を買ったか、を調べたものです。データの中身は次のようにになっています。

表9-5 データセットの内容

カラム名	内容
recency	最後に商品を買ってからの経過月数
history_segment	過去一年に費やされた購買金額セグメント
history	過去一年に費やされた実際の購買金額
mens	過去一年に男性向け商品を買ったか
womens	過去一年に女性向け商品を買ったか
zip_code	郵便番号をUrban(都市)、Suburban(郊外)、Rural(地方)として分類
newbie	過去12カ月以内の新規顧客であるか
channel	過去1年に客が購買したチャネル
segment	顧客に対してどのメールを送ったか
visit	メール受信後2週間以内にサイトに訪問したか
conversion	メール受信後2週間以内に商品を購入したか
spend	メール受信後2週間以内の購入金額

今回は、ある顧客に対して男性向けメールを送るべきか、女性向けメールを送るべきかの問題として取り扱い、コンバージョンはサイトへの再訪とします。

まずはデータを読み込み、ローカルに保存します。

^{†3} ポーランド科学アカデミーのSzymon Jaroszewicz博士の論文が参考になります。<http://www.ipipan.waw.pl/~sj/>

^{†4} <http://blog.minethatdata.com/2008/03/minethatdata-e-mail-analytics-and-data.html>

^{†5} http://www.minethatdata.com/Kevin_Hillstrom_MineThatData_E-MailAnalytics_DataMiningChallenge_2008.03.20.csv

```

import urllib.request
csv_url = "http://www.minethatdata.com/Kevin_Hillstrom_MineThatData_E-MailAnalytics_
DataMiningChallenge_2008.03.20.csv"
csv_filename = "source_data.csv"
with open(csv_filename, "w") as fp:
    data = urllib.request.urlopen(csv_url).read()
    fp.write(data.decode("ascii"))

```

次にpandasを利用して、CSVファイルを読み込み、データ構造を確認します

```

import pandas as pd
source_df = pd.read_csv(csv_filename)
source_df.head(10)

```

	recency	history_segment	history	mens	womens	zip_code	newbie	channel	segment	visit	conversion	spend
0	10	2) 100~200	142.44	1	0	Suburban	0	Phone	Womens E-Mail	0	0	0.0
1	6	3) 200~350	329.08	1	1	Rural	1	Web	No E-Mail	0	0	0.0
2	7	2) 100~200	180.65	0	1	Suburban	1	Web	Womens E-Mail	0	0	0.0
3	9	5) 500~750	675.83	1	0	Rural	1	Web	Mens E-Mail	0	0	0.0
4	2	1) 0~100	45.34	1	0	Urban	0	Web	Womens E-Mail	0	0	0.0
5	6	2) 100~200	134.83	0	1	Suburban	0	Phone	Womens E-Mail	1	0	0.0
6	9	3) 200~350	280.20	1	0	Suburban	1	Phone	Womens E-Mail	0	0	0.0
7	9	1) 0~100	46.42	0	1	Urban	0	Phone	Womens E-Mail	0	0	0.0
8	9	5) 500~750	675.07	1	1	Rural	1	Phone	Mens E-Mail	0	0	0.0
9	10	1) 0~100	32.84	0	1	Urban	1	Web	Womens E-Mail	0	0	0.0

図9-6 データ構造を確認する

今回は「男性向けのメールを送るか、女性向けのメールを送るか」という問題として扱うため、「メールを送らない」という実験をしたデータを捨てます。

```

mailed_df = source_df[source_df["segment"] != "No E-Mail"]
mailed_df = mailed_df.reset_index(drop=True)
mailed_df.head(10)

```

	recency	history_segment	history	mens	womens	zip_code	newbie	channel	segment	visit	conversion	spend
0	10	2) 100~200	142.44	1	0	Suburban	0	Phone	Womens E-Mail	0	0	0.0
1	7	2) 100~200	180.65	0	1	Suburban	1	Web	Womens E-Mail	0	0	0.0
2	9	5) 500~750	675.83	1	0	Rural	1	Web	Mens E-Mail	0	0	0.0
3	2	1) 0~100	45.34	1	0	Urban	0	Web	Womens E-Mail	0	0	0.0
4	6	2) 100~200	134.83	0	1	Suburban	0	Phone	Womens E-Mail	1	0	0.0
5	9	3) 200~350	280.20	1	0	Suburban	1	Phone	Womens E-Mail	0	0	0.0
6	9	1) 0~100	46.42	0	1	Urban	0	Phone	Womens E-Mail	0	0	0.0
7	9	5) 500~750	675.07	1	1	Rural	1	Phone	Mens E-Mail	0	0	0.0
8	10	1) 0~100	32.84	0	1	Urban	1	Web	Womens E-Mail	0	0	0.0
9	7	5) 500~750	548.91	0	1	Urban	1	Phone	Womens E-Mail	1	0	0.0

図9-7 「メールを送らない」データを捨てる

`zip_code`と`channel`はカテゴリー変数なので、ダミー変数に展開して、特徴ベクトルを作ります。

```
dummied_df = pd.get_dummies(
    mailed_df[["zip_code", "channel"]], drop_first=True)
feature_vector_df = \
    mailed_df.drop([
        "history_segment", "zip_code", "channel",
        "segment", "visit", "conversion", "spend"],
        axis=1)
feature_vector_df = feature_vector_df.join(dummied_df)
feature_vector_df.head(10)
```

	recency	history	mens	womens	newbie	zip_code_Surburban	zip_code_Urban	channel_Phone	channel_Web
0	10	142.44	1	0	0	1	0	1	0
1	7	180.65	0	1	1	1	0	0	1
2	9	675.83	1	0	1	0	0	0	1
3	2	45.34	1	0	0	0	1	0	1
4	6	134.83	0	1	0	1	0	1	0
5	9	280.20	1	0	1	1	0	1	0
6	9	46.42	0	1	0	0	1	1	0
7	9	675.07	1	1	1	0	0	1	0
8	10	32.84	0	1	1	0	1	0	1
9	7	548.91	0	1	1	0	1	1	0

図9-8 カテゴリー変数をダミー変数に展開

男性向けメールをTreatとしてフラグを付け、サイト訪問をコンバージョンとしてフラグを付けます。

```
is_treat_list = list(mailed_df["segment"] == "Mens E-Mail")
is_cv_list = list(mailed_df["visit"] == 1)
```

scikit-learnの`train_test_split`を利用して、ランダムに学習データと教師データに分けます。`train_test_split`は同じ長さの複数のリストを受け取り、学習データと教師データを返します。`test_size`はテストデータの比率を[0, 1]で指定します。また`random_state`を指定することで、分け方を固定できます。

```
from sklearn.model_selection import train_test_split

train_is_cv_list, test_is_cv_list, train_is_treat_list,
test_is_treat_list, train_feature_vector_df,
test_feature_vector_df = \
    train_test_split(is_cv_list, is_treat_list,
                    feature_vector_df, test_size=0.5,
                    random_state=42)

# indexをリセットする
train_feature_vector_df = \
    train_feature_vector_df.reset_index(drop=True)
test_feature_vector_df = \
```

```
test_feature_vector_df.reset_index(drop=True)
```

実験群と統制群の学習器をつくり、学習を行います。

```
train_sample_num = len(train_is_cv_list)

treat_is_cv_list = []
treat_feature_vector_list = []
control_is_cv_list = []
control_feature_vector_list = []

for i in range(train_sample_num):
    if train_is_treat_list[i]:
        treat_is_cv_list.append(train_is_cv_list[i])
        treat_feature_vector_list.append(
            train_feature_vector_df.loc(i))
    else:
        control_is_cv_list.append(train_is_cv_list[i])
        control_feature_vector_list.append(
            train_feature_vector_df.loc(i))

from sklearn.linear_model import LogisticRegression
treat_model = LogisticRegression(C=0.01)
control_model = LogisticRegression(C=0.01)

treat_model.fit(treat_feature_vector_list,
                treat_is_cv_list)
control_model.fit(control_feature_vector_list,
                  control_is_cv_list)
```

以後の集計と可視化については、前節のサンプルコードと同一であるため割愛し、グラフを描画します。

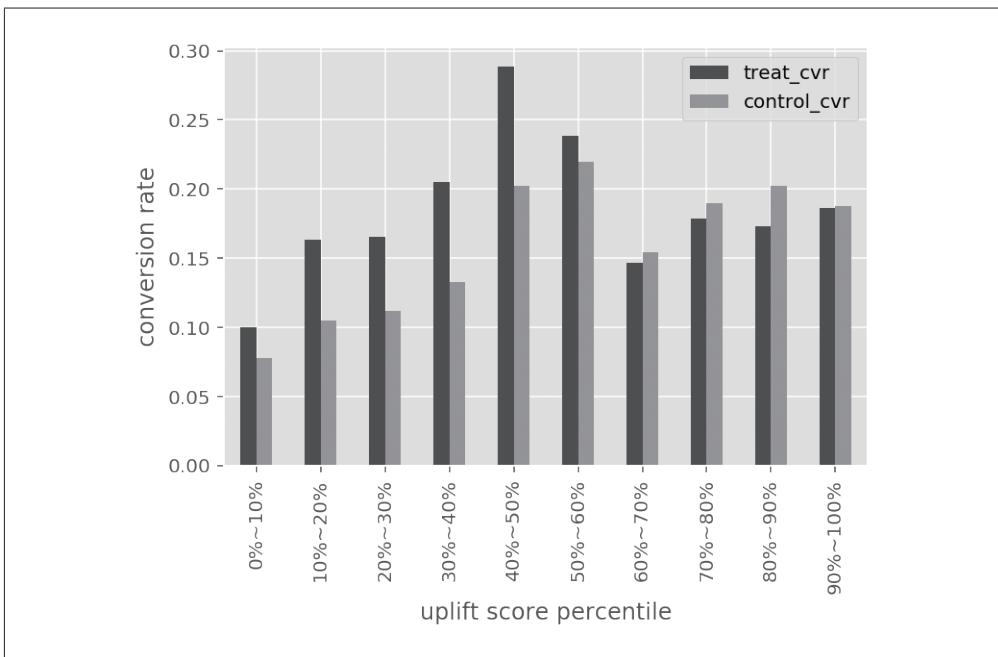


図9-9 10パーセンタイルごとにコンバージョン率を可視化

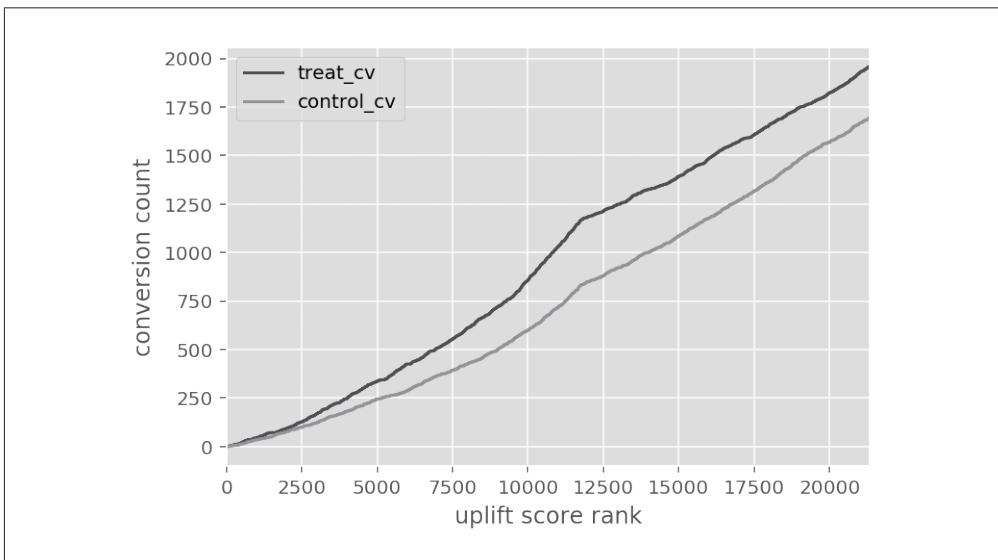


図9-10 コンバージョン件数の比較

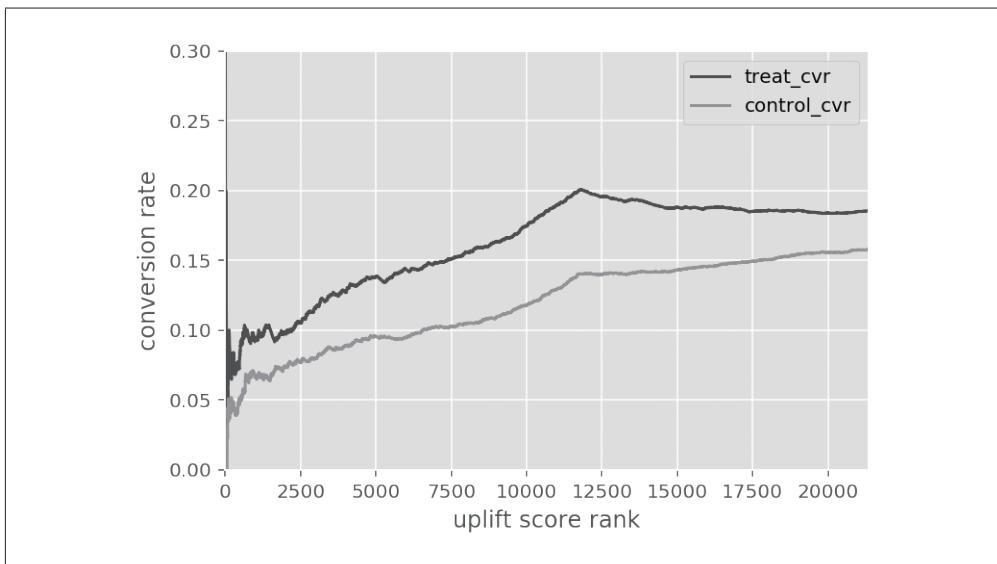


図9-11 コンバージョンレートの比較

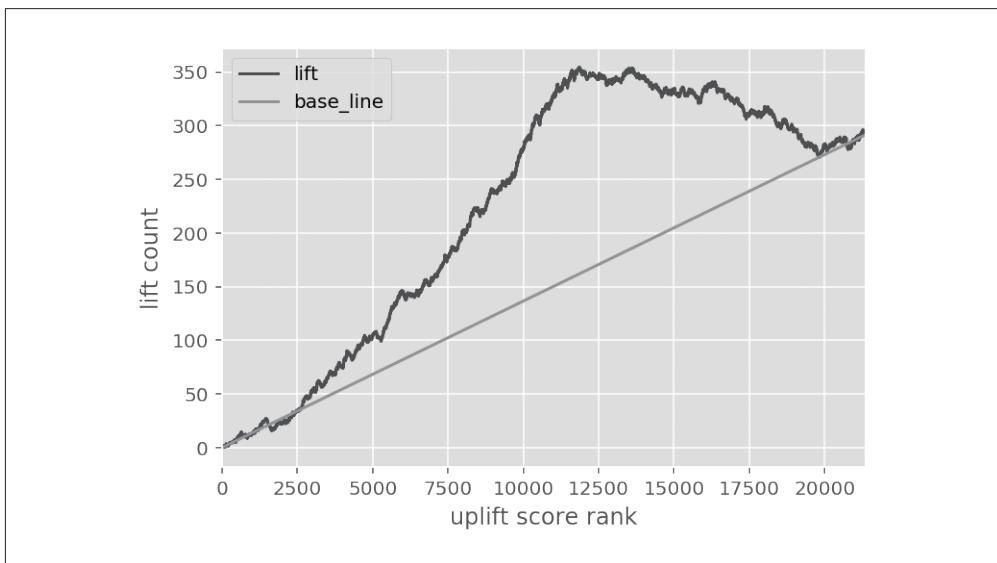


図9-12 コンバージョン上昇件数を可視化

10パーセンタイルごとのコンバージョンレートの比較図から、スコア上位60%のセグメントでは男性向けメールが女性向けメールよりもよく反応していることが分かります。また逆に下位40%は女性向けメールのほうが反応率がわずかに良いようです。

そのため、スコアの上位60%には男性向けメールを送付し、スコアの下位40%には女性向けメールを送付することで、より効果的なメール配信を実現できる可能性があることが分かりました。

9.7 Uplift Modeling を本番投入するには

Uplift Modeling を本番投入するには、次のような流れが必要です。1～5については、ここまで説明したアルゴリズムの作成と確認の中で実施します。本番環境で動かすには、6～11のプロセスが必要になります。

1. 実験したい介入行為を設計し、実験群と統制群で何を行うかを決める
2. 顧客の一部に対して、ランダム化比較試験を実施する
3. ランダム化比較結果を学習データとテストデータに分ける
4. 学習データから Uplift Modeling の予測器を作成する
5. テストデータから、Uplift Modeling のスコアの予測結果のグラフを描き、挙動を確かめる
6. Uplift Modeling のスコアのグラフから、スコアがいくつ以上の顧客に対して、介入行為を実施するのかを決める
7. 残りの顧客に対して、Uplift Modeling のスコアを予測する
8. 予測されたスコアから、介入行為を実施する対象の顧客を選出する
9. 選出された顧客のうち、一部を介入行為を実施しない対照群とし、残りを介入群とする
10. 介入群に介入行為を行う
11. 介入群と対照群のコンバージョンレートの比較を行い、介入行為の効果を計測する

以上の本番投入までの流れを図にすると、次のようになります。

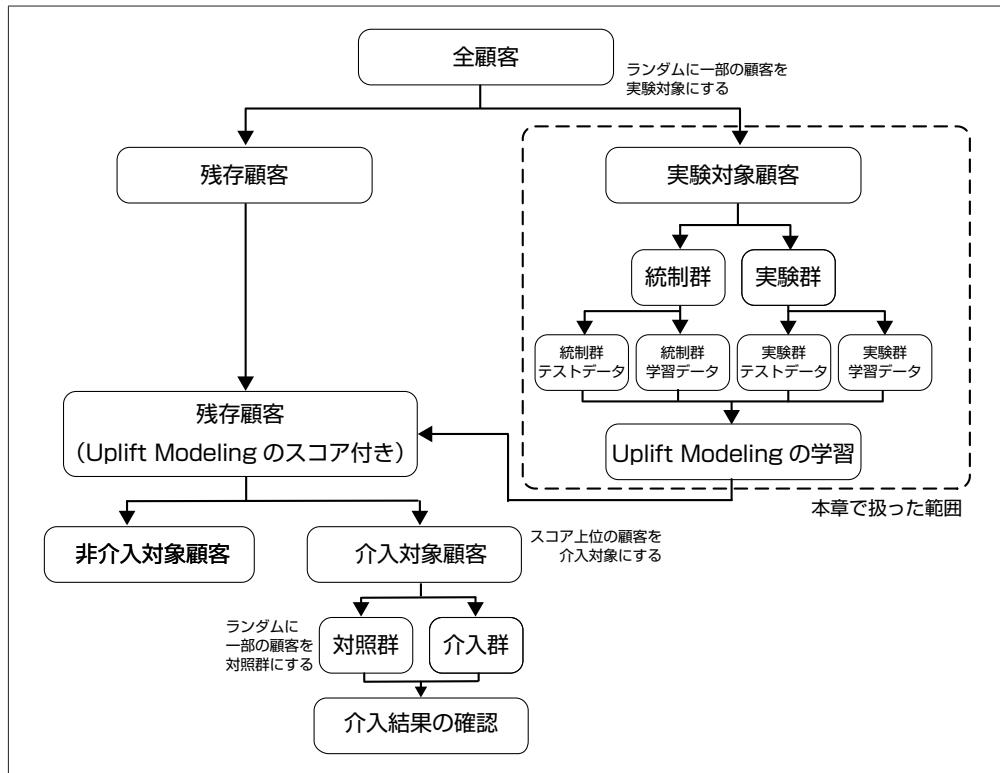


図9-13 Uplift Modelingを本番投入するまでの流れ

スコアが一定以上の顧客について対照群を用意せず、全員に介入行為を行うことも可能です。しかしこの場合、コンバージョンの増分については、テストデータによる結果から得た推定値でしか把握できません。

上記のような流れで本番投入することで、Uplift Modelingの活用の実施によってコンバージョンレートがどのくらい増えたかを実証できます。そして、Uplift Modelingによって増えた売上を明確な数字をもとに主張できるようになります。



Uplift Modelingは認知度が低く、効果を主張することが難しいことが多いため、介入群と対照群を用意することをお勧めします。

9.8 この章のまとめ

本章では、Uplift Modelingの概要を説明し、実際に簡易的なアルゴリズムを実装してみました。

Uplift Modelingは、ランダム化比較試験と顧客が持つ特徴量を組み合わせ、コンバージョンに転じやすい説得可能な顧客を予測するモデルを構築します。これにより、介入行為によってコンバージョンが増加するセグメントにのみ介入行為を実施することができ、逆に介入行為によってコンバージョンが減少するセグメントについては介入行為を控えることができます。

したがって、全員に介入行為を実施するよりも、多くのコンバージョンを得ることができるほか、介入行為に伴うマーケティング費用を大幅に削減することができ、マーケティング費用の効率的な分配を実現することができるでしょう

参考文献

- [sklearn_ml_book] Andreas C. Muller, Sarah Guido 著, 中田秀基訳.『Pythonではじめる機械学習—scikit-learnで学ぶ特微量エンジニアリングと機械学習の基礎』, オライリー・ジャパン, 2017.
- [python_ml] Sebastian Raschka 著, 株式会社クイープ 訳, 福島真太郎 監訳.『Python機械学習プログラミング 達人データサイエンティストによる理論と実践』, インプレス, 2016.
- [jupyter_book] 池内 孝啓, 片柳 薫子, 岩尾 エマ はるか, @driller. "PythonユーザのためのJupyter[実践]入門", 技術評論社, 2017.
- [dlscratch] 斎藤康毅.『ゼロから作るDeep Learning —Pythonで学ぶディープラーニングの理論と実装』, オライリー・ジャパン, 2016.
- [leanstartup] エリック・リース.『リーン・スタートアップ ムダのない起業プロセスでイノベーションを生みだす』, 日経BP, 2012.
- [runninglean] アッシュ・マウリヤ 著, 角征典 訳.『Running Lean —実践リーンスタートアップ (THE LEAN SERIES)』, オライリー・ジャパン, 2012
- [leananalytics] アリストア・クロール, ベンジャミン・ヨスコビッツ 著, 角征典 訳.『Lean Analytics —スタートアップのためのデータ解析と活用法 (THE LEAN SERIES)』, オライリー・ジャパン, 2015
- [dsculley] Sculley, D., Todd Phillips, Dietmar Ebner, Vinay Chaudhary, and Michael Young.『Machine learning: The high-interest credit card of technical debt.』(2014).
- [mlbestpractice] Martin Zinkevich, 『Rules of Machine Learning: Best Practices for ML Engineering』,http://martin.zinkevich.org/rules_of_ml/rules_of_ml.pdf
- [xgboost] Chen, Tianqi, and Carlos Guestrin.『XGBoost: A Scalable Tree Boosting System.』arXiv preprint arXiv:1603.02754 (2016).
- [sgb] Friedman, Jerome H.『Stochastic gradient boosting.』Computational Statistics & Data Analysis 38.4 (2002): 367–378.

- [gb] 小林淳一, and 高本和明.『確率勾配ブースティングを用いたテレコムの契約者行動予測モデルの紹介 (KDD Cup 2009 での分析より).』データマイニングと統計数理研究会 (第 12 回).
- [bergstra] Bergstra, James, and Yoshua Bengio.『Random search for hyper-parameter optimization.』The Journal of Machine Learning Research 13.1 (2012): 281–305.]
- [tpe] Bergstra, James S., et al. "Algorithms for hyper-parameter optimization." Advances in Neural Information Processing Systems. 2011.
- [tsne] van der Maaten, L.J.P.; Hinton, G.E.『Visualizing High-Dimensional Data Using t-SNE.』Journal of Machine Learning Research 9:2579–2605, 2008.
- [transfer] 神鳴敏弘.『転移学習.』人工知能学会誌 25.4 (2010): 572–580.
- [reinforcement-learning] 牧野貴樹, 濵谷長史, 白川真一ら.『これから強化学習』, 森北出版, 2016
- [ideanomaly] 井手剛.『入門 機械学習による異常検知—Rによる実践ガイド』, コロナ社, 2015
- [ideanomaly2] 井手剛, 杉山将.『異常検知と変化検知 (機械学習プロフェッショナルシリーズ)』, 講談社, 2015
- [kamishima] 神鳴敏弘, 推薦システムのアルゴリズム,<http://www.kamishima.net/archive/recsysdoc.pdf>, 2015
- [groupLens] Resnick, Paul, and Hal R. Varian.『Recommender systems.』Communications of the ACM 40.3 (1997): 56–58.
- [sarwar2001] Sarwar, Badrul, et al.『Item-based collaborative filtering recommendation algorithms.』Proceedings of the 10th international conference on World Wide Web. ACM, 2001.
- [fm2012] Rendle, Steffen.『Factorization machines with libfm.』ACM Transactions on Intelligent Systems and Technology (TIST) 3.3 (2012): 57.
- [implicitfm] Hu, Yifan, Yehuda Koren, and Chris Volinsky.『Collaborative filtering for implicit feedback datasets.』Data Mining, 2008. ICDM' 08. Eighth IEEE International Conference on. IEEE, 2008.
- [schafer] Schafer, J. Ben, Joseph A. Konstan, and John Riedl.『E-commerce recommendation applications.』Applications of Data Mining to Electronic Commerce. Springer US, 2001. 115–153.
- [contextawarerecom] Adomavicius, Gediminas, and Alexander Tuzhilin.『Context-aware recommender systems.』Recommender systems handbook. Springer US, 2011. 217–253.
- [Netflix_16] Gomez-Uribe, Carlos A., and Neil Hunt.『The netflix recommender system: Algorithms, business value, and innovation.』ACM Transactions on Management Information Systems (TMIS) 6.4 (2016): 13.

- [Sculley_15] Sculley, D., et al.『Hidden technical debt in machine learning systems.』Advances in Neural Information Processing Systems. 2015.
- [自然科学の統計学] 東京大学教養学部統計学教室, ed. 自然科学の統計学. Vol. 3. 東京大学出版会, 1992.
- [Benjamin_17] Benjamin, Daniel J., et al.『Redefine statistical significance.』Nature Human Behaviour (2017).
- [瀬々15] 瀬々 潤, 浜田 道昭, 講談社サイエンティフィク:生命情報処理における機械学習:多重検定と推定量設計 = Machine learning in bioinformatics, MLP機械学習プロフェッショナルシリーズ, 講談社 (2015)
- [岩波データサイエンス Vol3] 岩波データサイエンス刊行委員会, 岩波データサイエンス Vol. 3, 岩波データサイエンス, 岩波書店 (2016)
- [Microsoft_17] A/B Testing at Scale Tutorial <http://exp-platform.com/2017abtestingtutorial/>
- [David_17] Peeking at A/B Tests: Why it matters, and what to do about it David Walsh (Stanford University);Ramesh Johari (Stanford University);Leonid Pekelis (Stanford University)
- [ヤバい予測学] エリック・シーゲル 著, 矢羽野薫 訳.『ヤバい予測学 「何を買うか」から「いつ死ぬか」まであなたの行動はすべて読まれている』, 阪急コミュニケーションズ, 2013

あとがき

近年、機械学習を取り巻く環境は目まぐるしく絶え間なく変化し続けています。筆者の一人である有賀がこの本の最初の執筆を開始した2015年から2年以上経っているのですが、執筆期間中に多くの機械学習にまつわる変化が起きました。特に深層学習に関する変化は大きく、2012年にHintonがILSVCR (The ImageNet Large Scale Visual Recognition Challenge) と呼ばれるコンテストで深層学習を使い圧勝してから、執筆開始当初でも研究機関の間で深層学習は既に大きな広がりを見せっていました。それに加え、2015年のTensorFlowのリリースとともに一般のソフトウェアエンジニアにも爆発的に普及しました。その間にも、Pylearn2やTheanoといった深層学習の発展に大きく寄与してきたフレームワークが開発停止になるなど常に変化が起こっています。また、深層学習を利用した画像の物体認識や音声認識、音声合成、機械翻訳などは、普段の生活の中で当たり前に使われるようになり、スマートフォンがない時代には戻れないのと同じようなパラダイムシフトが起こっているのを感じています。

こうした中で、本書の執筆に取り組み続けていたのは、何度も同僚から機械学習の似たような質問を受ける度に、既存の理論寄りの書籍やハンズオン系の書籍ではカバーできていないけれど、業務の中では皆知っているべきことがたくさんあるのではないか、と思い続けてきたからです。恐らくそれは、探索的な分析をするということに対する経験の不足であったり、普段多くのソフトウェアエンジニアが取り組んでいるコンピュータシステムとの特性の違いであったりといった、データ分析と向き合うことで私たちが暗黙的に経験として学んできた事がまだまだ知られていないのではないかと思いました。敢えて深層学習に関してはあまり触れず、伝統的な機械学習を中心に本書を構成したのは、この本を読むことで私たちが普段の取り組みを通じてためてきた知見をお伝えしたかったからです。あまりこういったトピックに触れる機会がない独学で進んできた方にとって、本書が業務で機械学習を活用する上での一助となれば幸いです。

謝辞

本書の執筆には、多くの人達の協力なしには成し遂げられませんでした。

小宮篤史さんにはインターネット広告における実務の知見を元にしたレビューをしていただきました。shingoさんには、A/Bテストと効果検証について教えていただきました。

Python業界の会社の慰安旅行からはじまったコミュニティのPySpaからは、西尾泰和さん、上西康太さん、奥田順一さん、渋川よしきさん、若山史郎さん、山本早人さん、takabowさん、d1ce_さんからレビューをしていただきました。特に西尾さんには理論と歴史的な面での、奥田さんからは数学的な面での、上西さんからは対象読者であるソフトウェアエンジニアの観点から多岐にわたるレビューをいただきました。皆さんとのチャットでの議論が、本書の品質向上に繋がりました。

本書は2017年4月に開催された技術書典2、という同人誌展示即売会で頒布された、『BIG MOUSE DATA 2017 SPRING』という同人誌を下敷きにしています。当時、有賀が執筆していた本がなかなか出版することができなかつたため「どうせだから、みんなが抱えているお蔵入りした分析結果を寄せ集めて同人誌にしようぜ」という軽い気持ちで本書のプロジェクトはスタートしました。結果として同書は高い評価をいただき、オライリー・ジャパンから商業版の出版が行われることになりました。技術書典を主催されたTechBooster様、達人出版会様、および運営スタッフの皆様、そして同人誌版の出版を手伝っていただいた坪井創吾さん、hirekokeさんに感謝申し上げます。

企画の相談から、編集、図表から著者のケアまで多くの部分で支えていただいたオライリー・ジャパンの瀧澤さんのおかげで、ついに出版までこぎつけることができました。私たちの背中を押していただきありがとうございます。また、この本を生み出すきっかけを与えてくれたSBクリエイティブの杉山さんにも感謝します。

長期間に渡り明るく忍耐強く有賀を支えてくれた有賀恵理子と、結歌、夏織に感謝します。

●著者紹介

有賀 康顕（ありが みちあき）

電機メーカーの研究所、レシピサービスの会社を経て現在はCloudera所属。フィールドデータサイエンティストとして、データ活用や機械学習の支援を行う。

- <https://twitter.com/chezou>
- <https://www.slideshare.net/chezou>
- <https://chezo.uno/>

中山 心太（なかやま しんた）

電話会社の研究所、ソーシャルゲームの会社、機械学習によるウェブマーケティングの会社、フリーランスを経て、現在は株式会社Next Intを起業。自社サービスの開発のほか、ゲーム開発における企画や、機械学習案件の受託を行う。

機械学習、ゲームデザイン、ビジネス設計、新規事業企画等、広く薄く何でもやる高機能雑用。

- <https://twitter.com/tokoroten>
- <https://www.slideshare.net/TokorotenNakayama>
- <https://medium.com/@tokoroten/>

西林 孝（にしばやしたかし）

ソフトウェアエンジニア。独立系SIer、ソフトウェアベンダーを経て現在は株式会社VOYAGE GROUP所属。インターネット広告配信サービスの広告配信ロジックの開発に従事。

- <https://hagino3000.blogspot.jp/>
- <https://speakerdeck.com/hagino3000>
- <https://twitter.com/hagino3000>

●表紙の説明

本書の表紙の動物はオオアルマジロ（Giant armadillo）です。被甲目アルマジロ科オオアルマジロ属に分類される哺乳類で、本種のみでオオアルマジロ属を構成します。

南アメリカのサバンナから熱帯雨林に至る広い範囲に生息しており、夜行性で危険を感じると穴に隠れてしまうため発見するのが難しいそうです。大きなもので体長は 150 センチメートル、体重は 50 キロにもなります。背中が硬い鱗で覆われていますが、この鱗は体毛がうろこ状に変化したものです。アルマジロというと丸くなつて身を守るイメージがありますが、オオアルマジロにはこのような習性はありません。

森林やその周りの草原などに生息し、水辺を好みます。主にシロアリを食べますが、他にもアリや他の昆虫や幼虫、クモ、ミミズ、ヘビ、動物の死骸、植物などを食料にしています。前肢で蟻塚や地面を掘り起こし、中にいる獲物を食べます。一部で食用にもされることなどから乱獲され、生体数が激減、絶滅の可能性が高い動物といわれています。国際自然保護連合（IUCN）のレッドリストでは、絶滅危惧 II 類（危急）に分類されています。

仕事ではじめる機械学習

2017年10月22日 初版第1刷発行

著 者 有賀 康顕 (ありが みちあき)、中山 心太 (なかやま しんた)、
西林 孝 (にしばや したかし)

発 行 人 ティム・オライリー

制 作 株式会社トップスタジオ

印 刷・製 本 日経印刷株式会社

発 行 所 株式会社オライリー・ジャパン

〒160-0002 東京都新宿区四谷坂町12番22号

Tel (03)3356-5227

Fax (03)3356-5263

電子メール japan@oreilly.co.jp

Printed in Japan (ISBN978-4-87311-821-5)

乱本、落丁の際はお取り替えいたします。

本書は著作権上の保護を受けています。本書の一部あるいは全部について、株式会社オライリー・ジャパンから文書による許諾を得ずに、いかなる方法においても無断で複写、複製することは禁じられています。