

卒業論文

データマイニングを用いた

リスナー嗜好に基づく楽曲推薦システムの構築

東海林研究室

5年 情報工学科

出席番号 19 番 小林利彰

目次

第1章 序論	2
1.1. はじめに	2
第2章 情報フィルタリングについて	3
2.1. コンテンツベースフィルタリング	3
2.2. 数量化 I 類の概要	4
2.3. 協調フィルタリングの概要	7
第3章 構築システムの機能と構造	10
3.1. 提案手法	10
3.2. 印象のサンプリング	10
3.3. 数量化 I 類の実装	12
3.4. 協調フィルタリングの実装	13
3.5. Leave-one-out cross validation 法	15
第4章 数値実験	16
4.1. 目的	16
4.2. 手順	16
4.3. 使用楽曲	17
4.4. 結果	17
4.5. 考察	18
第5章 結論	19
付録	20
参考文献	34

第1章 序論

1.1. はじめに

近年、デジタル情報の通信分野での技術革新はめざましいもので、主にインターネットや携帯電話を通して我々もその情報や技術を生活の中で活用している。例えば、ニュース、音楽、映画、ゲーム、画像などの様々なコンテンツが提供されており、既に我々の身近なものとなっている。

本研究に直接関係のある音楽について見てみると、パソコンやポータブルプレーヤーの普及に伴って iTunes Store や mora などの様々な楽曲配信サービスが利用されている。iTunes Store は 2007 年 7 月までに楽曲ダウンロード件数が 30 億曲を突破したことが発表されている[1]。このように、楽曲数や利用件数から見ても楽曲配信サービスは新たな市場として成立している。また、2006 年 10 月からは日本で初である定額制の配信サービスを Napster Japan が開始したこともあり、これからますますその需要は拡大し、それに伴い配信楽曲数も増加すると考えられる[2]。

しかし、このような楽曲情報の急激な肥大化により、リスナーが自らの嗜好に合致した楽曲を探し出すことは困難となっている。それは、ある楽曲を検索する場合、リスナーがその楽曲のキーワードを持ち合わせていないと検索できないという問題があるためである。このような状況を解決するための 1 つの方法として情報フィルタリングシステムがあり、現在盛んに研究が行われている[3][4][5][6]。

情報フィルタリングシステムとは、膨大な情報の中からユーザの好みや興味に合致する情報をユーザに提供するシステムであり、楽曲推薦だけでなく、映画やブログなど様々な分野に応用されている。この情報フィルタリングシステムは、コンテンツベースフィルタリングシステムと協調フィルタリングシステムの 2 つの手法に大別できる[3][5]。コンテンツベースフィルタリングシステムと協調フィルタリングシステムがどのようなものであるのかという基本的な概要については 2 章で詳しく述べることにする。

協調フィルタリングシステムには、楽曲の内容や特徴を参照することなく推薦を行うことが出来るという大きな利点がある。しかし、システムの初期段階では楽曲に対するリスナーの評価数が少なく有用な推薦が出来ないという問題点がある。さらに、誰も評価していない楽曲は、もしリスナーの嗜好に適合しているとしても推薦することが出来ないという問題もある。

そこで、本研究ではこの問題に対処するためにデータマイニング手法の 1 つである数量化 I 類を用いて楽曲推薦システムを構築することを提案する。数量化 I 類を用いることで、利用するリスナー数に依存することなく、アーティストの性別や楽曲のテンポ、明るい曲か暗い曲か、などの楽曲特徴とその評価結果の因果関係を明示的に分析し、個々のリスナーの嗜好に合った楽曲を推薦することができる[8][9][10]。

さらに、ショッピングサイトなどの実用面、研究面などで成功を収めている協調フィルタリングを用いたシステムと比較し、その評価を行う。

第2章 情報フィルタリングについて

本章では、我々が構築したシステムに適用したコンテンツベースフィルタリングの1つである数量化I類と、従来から用いられている協調フィルタリングについて説明する。

2.1. コンテンツベースフィルタリング

コンテンツベースフィルタリング方式は、過去にリスナーが聴取して高評価だった楽曲の特徴を分析することでリスナーの嗜好傾向を示したプロファイルを生成し、そのプロファイルに類似した楽曲を推薦する方式である[3][4]。

リスナープロファイルはリスナーの関心を表現するキーワードベクトルからなり、このベクトルと、各楽曲内容を表すキーワードベクトルとの類似度を算出することで楽曲を選別する。

以下に簡単なコンテンツベースフィルタリングの一例をあげる。下の表1はあるリスナーが、楽曲配信サービスを利用して過去に購入した楽曲を表しているとする。このリストを見るだけで、「この顧客は他のジャンルよりはロックに興味があり、Aというアーティストが好きである。その次にBというアーティストが好きで、どちらかというと言楽をよく聞くようだ・・・」などと推測することができる。

表1 あるリスナーの聴取履歴

楽曲	ジャンル	アーティスト	洋or邦
1	ロック	A	洋楽
2	ロック	A	洋楽
3	ポップス	B	邦楽
4	ロック	B	邦楽
5	ポップス	A	洋楽
6	クラシック	C	洋楽

このような推測に基づいて、このリスナーにとって“ジャンル”・“アーティスト”・“洋楽 or 邦楽”のそれぞれがどれほど価値を持っているかをスコア化してプロファイルを作成し、スコアの高い楽曲を推薦する。図1にこのシステムの概要を示す。

この方式には問題点が2つある。その1つ目は、リスナーが過去に高い評価をした楽曲に類似した情報ばかりが推薦されてしまうため、リスナーの嗜好変化に追従することが困難なことである。よって、リスナーに新たな嗜好の発見を促す作用が働かない閉鎖的なシステムになってしまう場合がある。

問題点の2つ目は、既存技術において有効な楽曲特徴を抽出できないといったことである。本研究においても、個々のリスナーの嗜好に適合するような楽曲の指標は定めることが大変難しいために、同じ問題を含んでいる。

その一方で、コンテンツベースフィルタリングシステムは協調フィルタリングシステムに比べて、システムを使い始めたばかりのリスナーに対しても楽曲推薦が可能であるというメリットがある。そこで両者の長所を活かしたハイブリッド型フィルタリングを使った推薦システムも登場している[3]。

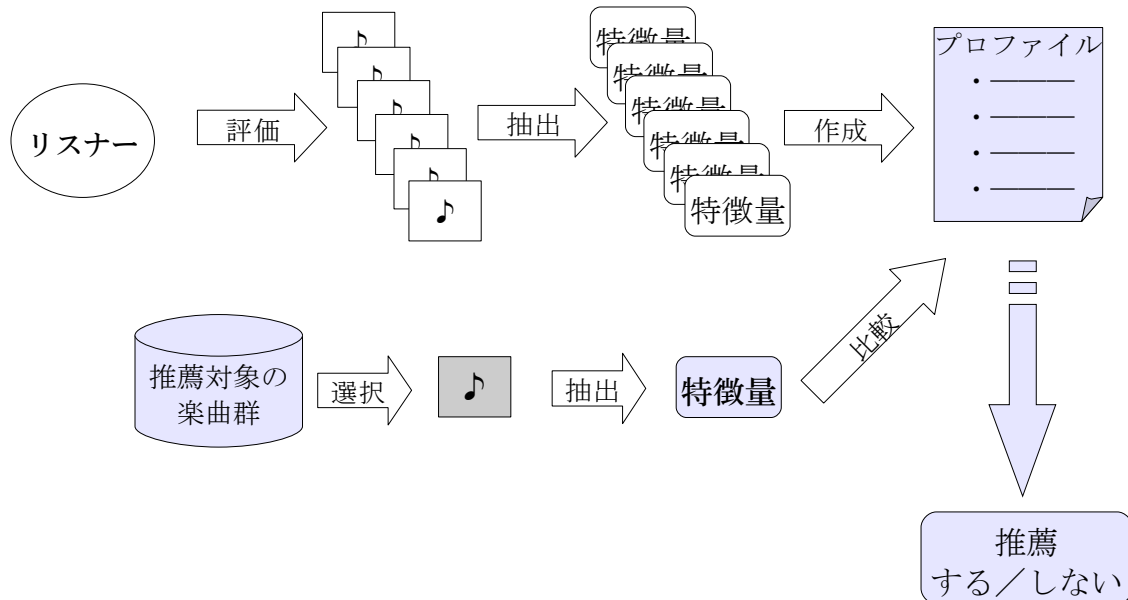


図1 コンテンツベースフィルタリングシステム概要

2.2. 数量化 I 類の概要

数量化 I 類とは、大量にあるデータの中から、規則性のあるルールなどを発見するデータマイニング (Data mining) における 1 つの手法である[9][10]。本研究では、楽曲推薦のためのコンテンツベースフィルタリングシステムを数量化 I 類を用いて構築する。

楽曲推薦において、音楽的特徴を我々がいくら考えても、その楽曲の嗜好との関係を見出すことは非常に複雑で困難である。しかし、このデータマイニングは、有用なデータと無用なデータの分析を行い、それによって分類や推定、予測が可能である[13][14]。以下では、数量化理論と数量化 I 類についての説明をする。

- 数量化理論

数量化理論は、日本独自の多次元データ分析法である。数量化理論には I 類、II 類、III 類、IV 類、V 類、VI 類までの 6 つの方法があるが、現在、I 類から IV 類までがよく知られている。日本国内で開発され、普及したが、海外においても本質的に同種の手法が提唱されていたものも少なくない[10]。

多変量解析の「重回帰分析」「判別分析」「主成分分析」等は、量的データを扱う分析方法であるが、いつも量的データの形で標本を得られるとは限らない。そこで、得られた質的データに適切な数量を与え、質的データを数量化することにより多変量解析が行えるようにすることを「数量化」という。統計で扱うデータには、「量的データ」と「質的データ」があり、量的データは数値の間隔に意味を持つデータであるが、質的データはその間隔に意味はもたなく他のものと区別したり、順序を示したりするデータである。

量的データ { 間隔尺度…絶対原点を持たない尺度データ
 比例尺度…絶対原点を持つ尺度データ

質的データ { 名義尺度…分類を示すデータで大小関係に意味は持たない
 順序尺度…順序関係の大小に意味を持つデータ

● 数量化 I 類

数量化 I 類は、量的データを分析する重回帰分析に対応する質的データを分析する方法である[8][9]。重回帰分析では、量的データである説明変量から、量的データである目的変量を予測する。これに対し数量化 I 類では、アイテム・カテゴリと呼ばれる質的データを得て、この値から量的データである外的基準（目的変量に対応する）を得る方法である。それを表で表したものが下の表 2 である。

表 2 アイテムと外的基準

標本No.	外的基準	アイテム	
		カテゴリ1	カテゴリ2
1	y_1	x_{11}	x_{21}
2	y_2	x_{12}	x_{22}
3	y_3	x_{13}	x_{23}
n	y_n	x_{1n}	x_{2n}

アイテム（項目）とは質問事項のようなものであり、カテゴリとはその回答のようなものである。アイテムは、例えば「貴方は数学が好きですか？」のようにアンケートの質問事項のように与えられ、カテゴリは「はい/いいえ」のように分類で与えられる。この質的データであるカテゴリから外的基準と呼ばれる量的データを予測するのが数量化 I 類という分析方法である。

例えば外的基準を英語のテスト（10点満点）の点数、アイテムとカテゴリをそれぞれ数学と物理の好き・嫌いとする、表 3 のように表せる。

表3 英語の点数とアイテムの関係

標本No.	英語点数	数学		物理	
		好き	嫌い	好き	嫌い
1	2	0	1	0	1
2	4	0	1	1	0
3	5	1	0	1	0
4	7	1	0	0	1
5	8	1	0	1	0
6	6	0	1	1	0
7	5	1	0	0	1
8	3	0	1	0	1

普通、好きか・嫌いか、ということは1と0では考えないが、このままでは計算することができないので、表2では「該当有り…1」「該当無し…0」と置き換えている。この置き換える変数のことを「ダミー変数」と言う。

ここで予測式Yを

$$Y = a_{11} * x_{11} + a_{12} * x_{12} + a_{21} * x_{21} + a_{22} * x_{22} \text{ とする。}$$

この式にそれぞれの標本のダミー変数を代入し得られた予測値 Y_i を用いて、最小二乗法によって正規方程式を得ることが出来る。

上の例だと最終的な予測式は

$$Y = 7 - 2.5 * x_{12} - 1.5 * x_{22} \text{ となる。}$$

この式を見ればわかるとおり、英語の得点は物理の好き嫌いよりも、数学が好きであるということに、より大きく影響を受けている。

この式を用いて推定した結果を下の表4に示す。

表4 推定した英語点数

推定点数
3
4.5
7
5.5
7
4.5
5.5
3

2.3. 協調フィルタリングの概要

協調フィルタリングとは、多くのリスナーの嗜好情報を蓄積し、あるリスナーと嗜好の類似した他のリスナーの情報を用いて自動的に推論を行う手法である [11][12]。リスナー A と関心が類似した別のリスナー B が高く評価した情報を、リスナー A にも薦めるというものであり、ロコミの原理に例えられることも多い。具体的には、リスナーの各楽曲に対する評価値を基に、リスナー間の類似度を計算し、類似度の高いリスナーの評価情報を優先的に参照し推薦を行う。

この方式では、コンテンツベースフィルタリングのように推薦内容が収束してしまう問題もなく、また、情報自体がどのような特性を持っているかに言及しないため、映画や音楽、ニュースやブログなどといったどのような情報にも適用できる利点がある。実際には、Amazon.com のような書籍販売サイト、飲食店ガイドページのように幅広い分野のサービスにおいて活用されている。

- 協調フィルタリングの最近の動向

協調フィルタリングにはリスナーの評価付けによる明示的なものと、システムの操作履歴（例えばブラウザの閲覧履歴）などを利用した暗黙的なものがある。3.3.で詳しく述べる GroupLens のような方式の協調フィルタリングが有効に働くには、リスナーの嗜好を示す大量のデータが必要である。自分の嗜好情報を数多く入力しても、比較するリスナーがいなければ意味がない。また、たとえリスナーの数が多くても、嗜好の比較が可能な人がいなければ推薦精度は極端に低くなる。

さらに言えば、最初に多くの対象に点数をつける作業は面倒であり、新しい楽曲が出てくるたびにいちいち評価する作業は長続きするとは思えない。そもそも、自分があまり好きではない楽曲を評価し、わざわざ悪い点をつけるようなリスナーは少ないと考えられる。従って、低い評価と無評価の区別がつかない可能性もある。このような理由のためか、得点づけが必要な協調フィルタリングは最近ではそれほど普及していない。

一方、明示的に採点処理をおこなう必要のない協調フィルタリング手法もある。たとえば、Amazon.com で本を購入すると、“この本を買った人はこういう本も買っています” というお薦めの本が表示される。推薦される本は、利用者の購入履歴や本などのジャンルに基づいて選ばれていると思われるが、利用者は本の購入以外に特別な作業をしていないのに、それなりに適切な本がリストアップされる。

GroupLens などの明示的な評価作業が必要な協調フィルタリングシステムよりも Amazon.com のように暗黙的に評価がおこなわれる方式のほうが、普及する可能性が高い。Amazon.com は、蓄積された膨大な情報と、独自に開発した協調フィルタリングシステムのアルゴリズムによって、本を購入したという単純な情報をもとに、有用なフィルタリングが行うことが可能だと考えられる。

- 協調フィルタリング共通の問題点

- コールドスタート問題

協調フィルタリングを用いた推薦システムで有効な推薦を行うには、数多くのリスナーの参加が不可欠である。しかしながら、推薦システムの初期段階では、リスナー数の少なさから類似リスナーが発見できず、有効な推薦が期待できないという問題が発生する。

- リスナーの信頼性の問題

リスナー間の類似度を求める際に、各情報に対する評価の類似度のみを用い、リスナーの信頼度は考慮されていない。このため、悪意あるリスナーが推薦システムを利用した場合、その推薦アルゴリズムを逆手に取り、システムの推薦精度を落とすように行動することが可能である。

- 従来の協調フィルタリングシステム[11]

1. Tapestry

Tapestry は e-mail、Netnews のフィルタリングを行なうシステムである。このシステムは協調フィルタリングの概念を初めて示した。システムはキーワードを用いて情報をフィルタリングする他、同じ関心を持つユーザ名を指定してフィルタリングする。これは自分と関心の同じユーザは、自分と同じように情報を評価する、という仮定に基づいている。システムは Netnews の記事とそれに対する評価(リプライ、コメント)を蓄積する。ユーザからの要求に応じてシステムはこれらの評価をフィルタリングに利用する。

2. GroupLens

GroupLens は Netnews を対象とした協調フィルタリングシステムである。ユーザの評価を得点で表すこと、集めた得点を元に個人の得点を予想することが特徴である。ユーザは記事を読み、それに対して5段階の得点を与える。システムは記事と各ユーザの得点を収集する。収集した得点の集計結果を元に個々のユーザの得点を計算する。

3. Ringo

Ringo は音楽の推薦を行なう協調フィルタリングシステムである。GroupLens と同様、ユーザからの得点を元に個々のユーザに応じた得点予想を行なう。

4. Fab

従来の協調フィルタリングシステムには、「誰かが情報を評価しなければならない」「多くの評価を必要とする」という問題点がある。これらは GroupLens、Ringo に共通する問題である。Fab は協調フィルタリングと内容に基づくフィルタリングの構成によるハイブリッドなシステム構成を採用し、双方の特徴を持つことにより上の問題を解決した。

5. PHOAKS

PHOAKS は URL 推薦システムである。GroupLens や Ringo のような得点に基づくフィルタリングと異なり、は自動的に Netnews から推薦情報(URL)を獲得する。PHOAKS は Netnews からキーワードと URL を取得する。対象となる記事は URL が記載されている記事および FAQ である。URL 抽出の方針は以下の通りである。

- i. クロスポストされた記事に記載された URL の除外
- ii. 署名に記載された URL の除外
- iii. 前の記事のリプライ部分に記載された URL の除外
- iv. テキストのブロックで囲まれた URL を取得

第3章 構築システムの機能と構造

3.1. 提案手法

本研究では、個々のリスナーの嗜好に基づいて楽曲を推薦するシステムを提案する。その概要としては、複数名の被験者の嗜好を抽出し、それを数量化 I 類を用いて構築したシステムに通すことで、リスナーに適した楽曲を推薦する。また提案システムと比較するため、同じ被験者の印象と楽曲を用いて協調フィルタリングシステムで楽曲推薦を行う。

被験者の嗜好を抽出する際には考慮しなければならない事柄があり、そのため我々は抽出方法を工夫することでその問題を解決した。それについては、3.2.で説明する。数量化 I 類と協調フィルタリングシステム数量化 I 類の実装については、それぞれ 3.3.と 3.4.で述べる。また、結果の比較方法については 3.5.で説明する。

3.2. 印象のサンプリング

楽曲推薦に限らず、何らかの情報をユーザに推薦するシステムの場合、ユーザの好みである嗜好情報を抽出し、それをシステムに通すことでそのユーザに推薦すべき情報を与えることができる。ここで問題となるのが、嗜好情報の抽出をどのようにするかということである。

楽曲推薦の場合、リスナーがある楽曲を聴取するたびにその印象を的確に記述してもらえるのであればそれに越したことはない。しかし、そのようなシステムだといちいち印象評価を求められることになり、リスナーの負担は大きく実用的であるとは言えない。

逆に、リスナーに評価を求めずに、システムが完全に自動的に嗜好情報を抽出する場合、リスナーの負担はほとんど無い。しかし、リスナーが直接に印象評価を返す上のシステムと比べ、システム側で勝手に判断した嗜好情報では推薦精度の面において劣る結果になると考えられる。

そこで今回は、リスナーの負担を出来るだけ抑え、なおかつ高い精度の評価を見込めるように、以下のような方法でサンプリングを行った。

- 準備段階
 - サンプリングに使用する全ての楽曲を聞く
 - その曲の中で特に特徴的（印象的）な部分やサビを 1 曲につき約 15 秒で切り取る
 - Web で評価できる環境を整える

- サンプルリング

- 準備した楽曲を、ランダムで再生する
- リスナーは1曲聴くたびに5段階で印象を評価する

このように、リスナーは曲の全てを聴取するわけではなく、1曲につきその曲の特徴が大きく現れるような約15秒を聴取する。さらに、自らの評価を難しい音楽的専門用語などを用いて表すのではなく、簡単な5段階評価とすることで、リスナーの手間を大きく減少させることができる[15]。

3.3. 数量化 I 類の実装

数量化 I 類の概要については 2.3. で述べたが、実際にアイテムとカテゴリをどのように分類し、ダミー変数を割り当てるかという問題がある。楽曲の特徴は、明るい・ノリが良い・癒される・静か・悲しい・早い、など様々な言葉で表すことが出来るが、それを数量化するとなると選択できるアイテムは限られてくる。このアイテムの選択は慎重に行う必要がある。

今回、数量化 I 類で使用したアイテム・カテゴリと、ある被験者の評価値の一部が下の表 5 である。

表 5 使用したアイテムとある被験者データの一部

曲	性別		言語		テンポ			明暗			年齢			印象 Y
	男	女	邦	洋	遅	—	早	暗	—	明	若	—	老	
	a11	a12	a21	a22	a31	a32	a33	a41	a42	a43	a51	a52	a53	
1	1	0	1	0	0	0	1	0	1	0	0	1	0	3
2	0	1	0	1	1	0	0	0	1	0	0	0	1	3
3	0	1	1	0	0	1	0	1	0	0	0	1	0	3
4	1	0	1	0	1	0	0	1	0	0	0	0	1	2
5	0	1	1	0	0	0	1	0	1	0	1	0	0	1
6	1	0	1	0	0	1	0	0	0	1	0	0	1	2
7	0	1	1	0	0	1	0	0	0	1	0	1	0	2
8	0	1	1	0	0	1	0	0	0	1	1	0	0	1
9	1	0	1	0	1	0	0	1	0	0	0	0	1	3
10	0	1	1	0	0	0	1	0	0	1	1	0	0	2
11	1	0	1	0	1	0	0	1	0	0	0	0	1	1
12	1	0	1	0	0	1	0	0	1	0	0	0	1	4
13	0	1	1	0	0	1	0	0	1	0	1	0	0	2
14	0	1	1	0	0	1	0	0	0	1	1	0	0	1
15	1	0	1	0	0	0	1	0	1	0	0	0	1	4

性別と言語は 2 カテゴリにしか分けられないが、それ以外のテンポ、明暗、年齢は 3 カテゴリに分類した。このようなデータを元に、2.2. で述べたようなアルゴリズムを用いて、個々のリスナーの印象評価を推定する。

尚、それぞれのアイテムの値は、我々が楽曲を 1 曲 1 曲聴いてみて主観的に決定したものであり、全てのリスナーがこのように判断するとは限らない。従って、客観的にアイテムの値を決定する方法を考案しなければならない問題がある。

3.4. 協調フィルタリングの実装

協調フィルタリングのアルゴリズムは既に幾つか考えられているが、今回はその先駆的なシステムであり過去に Amazon.com に採用されていた GroupLens を実装する[16]。

GroupLens は、2.2.で説明した協調フィルタリングの概要にほぼ当てはまり、リスナーは最初に幾つかの楽曲に対する評価値を入力する。この結果を他のリスナーの評価と比較することにより、他のリスナーと自分の嗜好情報の類似度を計算する。自分がまだ聴いていない楽曲の評価を知りたい場合には、他人の評価を類似度で重み付けして足し合わせ、その楽曲を自分がどう評価するのかを予測する。

例えば、6曲の楽曲について、A、B、C、Dの4人のリスナーが表6のように5段階の印象評価点を付けたとする。2番の楽曲を見ると、Aは5点を付けているが、Bは2点しか付けていない。もちろん全員が全ての楽曲を聴くわけではないので、空欄の部分もある。

表6 楽曲番号とリスナーの評価

楽曲番号	A	B	C	D
1	1	4	2	2
2	5	2	4	4
3			3	
4	2	5		5
5	4	1		1
6	?	2	5	?

ここで、評価値ベクトルの相関係数をリスナーの類似度として利用する。AとBはともに楽曲1、2、4、5、に対して対して評価を行っており、Aの評価値ベクトルは(1,5,2,4)、Bの評価値ベクトルは(4,2,5,1)となるが、これらの相関係数 r_{AB} は以下のようにして計算できる。ここで、 σ_A 、 σ_B は標準偏差、 \bar{A} 、 \bar{B} は評価値の平均、 $Cov(A, B)$ は共分散である。Aの評価値の平均は $\bar{A} = 3$ 、Bの評価値の平均は $\bar{B} = 3$ である。

$$r_{AB} = \frac{Cov(A, B)}{(\sigma_A * \sigma_B)}$$
$$= -0.8$$

AとBの評価には、大きな負の相関があることが分かる。一方、AとCの類似度は1、AとDの類似度は0となる。

Aが6番の楽曲をどう評価するかを予測したいときは、A以外のリスナーによる評価値に類似度の重みをつけて足し合わせて算出する。ここで、Jは6番の楽曲を評価しているリスナーの評価である。Aと類似度が高いCが5点を付けているため、算出結果は以下のように4.56という大きな予測値となる。

$$\begin{aligned} A_6 \text{ 予測値} &= \bar{A} + \frac{\sum_{J \in \text{評価者}} (J_6 - \bar{J}) r_{AB}}{\sum_J |r_{AB}|} \\ &= 4.56 \end{aligned}$$

このように予測値を算出し、実際のリスナーの評価と比較することで、精度を求める。

3.5. Leave-one-out cross validation 法

数量化 I 類と協調フィルタリングによる楽曲推薦システムの性能比較のために cross validation 法を用いる。

cross validation 法とは交差検定法とも呼ばれ、重回帰分析などの応用に用いられている検定法の 1 種である。具体的には、サンプル集合 X から平均二乗誤差を推定する方法であり、あるサンプルを用いて学習と実験を行いたい場合に、サンプル量が少なくて偏りが予想されるときに使う。

例えば、4-fold cross validation 法（4 分割交差検定法）であれば、サンプルをランダムに 4 等分する。等分したそれぞれを A、B、C、D とすると、次のように実験を行う。

(学習データ, 推定データ)

(ABC, D)

(ABD, C)

(ACD, B)

(BCD, A)

これによって得られた推定値と真値とを比べ、その誤差の平均を取れば、平均二乗誤差を求めることができ、偏りを防ぐことが可能である。

そして、本研究で利用する Leave-one-out cross validation 法は、その中でも全サンプルから、1つのサンプルを除き、残りのサンプルでモデルを作製したときに、あらかじめ抜いていた1つのサンプルを正確に予測することができるかを検定する方法である。

第4章 数値実験

4.1. 目的

この実験は、数量化 I 類を用いて構築したシステムが、楽曲の特徴に依存せずに推薦値を求めることが出来る協調フィルタリングと比較して、どの程度実用的なものかということを実証するという目的で行う。

4.2. 手順

被験者は楽曲の印象を 3.2. で述べたように 5 段階で評価する。そのデータを用いて数量化 I 類により被験者が評価した全ての楽曲の評価点に対する推定値を求める。その後、Leave-one-out cross validation 法により真の評価点と推定値との間の誤差を算出する。

同様に、協調フィルタリングシステムでも同じ被験者、同じ楽曲を用いて推定値を求め、その誤差を算出し、2つのシステムを比較し、精度を評価する。

手順 1. サンプルング

- 場所
東海林研究室
- 機材
Web ブラウザが使用できる PC
ヘッドホンやイヤホン
- 被験者
東海林研究室の 4・5 学年の学生（11 名）と指導教官（1 名）
他 2 名の計 14 名
- 楽曲
100 曲（使用楽曲については 4.3. で述べる。）
- 抽出するデータ
100 曲の楽曲に対する印象の 5 段階評価

手順 2. システムによる推定値の算出

数量化 I 類と協調フィルタリングシステムにより評価点の推定値を求める。推定する楽曲は、後に Leave-one-out cross validation 法によって検定することを考え、全楽曲から 1 曲ずつ取り出して推定していく。

手順 3. 数量化 I 類と協調フィルタリングのシステム比較

Leave-one-out cross validation 法によって求めた平均二乗誤差を利用して、推定結果を比較する。

4.3. 使用楽曲

本研究のような楽曲に関する研究の場合、最も重要な課題は使用する楽曲群の内容とその著作権である。そのため本研究ではRWC研究用音楽データベースを使用した[17]。RWC研究用音楽データベース (DB) は、研究者が研究目的に利用する上で、共通利用の自由、学術利用の自由が確保された音楽情報処理研究用DBである。技術研究組合新情報処理開発機構 (RWCP: Real World Computing Partnership) RWC音楽DBサブワーキンググループにより構築され、産業技術総合研究所 RWC音楽データベース管理責任者によって、研究者へ実費配布 (実質上、無償配布) されている。

このRWC音楽データベースは曲調・テンポ・曲の長さが多様なだけでなく、作曲家・作詞家・アーティスト・楽器演奏者の人数も豊富である。また、著作権の問題も無いために本研究に適している。

その中でも今回の実験で使用したのはポピュラー音楽データベースである[18]。ポピュラー音楽データベースはJ-pop (80曲) と洋楽 (20曲) で構成されている。この100曲を3.2.で述べた方法で効率よく嗜好情報を抽出できるように加工した。

4.4. 結果

数量化I類と協調フィルタリングのシステムを比較するために算出したLeave-one-out cross validation法による平均二乗誤差が、下の表7と表8である。

表7 数量化I類の誤差

被験者	平均自乗誤差 σ
A	1.364203
B	1.081989
C	1.114735
D	0.728225
E	1.312116
F	1.195053
G	0.907715
H	1.152816
I	1.363723
J	1.044326
K	1.056220
L	1.189002
M	1.070183
N	1.247100

平均：1.130529

表8 協調フィルタリングの誤差

被験者	平均自乗誤差 σ
A	1.299533
B	1.032097
C	1.272752
D	0.790691
E	1.203401
F	1.315604
G	0.908718
H	1.266828
I	1.131037
J	1.145432
K	0.968091
L	1.084917
M	0.987954
N	1.034090

平均：1.102939

2つの結果を比較すると、誤差の平均の差はとても小さい。この結果に対しての考察を4.5.でまとめる。

4.5. 考察

上の結果から数量化 I 類のシステムを評価する。表を見れば分かるように、二つのシステムの誤差はほぼ同じである。従って、システムを利用するリスナーに依存しない数量化 I 類のシステムでも、協調フィルタリングと同様の性能を得ることが出来たと言える。さらに、5段階評価の約1段階の誤差と考えるとそれほど悪くもないように思える。

先に述べたように協調フィルタリングシステムはある程度リスナー数を確保しなければ精度のよい推薦値を出してはくれない。仮に今回の数値実験において、10人未満の被験者しか確保できなかったとしたらその結果はより低いものになってしまうと考えられる。逆に、もっと多数の被験者を用いて数値実験を行えば、より精度の高い推薦値を得られるだろう。

数量化 I 類について考えてみると、アイテムの選択が大きな鍵である。今回の数値実験では {性別, 言語, テンポ, 明暗, 年齢} のアイテムをそれぞれ {2,2,3,3,3} のカテゴリに分割した。しかし、{性別, 言語, テンポ, 明暗} のアイテムしか使わず、さらにそれぞれ2つのカテゴリにしか分割しないと、精度は低くなってしまふ。このように楽曲推薦におけるアイテムとカテゴリをどのように定めると最適かということは今後実験を重ねていかなければ未知である。アイテムの総数をもっと増やし、個々のリスナーの嗜好にもっとも大きく影響を与えるアイテム群を見つけ出し、それを使って推定すればより良い精度が得られると考えられる。

第5章 結論

本研究では、データマイニング手法の1つである数量化I類を用いることで、楽曲の特徴と評価結果の因果関係を明示的に分析し、個々のリスナーの嗜好に合った楽曲を推薦するシステムを構成した。また、楽曲の特徴に関わらず推薦をすることができる協調フィルタリングシステムを本システムと比較する目的で構築した。2つのシステムを **Leave-one-out cross validation** 法によって比較することにより、その推薦精度を評価した。その結果、両システムの推薦精度は同様の値となり、今回の数値実験の内容からは本システムは協調フィルタリングと同様の性能を持つことを確認できた。ただし、被験者の人数や楽曲数などを変化させたり、サンプリング方法、数量化I類におけるアイテムの選出を工夫することで、より良い結果が得られると考えられる。

今後の課題としては、数量化I類のアイテムの総数を増やすために、より適切なものを選定する。そうすることで、リスナーの嗜好に関係する楽曲特徴を幅広く拾い上げることができ、本システムの精度をより高めることが可能だと思われる。そして、個々のリスナーの嗜好に基づいた精度の高い楽曲推薦システムを構築する。

付録

以下に数量化 I 類と協調フィルタリングを用いたシステムのソースを載せる。

- 数量化 I 類の楽曲推薦システム

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#define      sqr(x)      ((x)*(x))

//行列の作成を行う関数
double** make_mat( int rows, int cols )
{
    double **a; int c=1;

    a=(double **)malloc(sizeof(double*)*rows);
    *a=(double *)malloc(sizeof(double)*rows*cols);

    for(int i=1;i<rows;++i) a[ i ]=a[ 0 ]+cols*i;

    return a;
}

//行列の解放を行う関数
void free_mat( double** mat )
{
    free(*mat);
    free(mat);
}

//行列の表示を行う関数
void print_mat( double** A, int rows, int cols )
{
    for(int i=0;i<rows;i++)
    {
        for(int j=0;j<cols;j++)
        {
            printf("%8.5lf ",A[i][j]);
        }
        printf("\n");
    }
}
```

//行列のコピー($B=A$)を行う関数

```
void copy_mat( double** A, double** B, int rows,int cols )
{
    memmove(*B,*A,sizeof(double)*rows*cols);
}
```

//行列の部分コピー($B=A[r_from \sim r_to][c_from \sim c_to]$)を行う関数

```
void copy_submat( double** A, double** B, int r_from, int r_to, int c_from, int c_to )
{
    for(int i=r_from; i<=r_to; i++)
    {
        for(int j=c_from; j<=c_to; j++)
        {
            B[abs(r_from - i)][abs(c_from - j)] = A[i][j];
        }
    }
}
```

//卒研用。これでDを作成

```
void copy_submat2( double** A, double** B,
                  int r_from, int r_to, int c_from, int c_to )
{
    for(int i=r_from; i<=r_to; i++)
    {
        B[abs(r_from - i)][0] = 1.0;
        for(int j=c_from; j<=c_to; j++)
        {
            B[abs(r_from - i)][abs(c_from - j) + 1] = A[i][j];
        }
    }
}
```

//転置行列($B=A^T$)を求める関数

```
void t_mat( double** A, double** B, int rows, int cols )
{
    for(int i=0;i<rows;i++)
    {
        for(int j=0;j<cols;j++)
            B[j][i]=A[i][j];
    }
}
```

//行列積($C=A*B$)を求める関数

```
void prod_mat( double** A, double** B, double** C,
              int rows_A, int cols_A, int rows_B, int cols_B )
{
    for(int i=0;i<rows_A;i++)
```

```

    {
        for(int j=0;j<cols_B;j++)
        {
            C[i][j]=0;
            for(int k=0;k<cols_A;k++)
                C[i][j]+=A[i][k]*B[k][j];
        }
    }
}

```

```

void lu_decomp( double** src, double** dst, int* pivot, int rows, int cols )
{
    int r;
    // src -> dst に内容をコピー
    for( int r = 0; r < rows; ++r ){
        for( int c = 0; c < cols; ++c )
            dst[ r ][ c ] = src[ r ][ c ];
    }

    for( r=0; r < rows; ++r ) pivot[ r ] = r;

    int max_loop;
    if( rows > cols ) max_loop = rows;
    else max_loop = cols;

    for( int loop = 0; loop < max_loop; ++loop ){
        // pivot 計算
        double maxval = dst[ pivot[ loop ] ][ loop ];
        int maxrow = loop;
        for( int r = loop + 1 ; r < rows; ++r ){
            double tmpval = dst[ pivot[ r ] ][ loop ];
            if( tmpval > maxval ){
                maxval = tmpval;
                maxrow = r;
            }
        }

        int tmprow = pivot[ loop ];
        pivot[ loop ] = pivot[ maxrow ];
        pivot[ maxrow ] = tmprow;

        // L 成分計算
        // loop 列の各行を左上の値で割りながらコピー
        for( r = loop + 1 ; r < rows; ++r ){
            dst[ pivot[ r ] ][ loop ] /= dst[ pivot[ loop ] ][ loop ];
        }

        // 残差行列を計算

```

```

        for( r = loop + 1; r < rows; ++r ){
            for( int c = loop + 1; c < cols; ++c ){
                dst[ pivot[ r ] ][ c ] -= dst[ pivot[ r ] ][ loop ] * dst[ pivot[ loop ] ][ c ];
            }
        }
    }
}

```

```

void linear_equation( double** A, double* x, double* y, int N )
{
    double **LU;
    int *pivot;
    int row;
    double *z;

    LU = make_mat( N, N );
    pivot = ( int* )malloc( sizeof( int ) * N );
    z = ( double* )malloc( sizeof( double ) * N );
    // A -> LU に分解
    lu_decomp( A, LU, pivot, N, N );

    // Ax = LUx = y で Ux = z とおいて
    // Lz = y
    // Ux = z を順に解く
    for( int row = 0; row < N; ++row ){
        z[ row ] = y[ pivot[ row ] ]; // y は pivot で入れ替えるのに注意
        for( int col = 0; col < row; ++col )
            z[ row ] -= LU[ pivot[ row ] ][ col ] * z[ col ];
    }

    for( row = N-1; row >= 0; --row ){
        x[ row ] = z[ row ];
        for( int col = row + 1; col < N; ++col )
            x[ row ] -= LU[ pivot[ row ] ][ col ] * x[ col ];
        x[ row ] /= LU[ pivot[ row ] ][ row ];
    }

    free_mat( LU );
    free( pivot );
    free( z );
}

```

//N 次正方形の逆行列($B = A^{-1}$) を求める関数

```
void inverse_mat( double** A, double** B, int N )
```

```
{
    double *x,*y;

    x=(double*)malloc(sizeof(double)*N);
    y=(double*)malloc(sizeof(double)*N);

    for(int i=0;i<N;i++)
    {
        for(int j=0;j<N;j++)
        {
            if(i==j) y[j]=1.0;
        else    y[j]=0.0;
        }

        linear_equation(A,x,y,N);

        for(int j=0;j<N;j++)
        {
            B[j][i]=x[j];
        }

    }
    free(x);
    free(y);
}
```

//行列の内容を csv ファイルに書き出す関数

```
bool write_csv( char* filename, double** A, int rows, int cols )
```

```
{
    FILE *fp;

    if((fp=fopen(filename,"wb")) != NULL)
    {
        for(int i=0;i<rows;i++)
        {
            for(int j=0;j<cols;j++)
            {
                fprintf(fp,"%lf",A[i][j]);
                if(j != (cols-1)) fprintf(fp,",");
                else    fprintf(fp,"\r\n");
            }
        }
        return 1;
    }
    else return 0;
    fclose(fp);
}
```

```

//評価対象の1行を取り除く関数
void new_mat( int rows, double** data, double** temp, double** new_make )
{
    int i,j,k=0;

    for(i=0; i<100; i++){
        if(i!=rows)
            {
                for(j=0; j<9; j++)    new_make[k][j]=data[i][j];
                k++;
            }
        else{
            for(j=0; j<9; j++)
                temp[rows][j]=data[rows][j];
        }
    }
}

```

```

//csv ファイルから行列に値を読み込む関数
bool read_csv( char* filename, double** A, int rows, int cols )
{
    FILE *fp;
    int i,j;
    char *data,*q,*p;
    data=(char*)malloc(sizeof(char)*1024);

    if((fp=fopen(filename,"rb")) != NULL)
    {
        i=0;
        while(fgets(data,1024,fp) != NULL)
        {
            q=data;
            j=0;
            for(;;)
            {
                p = strtok( q, "," );
                if( !p ) break;
                q = NULL;
                A[i][j]=atof( p );
                j++;
            }
            i++;
        }
        fclose(fp);
        free(data);
        return 1;
    }
    else{

```

```

        free(data);
        return 0;
    }
}

int main(int argc, char* argv[])
{
    double **data,**D,**Y,**Dt,**DtD,**DtDi,**DDD;
    double **DDDY,**temp,**mat,**zansa,**gosa,**zansajijou;
    double jijougosa=NULL,a=NULL;
    int i,j;

    mat = make_mat(100,9);
    data = make_mat(99,9);
    D = make_mat(99,9);
    Y = make_mat(99,1);
    Dt = make_mat(9,99);
    DtD = make_mat(9,9);
    DtDi = make_mat(9,9);
    DDD = make_mat(9,99);
    DDDY = make_mat(9,1);
    temp = make_mat(100,9);
    zansa = make_mat(100,1);
    zansajijou = make_mat(100,1);
    gosa = make_mat(2,1);

    read_csv("itemman.csv",mat,100,9);

    for(i=0; i<100; i++){
        new_mat(i,mat,temp,data);
        copy_submat2(data,D,0,98,0,7);
        copy_submat(data,Y,0,98,8,8);
        t_mat(D,Dt,99,9);
        prod_mat(Dt,D,DtD,9,99,99,9);
        inverse_mat(DtD,DtDi,9);
        prod_mat(DtDi,Dt,DDD,9,9,9,99);
        prod_mat(DDD,Y,DDDY,9,99,99,1);

        for(j=1; j<9; j++)
        {
            a=a+DDDY[j][0]*temp[i][j-1];
        }
        a=a+DDDY[0][0];

        zansa[i][0]=abs(temp[i][8]-a);
        zansajijou[i][0]=sqr(temp[i][8]-a);
        jijougosa=jijougosa+zansajijou[i][0];
        a=0;
    }
}

```

```
gosa[0][0]=jijougosa/100;
gosa[1][0]=sqrt(jijougosa/100);
write_csv("zansa_man.csv",zansa,100,1);
write_csv("zansajjou_man.csv",zansajjou,100,1);
write_csv("gosa_man.csv",gosa,2,1);
```

```
free_mat(D);
free_mat(Y);
free_mat(Dt);
free_mat(DtD);
free_mat(DtDi);
free_mat(DDD);
free_mat(DDDY);
free_mat(zansajjou);
free_mat(zansa);
free_mat(mat);
free_mat(temp);
free_mat(gosa);
```

```
return 0;
```

```
}
```

- 協調フィルタリングの楽曲推薦システム

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define N 100
#define P 14
#define sqr(x) ((x)*(x))

//楽曲データ・評価データのファイル読み込み
//書き込みファイルオープン
int read_data(int valuation[N][P], char title[N][25], char name[P][10])
{
    FILE *fp_valu,*fp_mu,*fp_name;
    int i,j;

    fp_valu=fopen("valuation_precision.csv","r");
    if(fp_valu==NULL)
    {
        printf("valuation-file open shippai!!\n");
        return 1;
    }else printf("valuation-file open!!\n");

    for(i=0; i<N; i++){
        for(j=0; j<P; j++) fscanf(fp_valu,"%d",&valuation[i][j]);
        fprintf(fp_valu,"\n");
    }
    fclose(fp_valu);

    fp_mu=fopen("music.csv","r");
    if(fp_mu==NULL)
    {
        printf("music-file open shippai!!\n");
        return 1;
    }else printf("music-file open!!\n");

    for(i=0; i<N; i++) fscanf(fp_mu,"%s\n",title[i]);
    fclose(fp_mu);
    fp_name=fopen("name.csv","r");
    if(fp_name==NULL)
    {
        printf("name-file open shippai!!\n");
        return 1;
    }else printf("name-file open!!\n");
}
```

```

for(i=0; i<P; i++) fscanf(fp_name, "%s\n", name[i]);
fclose(fp_name);

printf("\n\n リスナーの評価結果\n (曲");
for(i=0; i<P; i++)
    printf("-%s", name[i]);
printf("の順) \n");
for(i=0; i<N; i++){
    printf("%25s", title[i]);
    for(j=0; j<P; j++)    printf("%3d", valuation[i][j]);
    printf("\n");
}

return 0;
}

```

//全リスナーの各平均を算出

```

void person_average(int valuation[N][P], double average[])
{
    int    i,j,cnt;

    for(i=0; i<P; i++)
    {
        cnt=0;
        for(j=0; j<N; j++)
        {
            average[i]=average[i]+valuation[j][i];
            if(valuation[j][i]==0) cnt=cnt+1;
        }
        average[i]=average[i]/(N-cnt);
    }
}

```

//全リスナーの相関を算出

```

void correlation(int valuation[N][P], double corre[P][P])
{
    int cnt=0;
    int i,j,k,l;
    double avesx=0,avesy=0,bunsany=0,bunsanx=0,kyobunsan=0;

    for(i=0; i<P-1; i++){
        for(j=P-1; j>i; j--){
            for(k=0; k<N; k++){
                if(valuation[k][i]!=0 && valuation[k][j]!=0)
//どちらも評価している楽曲だけで平均を出す
                {

```

```

        avesx=avesx+valuation[k][i];
        avey=avesy+valuation[k][j];
    }else cnt=cnt+1;
}
avesx=avesx/double(N-cnt);
avey=avesy/double(N-cnt);
for(l=0; l<N; l++){
    if(valuation[l][i]!=0 && valuation[l][j]!=0){
        bunsanx=bunsanx+sqr(valuation[l][i]-avesx);
        bunsany=bunsany+sqr(valuation[l][j]-avey);
        kyobunsan=kyobunsan+
            ((valuation[l][i]-avesx)*(valuation[l][j]-avey));
    }
}
/****相関=共分散/(標準偏差×標準偏差)****/
corre[i][j]=kyobunsan/((sqr(bunsanx))*(sqr(bunsany)));
avesx=0;
avey=0;
bunsany=0;
bunsanx=0;
kyobunsan=0;
cnt=0;
}
}
}

```

//未評価曲の評価値を予測

```

int estimate( int number, //m : 人数
              int valuation[N][P], //user[][] : 実評価
              double average[], //ave[] : リスナー平均
              double corre[P][P], //soukan[][] : 類似度
              int not_valu[], //no_valu[] : 未評価曲
              double estimate[]) //valu_calcu[] : 評価予測
{
    int i,j,k,m;
    int cnt=0,w=0;
    double ave_valu[N][P];
    double weight[N],corre_add[N];
    double sum=0,sum_weight=0,sum_add=0;

    for(k=0; k<N; k++){
        if(valuation[k][number]==0){
            not_valu[w]=k;
            w=w+1;}
    }

    for(i=0; i<w; i++){
        k=not_valu[i];

```

```

        for(j=0; j<P; j++){
            for(m=0; m<N; m++){
                if(valuation[k][j]!=0){
                    sum=sum+valuation[m][j];
                    if(valuation[m][j]==0)
                        cnt=cnt+1;
                }
            }
            ave_valu[k][j]=(sum-valuation[k][j])/(N-cnt-1);
            sum=0; cnt=0;
        }
    }

    for(i=0; i<N; i++)
    {
        weight[i]=0;
        corre_add[i]=0;
    }

    for(i=0; i<w; i++){
        k=not_valu[i];
        for(j=0; j<P; j++){
            if(valuation[k][j]!=0){
                sum_weight=sum_weight+(valuation[k][j]-
                    ave_valu[k][j])*(corre[j][number]+corre[number][j]);
                sum_add=sum_add+abs((corre[j][number]+corre[number][j]));
            }
        }
        weight[i]=sum_weight;
        corre_add[i]=sum_add;
        sum_weight=0;
        sum_add=0;
    }

    for(i=0; i<w; i++)    estimate[i]=average[number]+(weight[i]/corre_add[i]);

    return w;
}

int main(void)
{
    int        i,j,x,temporary,n,m;
    int        user[N][P];
    int        no_valu[N];
    double num=0,sum=0;
    double soukan[P][P],zansajijou[N][P];
    double valu_calcu[N],jijougosa[P],ave[P];
    char        music[N][25],name[P][10];
    FILE        *fp_out1,*fp_out2,*fp_out3;

```

```

printf("強調フィルタリングシステム\n アルゴリズム=GroupLens\n\n");

fp_out1=fopen("Recommendation value.csv","w");
fp_out2=fopen("zansajjou.csv","w");
fp_out3=fopen("jijou-gosa.csv","w");

/****評価データの読み込み****/
read_data(user,music,name);

for(n=0; n<N; n++){
    for(m=0; m<P; m++){
        temporary=user[n][m];
        user[n][m]=NULL;
        for(i=0; i<P; i++){
            ave[i]=0;

            /****全リスナーの各平均を求める****/
            person_average(user,ave);

            /****相関(類似度)の算出****/
            correlation(user, soukan);
            for(i=0; i<P; i++){
                for(j=0; j<P; j++){
                    if(abs(soukan[i][j])>1)
                        soukan[i][j]=NULL;
                }
            }

            /****評価予測****/
            x=estimate(m, user, ave, soukan, no_valu, valu_calcu);
            for(i=0; i<x; i++){
                fprintf(fp_out1,"%9.5f",valu_calcu[i]);
                zansajjou[n][m]=sqr(temporary-valu_calcu[i]);
                fprintf(fp_out2,"%9.5f",zansajjou[n][m]);
            }

            user[n][m]=temporary;
        }
        fprintf(fp_out1,"\n");
        fprintf(fp_out2,"\n");
    }
}

for(i=0; i<P; i++){
    for(j=0; j<N; j++){
        num=num+zansajjou[j][i];
    }
    jijougosa[i]=num/N;
}

```

```

    fprintf(fp_out3,"%8s",name[i]);
    fprintf(fp_out3,"  $\sigma^2$ =%f",jijougosa[i]);
    fprintf(fp_out3,"  $\sigma$ =%fn",sqrt(jijougosa[i]));
    sum=sum+sqrt(jijougosa[i]);
    num=num+1;
}

fprintf(fp_out3,"\n\n 平均自乗誤差の全平均=%fn",sum/P);

fclose(fp_out1);
fclose(fp_out2);
fclose(fp_out3);

return 0;
}

```

参考文献

- [1] News Release: <http://www.apple.com/jp/news/2007/jul/31itunes.html/>.
- [2] Napster Japan: <http://www.napster.jp/>.
- [3] 黒瀬崇弘, 梶川嘉延, 野村康雄: 感性情報を用いた楽曲推薦システム, 第14回データ工学ワークショップ (DEWS2002), 8-P-6 (2003-03).
- [4] 鹿又広行, 小林利彰, 東海林智也: HMMを用いた楽曲推薦システムの構築. FIT2007.
- [5] 土方嘉徳, 岩濱数宏, 西田正吾: 決定木を用いた内容に基づく音楽情報フィルタリング. インタラクション 2005.
- [6] 西尾毅士, 田村哲嗣, 速水悟: より良い音楽推薦システムへ向けた試み.
- [7] 小原恭介, 山田剛一, 絹川博之, 中川裕志: bloggerの嗜好を利用した協調フィルタリングによるWeb情報推薦システム. JSAI2005.
- [8] 数量化について: <http://gucchi24.hp.infoseek.co.jp/SUR1.htm>.
- [9] 中森義輝: 感性データ解析. 森北出版, 2000.
- [10] 長谷川勝也: ゼロからはじめてよくわかる多変量解析. 技術評論社, 2005.
- [11] 福原知宏: 協調フィルタリングに関する研究動向. 情報システム学専攻ロボティクス講座, 1998.
- [12] 増井俊之: インターフェイスの街角(93) – 本棚演算. <http://pitecan.com/UnixMagazine/PDF/if0512.pdf>.
- [13] マイケルJ.A.ベリー, ゴードンS.リノフ: データマイニング手法. 海文堂, 1999.
- [14] 石井一夫: 図解よくわかるデータマイニング. 日刊工業新聞社, 2004.
- [15] 鹿又広行: 音楽的特徴と嗜好情報に基づいた楽曲推薦システムの構築.
- [16] 渡辺崇文, 廣安知之, 三木光範: 協調フィルタリング. <http://mikilab.doshisha.ac.jp/dia/research/report/2007/1019/004/report20071019004.html>.
- [17] RWC 研究用音楽データベース: <http://staff.aist.go.jp/m.goto/RWC-MDB/index-j.html>.
- [18] 後藤真孝, 橋口博樹, 西村拓一, 岡隆一: RWC 研究用音楽データベース: ポピュラー音楽データベースと著作権切れ音楽データベース. 情報処理学会 音楽情報科学研究会, 2001-MUS-42-6, Vol.2001, No.103, pp.35-42, October 2001.