Chapter 1

導入：関係データ解析とは

本章では，機械学習の文脈における関係データ解析の導入として，関係データとはどのようなものか，そして関係データの解析ではどのようなことが話題となるかについて説明します．

1.1 統計的機械学習

そもそもデータを使って「学習する」とはどういうことでしょうか．まずはこの点からはじめたいと思います．

1.1.1 データからの学習

統計的機械学習 (**statistical machine learning**) は，多くのデータが与えられたときに，それらデータのもつ統計的な特徴を活用して所望の処理を行う技術です．たとえば，「自社の購買データを解析して，売れ筋の商品群を見つけたい」というタスクを考えます．この場合，まずは昨日 1 日の売り上げ情報を参照します．この売り上げ情報を x とすると，この x の内容を精査することで，たとえば昨日最も売り上げた個数が多かった商品 z が見つかります．靴屋さんの場合なら，$z =$ “メーカー N 社製のスニーカー” などとなります．しかし，昨日の売り上げだけでは，その商品を売れ筋商品とするのは根拠に欠けます．そこで，仮に過去 50 日分にさかのぼって売り上げ

情報を収集して，その全体から売れ筋商品群を見つけることを考えます．このとき，売り上げ情報は 50 日分とたくさんあるので，日付を表す**インデックス** (**index**) $i \in \{1, 2, \ldots, 50\}$ を導入して区別しましょう．すると，目的のタスクは，$\boldsymbol{X} = \{x_i\}, i = 1, \ldots, 50$ という**観測データ** (**observations, observed data**) が与えられたとき，その中から他の商品に比べて売り上げ高が高い商品群を発見する問題といいかえられます．この例のように，観測データ $\boldsymbol{X} = \{x_i\}$ というデータ集合を用いて何らかの問題を解決する，という枠組みが統計的機械学習では最も頻繁に用いられます．

今度は画像認識問題を考えてみましょう．近年の巨大 ICT 企業の人工知能研究のトレンドの 1 つとして，大量の画像データから，「画像に写っているものは何かを推定する」[*1] 技術が発展しています．これらの問題は，統計的機械学習の最先端の課題の 1 つですが，枠組みとしては 1 点を除いて上記の例とそれほど変わりません．その点とは，**ラベル** (**label**) 情報が必要な点です．学習機械は，たとえば「犬」や「ピザ」とは何かをまったく知らないため，これを教えるための情報が必要です．たとえば i 番目の画像が柴犬の画像だったとすると，画像データ x_i に加えて，その内容を表すラベル情報 $y_i =$ “柴犬” を同時に準備します．したがって，目的のタスクは，N 枚の画像および対応するラベル情報からなる観測データ $\{\boldsymbol{X}, \boldsymbol{y}\} = \{(x_i, y_i)\}, i = 1, 2, \ldots, N$ を用いて，未知の画像 x からラベル y を計算する関数（写像）を推定する問題といいかえることができます．

1.1.2 教師有り学習と教師無し学習

売れ筋商品発見の問題では，「売れ筋商品とは何か」という答え，あるいは正解例は与えられていません．ですので，問題を解く側が「売れ筋商品とは何か」という定義について何らかの仮定を与え（上記の例では「売り上げ高が高い商品」），観測データ $\boldsymbol{X}$ を解析します．一方，画像認識の問題では，各画像データ x_i に対して，「これは犬です」「これはピザです」といったラベル情報 y_i が付与されています．したがって，このラベル情報をできるだけ正確に再現できるように画像データ $\boldsymbol{X}$ を利用します．このラベル情報は，

*1 たとえば http://googleresearch.blogspot.jp/2014/11/a-picture-is-worth-thousand-coherent.html,http://research.microsoft.com/en-us/news/features/dnnvision-071414.aspx

学習機械にとっての正解となるため，**教師情報**とも呼ばれます．そして，画像認識問題のように教師情報つきのデータセットを用いる問題を**教師有り学習** (**supervised learning**)，一方で商品群同定問題のように教師情報のないデータセットを用いる問題を**教師無し学習** (**unsupervised learning**) と呼びます．

教師有り学習に用いる教師データは，通常人間の手作業で維持・更新されます．たとえば，売れ筋の商品群の同定問題に教師情報をつけることを考えると，

- 取り扱う商品全体に対して専門的な知識をもつ人に意見をきく
- 個別の商品ドメインごとの専門家の意見を集約する
- 収集した意見を，目的のタスクを解決するために有効であると思われる形式の教師情報へと変換する

といった作業が必要になると想定されます．すなわち，教師データの作成には大きなコストがかかるのが一般的です．クラウドソーシング (crowdsourcing) によって大量かつ安価に教師データを作成することも可能ですが，一般にクラウドソーシングで得られる教師データは質が低いという問題がありますし，安価とはいってもやはり時間的，金銭的に負担が発生することは免れません．一方で，正解がわかっているため，その正解を再現するためのタスクにおいては教師無し学習よりも教師有り学習のほうがよい性能を示すものとされています[6]．教師無しデータは，利用可能なデータ群を機械的に手元に収集すればデータセットが作成可能です．目的のタスクについてのヒント（教師情報）がないため，ユーザ側が設計・選択する部分，つまり特徴量や統計的な振る舞いに関する仮定などが間違っている場合，目的のタスクでの性能が上がらない可能性があります．

本書で扱う関係データ解析の多くの課題では，教師無し学習を想定します．

1.2 関係データとは

機械学習技術の文脈における**関係データ** (**relational data**) とは，複数のデータの間に観測，定義される「関係」に着目したデータのことです．

最も典型的な関係データは，ソーシャルネットワークサービス (Social Net-

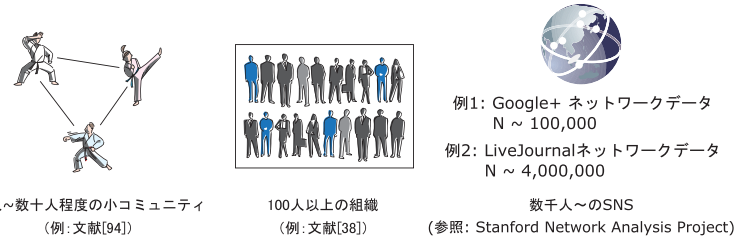


図 1.1　人間関係のネットワークの例.

work Service, SNS) のユーザ間の関係情報でしょう．SNS の各ユーザはアカウントを開設することで SNS サービスの世界のオブジェクトとなります．そして，SNS はこれらオブジェクト間に「フォロー」,「友人」などといった「関係」を定義して，オブジェクト同士のつながりに基づいたサービスを提供します．SNS の本質的な情報は，「どのユーザとどのユーザがつながっているか」というユーザ間の関係にあり，この関係の情報が本書で興味の対象となる関係データになります．SNS はアクティブにサービスを利用してもらい，さらにつながりを広げていってもらうようにさまざまな工夫を凝らします（例：ユーザが興味をもつ，あるいは知り合いの可能性があるユーザを紹介するなど）．そのためには関係データの解析技術を用いて，より効率のよい施策を検証する必要があります．

このような「人間関係」は，社会的な営みがある以上どこにでも観測される情報であり，その解析技術は広範な応用範囲をもつことが期待されます．古い研究では 10 数人程度の小さなコミュニティ内の人間関係のデータが広く用いられていました．有名な karate club データ[94] はその代表例です．また，100 人以上のコミュニティのデータとしては Enron 社内の E メールデータセット[38] が有名です．ここ 2, 3 年では SNS サービスのサブセット（部分集合）を収集して数千人，数万人，あるいはさらに大きな桁数のユーザネットワークを解析することも可能になっています（**図 1.1**）[89, 92]．

人間関係のネットワークは，古くは感染症の拡大，流行のモデル化[36] などの目的でさかんに研究されてきました．これらの技術は現在でも広範な関係データ解析タスクで利用されています．たとえば，SNS 上の人間関係データ解析には，口コミマーケティング (viral marketing) の効率化というタスク

があります．その場合，ネットワーク内の誰に，どのような情報を，どのようなタイミングで入力すればネットワーク内で所望の情報拡散を実現できるかが興味の対象となります[35, 93]．

インターネットのホームページ間のハイパーリンクもホームページとホームページの間の「リンク」という関係に相当します．Internet live stats[*2] によれば現在の世界のホームページ総数は，本書の執筆時点でほぼ 10 億ページとなっており，インターネットのハイパーリンクグラフは最も巨大な関係データの 1 つといえるでしょう．インターネットの各ページはそれ自体が多くの情報を含んでいるため，各ページの内容を観測データと考えると，ページ総数を N とする観測データとして解析が可能となります．しかし，関係データとしてとらえる場合には，多くの場合，各ページをつなぐハイパーリンク構造のみに興味があります．たとえば，多くのページからリンクを張られているページにはどのようなものがあるか，各ページのもつリンク数の分布はどうなっているか，あるいはハイパーリンクの分布からどのページがどの程度価値があるのか[63]，など，この巨大な関係データは多くの技術的課題の検証題材となり得ます．

ここまでの例では SNS のユーザ同士，ホームページ同士というように同じ種類のオブジェクト同士の関係に着目してきましたが，異なる種類のオブジェクトの間にも関係は定義できます．最も簡単な例として，購買履歴データがあります．すでに説明した「売れ筋商品の発見」の例を考えます．各日のデータ x には，どの商品がどれだけ売り上げを記録したかが保存されているという想定でした．ここで，購買行動の相手側，すなわち顧客の情報が得られると仮定します．つまり，「どの商品が，どの顧客に，どれだけ売れたか」が追跡できる状況です．この場合，「商品」という特定の種類のオブジェクトと，「顧客」という異なる種類のオブジェクトの間に「購入した」という関係を定義することができるでしょう．すなわち，購買履歴データは商品と顧客の間の関係データとして解析することも可能になるということです．ほかにも，特許や論文などの技術文書データは，そのキーワードや参照する文献，あるいは発明人・論文著者名などで検索することができます．この場合，「文書」,「キーワード」,「参照文書」,「著者」といった異なるオブジェク

*2　http://www.internetlivestats.com

ト間の関係データとして表現できます．データベースを日常的に扱う読者にとって，「関係データ」という語は関係データベース (relational database) を想起させるかもしれません．結論からいうと，本書で取り扱う関係データはすべて関係データベースで表現することが可能です．関係データベースの場合には，より多種類のオブジェクトを多種類の「関係」で対応づけます．これは本書で取り扱う関係データをさらに拡張した類のデータセットとなります．関係データ解析技術は近年急速に発展していますが，オブジェクト間に非常に多種類の「関係」が定義可能な場合に対する技術は，まだ成長途上といえます[30, 48]．

最後に，これらの関係データ（ネットワーク）セットの入手先について述べます．最もよくまとまっているものとしてスタンフォード大学のレスコベック准教授のチームによる Stanford Network Analysis Project*3 を紹介しておきます．レスコベック准教授は，まだ若年ながら関係データ（ネットワーク）の研究の世界的な権威といってよい研究者です．同ページからはデータセット以外にも興味深い情報を得ることができると思います．

1.3　関係データの表現

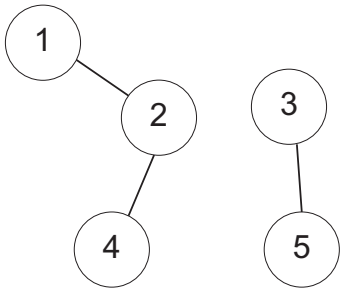
本書では関係データを統計的に取り扱うため，その数学的な表現方法が重要となります．統計的機械学習の分野では，大きく分けてグラフ表現と行列（多次元配列）表現の 2 つの表現方法を用います．本書では主に後者の表現を利用しますが，比較のために両者について説明します．

1.3.1　関係データのグラフ表現

SNS や人間関係などのネットワーク関係データの解析タスクが対象の場合，関係データは**グラフ** (**graph**) の言葉で表現されるケースが多いです．ただし，ここでの「グラフ」は， 計算機科学の文脈におけるものです．

グラフ $G = \{V, E\}$ は**頂点** (**vertex**) の集合 $V = \{v_1, v_2, \ldots, v_{|V|}\}$ と「2 つの頂点の組」である**辺** (**edge**) の集合 $E = \{e_1, e_2, \ldots, e_{|E|}\}$ で定義されます．頂点はノード，辺はエッジとも呼ばれます．各辺は関係する頂点の対で

*3　http://snap.stanford.edu



頂点（ノード，オブジェクト）　$V = \{1, 2, 3, 4, 5\}$

辺（エッジ）　$E = \{(1, 2), (2, 4), (3, 5)\}$

図 1.2　一般に関係データはグラフで表現することが可能です．グラフは頂点と頂点間をつなぐ辺で構成されるため，オブジェクトを頂点，観測されたオブジェクト間の関係を辺とみなすことによってグラフ表現が可能であるからです．

特定できます．たとえば $e_1 = (v_1, v_2), e_2 = (v_1, v_4)$ といった具合です．

注目する関係データにおいて，関係で接続される主体を**オブジェクト** (**object**) と呼ぶと，グラフの頂点をオブジェクトと見立て，オブジェクト間に観測される関係を辺と対応づけることによって関係データをグラフで表現することが可能になります．簡単なグラフの例を**図 1.2** に示します．

グラフは基本的にネットワークを人の目で理解するための図形ですので，たとえば関係データの解析結果をアプリケーション上で美しく描画（可視化）することを想定している場合などに有用だと考えられます．また，グラフデータの解析は，古くから計算機科学でさかんに研究されており，**グラフ理論** (**graph theory**) に基づく各種グラフ処理技術の蓄積はしっかりとした教科書などにまとめられています．グラフ表現を用いることで，これらの資産を関係データ解析に用いることができるようになるのも利点です．興味のある方はまずは元北海道大学准教授の井上純一先生による講義ノート[24] を参照してください．教科書としてはウィルソンによる著書[88] があります．

1.3.2 関係データの行列（多次元配列）表現

伝統的なグラフ表現に対し，近年の統計的機械学習の文献においては，関係データを**行列** (**matrix**) あるいは 2 次元配列で表現することもよくあります．行列で表現することでグラフによる表現に比べて情報量が増えるわけではありませんが，数値的，統計的な最適化計算時の表現と親和性が高いこと，実装時の取り扱いの明快さ，および表現の容易さから，本書でも関係データは主として行列（多次元配列）として記述します．

なお，本書を通じて，ボールド体の大文字ラテン文字は行列あるいはベクトルの集合を（例：$\boldsymbol{X}, \boldsymbol{Y}$），ボールド体の小文字ラテン文字はベクトルを（例：$\boldsymbol{w}, \boldsymbol{x}_i$），通常フォントの大文字ラテン文字は定数を（例：$C, D$），通常フォントの小文字ラテン文字は単一の変数あるいはインデックスを（例：a, b）を表現するものとします．また，$\mathbb{R}$ は実数全体，$\mathbb{Z}$ は整数全体，$\mathbb{N}$ は自然数全体を，上つきの $+$ は非負を表します．

行列として関係データを表現する場合，**隣接行列** (**adjacency matrix**) による表現と**接続行列** (**connectivity matrix**) の 2 種類の表現がありますが，機械学習の文献では前者を用いることが多いため，本書では隣接行列だけを用いて表現します．

まず**図 1.3** をもとにして説明します．先ほどのグラフ表現した関係データ（グラフ）を行列 $\boldsymbol{X}$ として表現します．まず，オブジェクト（グラフ表現時のノード）を表すインデックスとして $i, j \in \{1, 2, \ldots, N\}$ を導入します．この図では $N = 5$ になっています．関係（グラフ表現時のエッジ）は 2 つの端点オブジェクトで指定できますので，オブジェクト i, j 間の関係を $x_{i,j}$ と表現します．i, j は 1 から N の間を動くので，i を行インデックス，j を列インデックスと考えると $x_{i,j}$ 全体を 1 つの行列 $\boldsymbol{X} = (x_{i,j}), i, j = 1, 2, \ldots, N$ としてコンパクトに表記することが可能であることがわかります．この行列は計算機プログラム上では 2 次元配列（たとえば C 言語などでは $x[i][j]$ とします）として表現することができます．$\boldsymbol{X}$ は行列なので，各 $x_{i,j}$ は何らかの数値をもつ必要があります．この点については次の 1.3.3 項で説明します．

関係データを行列表現することのメリットは，古くは多変量解析，近年でいえば深層学習に至るまで統計的機械学習でなじみの深いデータ表現であることです．したがって，統計的機械学習による関係データ解析を実施する際

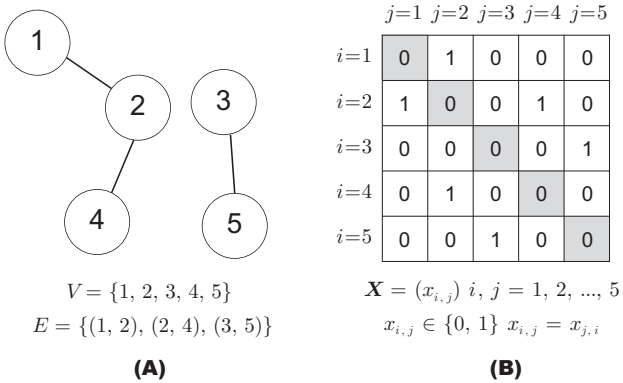


図 1.3 関係データの行列表現．(A) もととなる関係データのグラフ表現．辺には重みがなく，関係の有無だけが観測されていると仮定しています．(B) 対応する関係データの行列表現．関係がある辺は $x = 1$, 関係がない辺は $x = 0$ としています．自己リンク $x_{i,i}$ については (A) のグラフ表現の図からは読み取れないので，ここではすべての i について $x_{i,i} = 0$ としています．

に通常のデータに対する一般的な表現方法と方法論が利用可能となります．

また，行列表現することは線形代数による問題，モデル，アルゴリズムの理解と定式化を可能とします．このことで，計算機実装の際に高速な線形代数計算ライブラリを利用したり，線形代数，多変量解析の立場から新しいモデルを考案するといったことが可能になります．

1.3.3 関係の値の表現

2 つのオブジェクトの間の関係はどのような値をもちうるでしょうか．たとえば，すでに紹介した Enron 社内の E メールデータセット[38] のように，自社内のメール送受信データを集積して社員間の関係データとして収集，解析することを考えます．この社員同士の関係を対象として検討します．

まず，「リンクが存在するかどうか」のみを表現する方法があります．これはグラフ表現をするとわかりやすくなります．辺の集合 E 内にある辺（関係）e が存在する場合 $(e \in E)$ ならば当該の関係が「存在する」，$e \notin E$ ならば「存在しない」とする表現です．辺が存在する頂点対，つまり社員間は「関係がある」，辺が存在しない社員間は「関係がない」ということになりま

Chapter 4

行列分解

ここで 1.5.1 項で述べた，映画推薦の話に立ち戻ります．1.5.1 項では映画評価の予測に焦点を当てていましたが，これ以外にもいくつかのタスクが考えられます．たとえば，ある顧客と映画の趣味が似ているのはどの顧客でしょうか．また，ある映画と同じジャンルに属するのはどの映画でしょうか．評価データが大きすぎる場合，それをうまく圧縮する方法はないでしょうか．
行列分解は欠損値の予測だけでなく，これらのタスクも包括的に解くことができる，非常に強力な手法です．予測精度も高く，また基本アイデアはシンプルで実装も簡単です．本章では行列分解の基本的なアイデア，アルゴリズム，拡張について説明します．

4.1 準備

本章より使用する表記を導入します．

転置 行列 $\boldsymbol{A}$ の転置を $\boldsymbol{A}^\top$ と表記します．
縦ベクトルと横ベクトル 特に言及しない限り，任意のベクトル $\boldsymbol{a} \in \mathbb{R}^I$ は縦に並ぶもの，すなわち $I \times 1$ 行列として扱います．向きを変えたい場合，すなわち $1 \times I$ 行列として扱いたい場合は転置記号を用いて $\boldsymbol{a}^\top$ と表記します．
インデックス集合（添字集合） ある自然数 $N \in \mathbb{N}$ について，1 から N までの自然数の集合を $[N] = \{1, 2, \ldots, N\}$ と表記します．
単位行列とゼロ行列 $N \times N$ の単位行列を $\boldsymbol{I}_N$ と表記します．またすべての

要素の値が 0 で与えられる $N \times N$ 行列を $\boldsymbol{O}_N$ と表記します．ただしサイズが文脈より明らかな場合はそれぞれ省略して $\boldsymbol{I}, \boldsymbol{O}$ と表記します．

インデックス ある $I \times J$ 行列 $\boldsymbol{A}$ が与えられたとき，i 番目 $(i \in [I])$ の行ベクトルを $\boldsymbol{a}_{i:}^\top = (a_{i1}, a_{i2}, \cdots, a_{iJ})$，$j$ 番目 $(j \in [J])$ の列ベクトルを $\boldsymbol{a}_{:j} = (a_{1j}, a_{2j}, \cdots, a_{ij}, \cdots, a_{Ij})^\top$ と表記します．また $\boldsymbol{A}$ の (i,j) 成分を $[\boldsymbol{A}]_{ij}$ あるいは a_{ij} と表記します．まとめると以下のような表現になります．

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1J} \\ a_{21} & a_{22} & \cdots & a_{2J} \\ \vdots & \vdots & & \vdots \\ a_{I1} & a_{I2} & \cdots & a_{IJ} \end{pmatrix} = \begin{pmatrix} \boldsymbol{a}_{1:}^\top \\ \boldsymbol{a}_{2:}^\top \\ \vdots \\ \boldsymbol{a}_{I:}^\top \end{pmatrix} = \begin{pmatrix} \boldsymbol{a}_{:1} & \boldsymbol{a}_{:2} & \cdots & \boldsymbol{a}_{:j} \end{pmatrix}$$

また線形代数の表記を以下にまとめます．いずれも標準的な表記ですので，馴染みがある方は読み飛ばし，必要に応じて参照してください．線形代数についてより基礎的な部分から確認したい方は文献[22, 52] などを参照してください．

ベクトルの内積とノルム ベクトル $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^I$ の内積を $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{I} x_i y_i$ と定義します．またこの内積から自然に定義される ℓ^2 ノルムを $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$ と表記します．

行列の内積とノルム 行列 $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{I \times J}$ の内積を $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \sum_{i=1}^{I} \sum_{j=1}^{J} x_{ij} y_{ij}$ と定義します．またこの内積から自然に定義されるフロベニウスノルムを $\|\boldsymbol{X}\|_{\mathrm{Fro}} = \sqrt{\langle \boldsymbol{X}, \boldsymbol{X} \rangle}$ と表記します．

アダマール積 行列 $\boldsymbol{A}, \boldsymbol{A}' \in \mathbb{R}^{I \times J}$ のアダマール積 $\boldsymbol{A} * \boldsymbol{A}'$ は $I \times J$ 行列を返す演算であり，その (i,j) 成分 $(i \in [I], j \in [J])$ の値は $a_{ij} a'_{ij}$ で与えられるとします．

ベクトルと行列の積 $K \in \mathbb{N}$ に対し，ベクトル $\boldsymbol{a}, \boldsymbol{a}' \in \mathbb{R}^I, \boldsymbol{b} \in \mathbb{R}^J$ および行列 $\boldsymbol{A} \in \mathbb{R}^{I \times J}, \boldsymbol{B} \in \mathbb{R}^{J \times K}$ が与えられたときこれらの積は以下のように与えられるとします．

- $\boldsymbol{a}^\top \boldsymbol{a}'$ はスカラを返し，その値は $\langle \boldsymbol{a}, \boldsymbol{a}' \rangle$
- $\boldsymbol{a}\boldsymbol{b}^\top$ は $I \times J$ 行列を返し，その (i,j) 成分 $(i \in [I], j \in [J])$ の値は $a_i b_j$
- $\boldsymbol{A}\boldsymbol{b}$ は I 次元ベクトルを返し，その $i \in [I]$ 番目の値は $\boldsymbol{a}_{i:}^\top \boldsymbol{b}$
- $\boldsymbol{A}\boldsymbol{B}$ は $I \times K$ 行列を返し，その (i,j) 成分 $(i \in [I], j \in [J])$ の値は $\boldsymbol{a}_{i:}^\top \boldsymbol{b}_{:k}$

ランク 行列 $\boldsymbol{A} \in \mathbb{R}^{I \times J}$ に対し，非ゼロな固有値の数[*1] を $\boldsymbol{A}$ のランクと呼びます．定義より $\boldsymbol{A}$ のランクは 0 以上かつ $\min(I, J)$ 以下です．

微分 $\theta \in \mathbb{R}$ を引数にとる 1 回微分可能な関数 $f : \mathbb{R} \to \mathbb{R}$ に対し，導関数を $\frac{\partial f}{\partial \theta}$，またある点 $\theta' \in \mathbb{R}$ における微分係数を $\frac{\partial f}{\partial \theta}|_{\theta=\theta'}$ と表記します．

勾配 $\boldsymbol{\theta} \in \mathbb{R}^I$ を引数にとる 1 回微分可能な関数 $g : \mathbb{R}^I \to \mathbb{R}$ に対し，ある次元 $i \in [I]$ に関する偏導関数を $\frac{\partial g}{\partial \theta_i}$ と表記します．すべての次元に関する偏微分をまとめたものを勾配と呼び，$\nabla g = (\frac{\partial g}{\partial \theta_1}, \frac{\partial g}{\partial \theta_2}, \ldots, \frac{\partial g}{\partial \theta_I})^\top$ と表記します．また，ある点 $\boldsymbol{\theta}' \in \mathbb{R}^I$ における勾配の値を $\nabla g(\boldsymbol{\theta}') = (\frac{\partial g}{\partial \theta_1}|_{\boldsymbol{\theta}=\boldsymbol{\theta}'}, \frac{\partial g}{\partial \theta_2}|_{\boldsymbol{\theta}=\boldsymbol{\theta}'}, \ldots, \frac{\partial g}{\partial \theta_I}|_{\boldsymbol{\theta}=\boldsymbol{\theta}'})^\top$ と表記します．

4.2 単純行列分解

1.5.1 項で述べた映画推薦の例ではデータが「顧客が行，映画が列として並び，要素の値が評価値となるような行列」として与えられました．この行列のサイズは非常に大きなものになりうることに注意してください．町のレンタルビデオ店のように顧客数が多くなければ問題となりませんが，Amazon や Netflix のように世界中で展開しているようなインターネットサービスであれば数千万もの顧客をかかえることもあります．また映画に関しても毎年新しい映画が公開されていくことを考えると，年々その総数は大きくなっていくのは間違いありません．そのような状況の中で映画評価をそのまま生の値で保持していくのは，その規模の大きさゆえ難しい場合が考えられます．なにかよい方法はないでしょうか．

解決策の 1 つとして，データを圧縮することを考えましょう．顧客数を $I \in \mathbb{N}$，映画数を $J \in \mathbb{N}$ とします．このとき顧客 $i \in [I]$ による映画 $j \in [J]$ の評価を x_{ij} とおくと，これを集めたものは $I \times J$ 行列 $\boldsymbol{X}$ として表現できます．ここで x_{ij} の値が高ければ高いほど評価が高いことを示すとします．さて，ここである架空の顧客である $a \in [I]$ さんのことを考えます．a さんは

[*1] 固有値の定義については 2.3.3 項を参照してください．

映画マニアで，公開されているありとあらゆる映画を見たことがあり，それらすべてについて評価をつけているような人です．また a さんは派手な映画が好きで，特にアクションシーンが多くある映画には高評価をつけます．一方，ホラーは苦手で，そのような映画も見ることは見ますが必ず低評価をつけます．このようなとき，もし映画のジャンルが事前にわかっていれば a さんの評価はある程度予想することができるのではないでしょうか．すなわちある映画 $j \in [J]$ についてホラー成分があれば 1，なければ 0 をとるような変数 $v_{j,\,\text{ホラー}}$ と，同じくアクション成分があれば 1，なければ 0 をとるような変数 $v_{j,\,\text{アクション}}$ があったとすると，顧客 a さんの映画 j に対する評価は，たとえば

$$x_{aj} \simeq v_{j,\,\text{アクション}} - v_{j,\,\text{ホラー}} \tag{4.1}$$

のように予測できるでしょう．すなわち評価 x_{aj} は，もし映画 j がアクションでかつホラーでなければ 1，ホラーでかつアクションでなければ -1，それ以外の場合は 0 となるようなモデルです．

さて，a さんだけでなくほかの顧客のことも考えてみましょう．ほかの顧客の中には a さんと同じような嗜好 (アクション好き，ホラー嫌い) をもつ人もいるでしょうが，なかには正反対な嗜好 (ホラー好き，アクション嫌い) をもつ人もいるはずです．このことを考えると，アクション好きの度合い，またホラー好きの度合いを表現するような変数 $u_{i,\,\text{アクション}}, u_{i,\,\text{ホラー}}$ を $i \in [I]$ について用意することで，a さん専用だった予測式 (4.1) を以下のように一般化できます．

$$x_{ij} \simeq u_{i,\,\text{アクション}} v_{j,\,\text{アクション}} + u_{i,\,\text{ホラー}} v_{j,\,\text{ホラー}}$$

このように，すべての顧客のアクション，ホラーの好き具合 (全部で $2I$ 個の情報) とすべての映画のアクション，ホラー成分 (全部で $2J$ 個の情報) がわかっていたとすると，すべての評価はたかだか $2(I+J)$ 個の情報から復元できてしまうことがわかります．I と J が同じスピードで増えるとすると，すべての評価の数は IJ 個なので，2 乗オーダから線形オーダへと大幅に情報を圧縮できたといえます．

上記の例では映画の評価が「アクションとホラー」という 2 つの評価軸のみから決まるという極端な場合を考えていましたが，現実はもっと複雑で

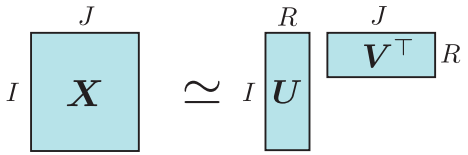


図 4.1 行列分解．

す．より一般の場合を考え，$R \in \mathbb{N}$ 個の評価軸が存在すると仮定しましょう．評価 x_{ij} はすべての顧客 $i \in [I]$，すべての映画 $j \in [J]$ に対して観測されているとし[*2]，これをまとめて $\boldsymbol{X} \in \mathbb{R}^{I \times J}$ と表記します．顧客 i の R 個の嗜好をまとめたものを $\boldsymbol{u}_{i:} \in \mathbb{R}^R$，またそれを行方向にまとめたものを $\boldsymbol{U} \in \mathbb{R}^{I \times R}$ とします．同様に映画 j の R 個の成分をまとめたものを $\boldsymbol{v}_{j:} \in \mathbb{R}^R$，またそれを行方向にまとめたものを $\boldsymbol{V} \in \mathbb{R}^{J \times R}$ とします．このとき，$x_{ij} \simeq \sum_{r=1}^{R} u_{ir} v_{jr} = \boldsymbol{u}_{i:}^\top \boldsymbol{v}_{j:}$ であり，これを $i \in [I], j \in [J]$ についてまとめると以下のように書くことができます．

$$\boldsymbol{X} \simeq \boldsymbol{U}\boldsymbol{V}^\top \tag{4.2}$$

このとき，IJ 個の成分をもつ $\boldsymbol{X}$ は IR 個の成分をもつ $\boldsymbol{U}$ と JR 個の成分をもつ $\boldsymbol{V}$ の積によって表現され，仮に $I = J$ とすると，R が I よりも十分小さい場合は情報が I^2 から $2IR$ へと $2R/I (\ll 1)$ 倍に縮約されたといえます (**図 4.1**)．

$\boldsymbol{X}$ を式 (4.2) の形に分解することを $\boldsymbol{X}$ の**ランク R 行列分解** (**rank-R matrix decomposition**)，また $\boldsymbol{U}$ と $\boldsymbol{V}$ を**因子行列** (**factor matrix**) と呼びます．

注意

行列分解の名前に「ランク R」とついているとおり，$\boldsymbol{U}, \boldsymbol{V}$ がいずれもランク R であれば，$\boldsymbol{U}\boldsymbol{V}^\top$ もランク R となります．

*2 $\boldsymbol{X}$ に未観測の要素，すなわち欠損値が含まれている場合については 4.5 節で議論します．

4.2.1 目的関数

さて，上の例では $\boldsymbol{U}$ は顧客の嗜好，$\boldsymbol{V}$ は映画のジャンル別成分を表しているとし，またそれらはこちら側で使える情報として与えられていると仮定しました．しかしながら現実にはこのような情報が使えることはまれで，分解を求める (すなわち情報を圧縮する) には $\boldsymbol{U}, \boldsymbol{V}$ は $\boldsymbol{X}$ から求める必要があります．では，どうやって $\boldsymbol{U}$ および $\boldsymbol{V}$ を求めればいいのでしょうか．もちろんでたらめな $\boldsymbol{U}$ や $\boldsymbol{V}$ を求めたいわけではなく，圧縮の精度を高めるため，$\boldsymbol{X}$ をうまく表現できるような $\boldsymbol{U}$ と $\boldsymbol{V}$ を求めることが目的となります．「$\boldsymbol{X}$ をうまく表現する」とはどういうことなのか，さまざまな捉えかたがありますが，1 つの考えかたとしては $\boldsymbol{X}$ と $\boldsymbol{U}\boldsymbol{V}^\top$ の差が小さければ $\boldsymbol{U}\boldsymbol{V}^\top$ は「$\boldsymbol{X}$ をうまく表現している」といえるでしょう．以降，この差を近似誤差 (あるいは単に誤差) と呼び，

$$\boldsymbol{E} = \boldsymbol{X} - \boldsymbol{U}\boldsymbol{V}^\top$$

と表記します．$\boldsymbol{E}$ の各要素が 0 に近ければ近いほど $\boldsymbol{U}\boldsymbol{V}^\top$ が $\boldsymbol{X}$ のよい近似であることを示しています．

しかし，まだ決めるべき問題があります．「誤差 $\boldsymbol{E}$ が小さい」といったとき，$\boldsymbol{E}$ の大小はどのように測ればいいでしょうか．これは一意に決まるものではなく，さまざまな測りかたが考えられますが，機械学習においては絶対誤差と 2 乗誤差がよく用いられます．絶対誤差は全要素の絶対値の和 ($\sum_{i=1}^{I}\sum_{j=1}^{J}|e_{ij}|$) として定義されます．2 乗誤差は全要素の 2 乗の和 ($\|\boldsymbol{E}\|_{\mathrm{Fro}}^2$) として定義されます．絶対値誤差は $\boldsymbol{E}$ の各成分の大きさに比例する形で誤差を計測するため，たとえ $\boldsymbol{E}$ の一部が大きな値をとったとしても全体としてはそれほど大きくなりません．そのため $\boldsymbol{X}$ の一部に大きな値をとる異常値がまぎれこんでいた場合でも影響を受けにくくなります．反対に 2 乗誤差は各成分の大きさの 2 乗で誤差を計測するため，このような異常値の影響を受けやすくなります．一方，2 乗誤差は実数空間全体でなめらかな関数となっているため微分が常に定義され，微分を用いた連続最適化の技法を使うことができます．本章ではこの利点を活かし，2 乗誤差を採用することにします．

これまでの議論により，行列分解は固定した R のもと $\boldsymbol{E}$ の 2 乗誤差が最小となるような $\boldsymbol{U}$ と $\boldsymbol{V}$ を求める問題として定式化されました．これは数式を使って以下のように書くことができます．

$$\min_{\boldsymbol{U},\boldsymbol{V}} \frac{1}{2}\|\boldsymbol{E}\|_{\mathrm{Fro}}^2 = \min_{\boldsymbol{U},\boldsymbol{V}} \frac{1}{2}\|\boldsymbol{X} - \boldsymbol{U}\boldsymbol{V}^\top\|_{\mathrm{Fro}}^2 \tag{4.3}$$

ほかの行列分解と区別するため，本書では問題 (4.3) を**単純行列分解** (**simple matrix decomposition**) と呼ぶことにします．

> **注意**
>
> 式 (4.3) の 2 乗誤差の前に係数 1/2 がついています．これは後に説明する微分の係数をキャンセルするために形式的に導入したものです．1/2 に限らず，0 以上の値であれば 2 乗誤差の係数によって解はいっさい変化しないことに注意してください．

4.2.2 最適化

式 (4.3) で定義された目的関数は連続かつ下に有界であるため，その導関数が 0 となるような $\boldsymbol{U}, \boldsymbol{V}$ をすべて求めれば，そのうち少なくとも 1 つは最適解となります．残念ながら，導関数を 0 としたときの連立方程式は代数的に解くことはできず，以下のような別の方法が必要となります．

特異値分解 本書では詳細を省きますが，問題 (4.3) の解は**特異値分解** (**singular value decomposition**) によって求めることができます．特異値分解は LAPACK(Linear Algebra PACKage)[*3] など，十分に信頼性の高い，効率的な実装が複数提供されています．また解も通常一意に定まります．ただし，特異値分解は式 (4.3) の形を最適化することに特化しているため，ほかの目的関数の解を求めるのには適していません．また $\boldsymbol{X}$ に欠損値が含まれている場合，一般に特異値分解をそのまま適用することはできません．

勾配法による最適化 この方法は，勾配さえ計算できればどのような目的関数にも適用できます．また制約の追加も比較的容易です．さらに確率勾配法 (4.4.4 項) とあわせることで計算効率性を高めることができ，$\boldsymbol{X}$ が大規模な場合に威力を発揮します．一方，R が 2 以上の場合[*4]，解には初期値依存性

*3 http://www.netlib.org/lapack/

*4 $R = 1$ かつ $\boldsymbol{X}$ のランクが 1 以上のとき問題 (4.3) は凸ですが，それ以外の場合は非凸になるためです．凸最適化の一般的なトピックについては文献[31] を参照してください．

があり，毎回同じ解が求まるとは限りません．

4.2.3 類似度としての解釈

冒頭で映画評価 $\boldsymbol{X}$ を圧縮する問題を考えたとき，$\boldsymbol{U}$ は顧客，$\boldsymbol{V}$ は映画の特徴を表すようなもので，ともに使える情報として与えられていた状況を考えていました．一方，行列分解の問題では，$\boldsymbol{X}$ のみが与えられたもとで式 (4.3) を解き $\boldsymbol{U}, \boldsymbol{V}$ を求めます．いったい行列分解は何をやっていることになるのでしょうか．行列分解で得られた $\boldsymbol{U}, \boldsymbol{V}$ はどのような解釈ができるのでしょうか．

より具体的に理解するため，式を追って説明します．式 (4.2) を $\boldsymbol{X}$ の要素ごとに分解すると以下のように書き直せます．

$$x_{ij} = \boldsymbol{u}_{i:}^{\top} \boldsymbol{v}_{j:} + e_{ij} \tag{4.4}$$

ここで簡単のため $\boldsymbol{u}_{i:}$ と $\boldsymbol{v}_{j:}$ のノルムは 1 に正規化されている (すべての $i \in [I], j \in [J]$ について $\|\boldsymbol{u}_{i:}\| = \|\boldsymbol{v}_{j:}\| = 1$) と仮定します．このとき，$\boldsymbol{u}_{i:}^{\top} \boldsymbol{v}_{j:}$ は正規化されたベクトル同士の内積であり，映画 i と顧客 j の**類似度 (similarity)** として解釈できます．すなわち $\boldsymbol{u}_{i:}$ と $\boldsymbol{v}_{j:}$ が同じ方向を向いていたら 1，反対の方向を向いていたら -1 の値をとります．また x_{ij} の値も -1 から 1 の間を取るように正規化されているとします[*5]．この解釈のもと，式 (4.4) は以下のように書き直せます．

$$\text{映画 } i \text{ に対する顧客 } j \text{ の評価} = \text{映画 } i \text{ と顧客 } j \text{ の類似度} + \text{誤差} \tag{4.5}$$

見やすいように映画 i と顧客 j の類似度を $\mathrm{sim}(\boldsymbol{u}_{i:}, \boldsymbol{v}_{j:})$ と書くとすると，式 (4.5) の誤差は $x_{ij} - \mathrm{sim}(\boldsymbol{u}_{i:}, \boldsymbol{v}_{j:})$ なので，「誤差を小さくする」ということは「x_{ij} と $\mathrm{sim}(\boldsymbol{u}_{i:}, \boldsymbol{v}_{j:})$ をできるだけ近くする」ことに対応することがわかります．すなわち，顧客 i が映画 j を高く評価するようであれば $\boldsymbol{u}_{i:}$ と $\boldsymbol{v}_{j:}$ をできるだけ同じ方向になるように，反対に低く評価するようであれば $\boldsymbol{u}_{i:}$ と $\boldsymbol{v}_{j:}$ をできるだけ反対方向になるように決めてやるということです．

このようにして得られた $\boldsymbol{U}$ と $\boldsymbol{V}$ は，ある意味で顧客と映画の特徴を捉えたものだと解釈することができます．例として 4.2 節で登場した「アクション好き，ホラー嫌い」な a さんについてふたたび考えてみましょう．今，映画 j のホラー要素，ロマンス要素，などを R 項目選び，評論家にそれぞれを記入してもらったとし，それが R 次元のベクトル $\boldsymbol{v}_{j:}$ (これを特徴ベクトルと呼びます) で表現されているとします．これをすべての映画について集め，a さんが記入した評価 $\boldsymbol{x}_{a:}$ と合わせて誤差最小化により $\boldsymbol{u}_{a:}$ を求めることを考えましょう．これは数式としては $\mathrm{argmin}_{\boldsymbol{u}_{a:}} \|\boldsymbol{x}_{a:} - \boldsymbol{V}\boldsymbol{u}_{a:}\|^2$ と書けます．このとき求められた R 次元ベクトル $\boldsymbol{u}_{a:}$ は，$\boldsymbol{x}_{a:}$ によって表現される a さんの嗜好性からアクション系映画の特徴ベクトルに近く，またホラー系映画の特徴ベクトルとは異なるような表現が得られることになります．このことから，もし $\boldsymbol{V}$ がうまく映画の特徴を捉えていたなら，$\boldsymbol{U}$ にも顧客の特徴を捉えた表現が求まることがわかります．実際の行列分解では $\boldsymbol{V}$ も同時に推定しますが，基本的には同じように考えることができます．このようにして得られた $\boldsymbol{U}, \boldsymbol{V}$ のことを潜在空間と呼びます．

注意

上記の $\boldsymbol{V}$ のように専門家によって映画の特徴ベクトルが与えられる場合，それらは映画推薦にとても役立つ情報だといえます．たとえばホラー好きの顧客にはホラー要素の高い映画を推薦すれば高い確率で満足してもらえるでしょう．しかしながらこのように細かい属性を得ようとすると専門家への報酬など大きなコストがかかってしまいます．行列分解はこのような特徴ベクトルを映画評価から逆推定する方法だと捉えることができます．

4.3 さまざまな行列分解

式 (4.3) は行列分解の最も基本的な形です．これだけだとやや単純過ぎるため，実際にはデータの特徴に合わせて拡張した手法が使われることがあります．その中でもよく使われるものを紹介します．

4.3.1 ℓ^2 正則化行列分解

単純行列分解 (4.3) では最小化の対象として誤差のみに着目していました．

*5 1 点から 5 点までの 5 段階評価である映画評価の場合，x_{ij} を $2(x_{ij}/5) - 1$ のように変換すればこの条件を満たします．