卒業論文

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

東海林研究室

5 年情報工学科　出席番号 12 番

鹿又　広行

目次

1章　序論

1-1　はじめに

近年、携帯音楽プレイヤーや音楽再生可能な携帯電話の普及が広がりを見せている。これに伴い、楽曲のインターネット配信技術が急速に発展し、また、楽曲の購入が容易なものとなっている[1]。パソコンに楽曲をデータとして保存し、音楽プレイヤーなどへ好きな楽曲データを移動させて音楽を楽しむ手軽さが人気を呼んでいるわけだが、あまりにも楽曲数が膨大になると、移動させる楽曲の選択にも時間がかかり、「手軽」とはいえなくなる。選択するのが面倒だという聴取者もいるのが現実で、適当な楽曲を移動し、一定のアーティストやジャンルに依らない様々な楽曲を聴く聴取者が増加している[2]。

当然ながら聴取者は各人の嗜好を持っているため、本人が無意識のうちに「音楽を聴くのならこの曲・このジャンル・このアーティスト」といった指針により移動楽曲の選択を行っているはずである。つまりこの選択を誤っていい加減な楽曲選択を行うと、実際に聴いたとき「なぜこの曲を入れたのか？」「この曲は聴きたくない」といった事態になりかねない。

また、その一方、「何かいい曲はないだろうか？」「最近デビューしたアーティストの曲でおすすめは？」「これから流行りそうな曲は？」といったような考えを持って、楽曲購入の際に未知のジャンル・アーティストの楽曲を求めている人も少なからずいる。事実このような場合は聴取者が実際に楽曲を視聴してみなければ自分の嗜好に一致するか評価できず、視聴できる場が多く提供されていないのが現状である。

以上の問題があってか楽曲の推薦システムの研究が数多く行われてきており、少ないながらもインターネットを介しての推薦楽曲の提供の試みもなされている[3][4][5]。様々な手法により楽曲推薦システムが構築されているが、最も推薦の性能が高い手法が何なのかについては未だ明確ではない。我々は事前研究の形で HMM（隠れマルコフモデル）を推薦手法として楽曲推薦を行い、その手法の性能を評価するための実験を行ったが、あまり良い結果は得られなかった[6]。また、その「嗜好情報の収集時間が長い」、「嗜好の判断が困難」といった問題があり、特に収集時間については一人約2時間を要し、これらの問題は実際のシステム利用上の大きな障害となる。

そこで本研究では「嗜好情報の収集時間が長い」、「嗜好の判断が困難」という問題を解決する楽曲推薦システムの構築を目指す。「嗜好情報の収集時間が長い」という問題を解決するために印象的なフレーズを切り出し、ウェブブラウザを用いネットワークを通じて嗜好情報を収集する。また、「嗜好の判断が困難」という問題を解決するために、データマイニング手法において個人向け推薦コンテンツを提供するに適しているといわれる記憶ベース推論のｋ近傍法を楽曲推薦のための手法として提案する[7][8]。さらに実際に被験者の楽曲に対する嗜好の収集実験を行い、被験者の嗜好情報と楽曲の音楽的特徴をもとに楽曲推薦の結果を検証する。

1-2　本論文の構成

本論文は、本章を含めて 7 章で構成される。
2 章では、構築する楽曲推薦システムの構成について説明する。
3 章では、記憶ベース推論の k 近傍法による嗜好印象の推定方法と、しきい値による楽曲の推薦の方法を説明する。
4 章では、3 章までに述べた推薦手法を用いて行った実験の内容とその結果について報告する。
5 章では、本研究の内容とその成果をまとめる。また、残された今後の課題についても述べる。
6 章では、本研究で作成したソースコードを掲載する。
7 章では、本論文を作成するにあたって参考にした資料を掲載する。

2章　楽曲推薦システムの構築

2-1　システムの仕様

本システムでは、楽曲推薦対象者の既知の楽曲に対する嗜好印象を入力し、未知の楽曲から推薦楽曲を提供する。

入力された嗜好印象はCSVファイルとして保存され､R言語で作成されたプログラムによりCSVファイルの読み込みを行い楽曲推薦を行う[9][10]。

2-2　楽曲の嗜好印象の入力について

事前研究での実験で目立った大きな欠点として、嗜好印象の入力を専用用紙を用いて手書きで行っていたという煩わしさがあった。また、そのデータをパソコンで処理する際に、再度用紙からパソコンへデータを入力する必要があり、2度手間になっていた。本研究では、その点を反省し、PHPによる入力フォームの制作を行った[11][12][13]。ユーザはパソコンのウェブブラウザを利用し、ネットワーク越しにこのフォームから嗜好印象の入力を行う。

以下にフォームでのデータ入力方法をまとめた。

図1は楽曲推薦対象者の名前(ユーザネーム)を入力する画面のスクリーンショットである。ここで入力した名前が、出力されるCSVファイル名に反映し、後々の処理が簡単となる。

この画面で名前を入力し、「Enter」ボタンをクリックすると図2の画面へとリンクする。図2の画面が表示されると自動的に楽曲(SWF形式)の再生を開始する。被験者は、このとき再生されている楽曲に対し、好きか嫌いかを5段階で評価する。楽曲はリピート再生されるようになっており、被験者がラジオボタンで評価を入力し「次の曲へ」ボタンをクリックすると、すべての楽曲を評価し終えていたなら図3の画面へ、そうでなければ図2の画面へとリンクし、全楽曲を評価し終えるまで同様の作業を行う。

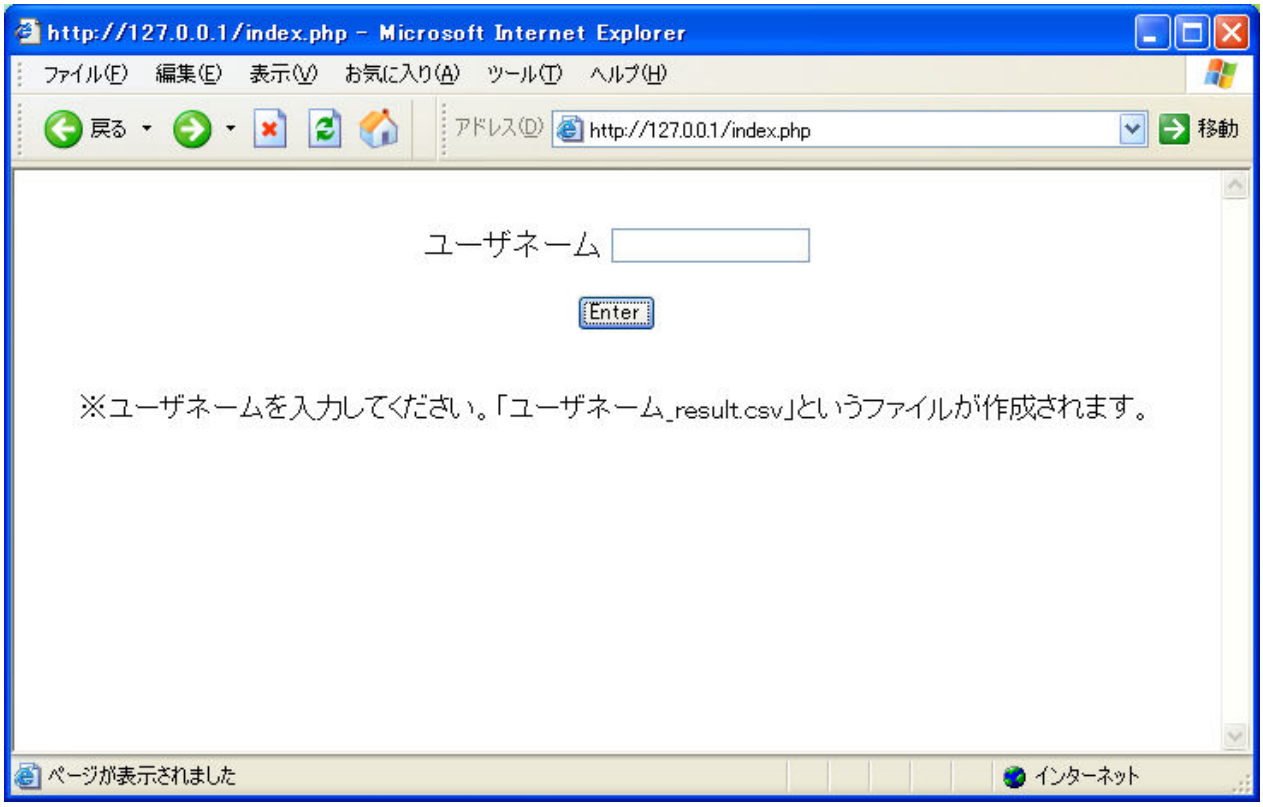

図 1　ユーザネーム入力画面

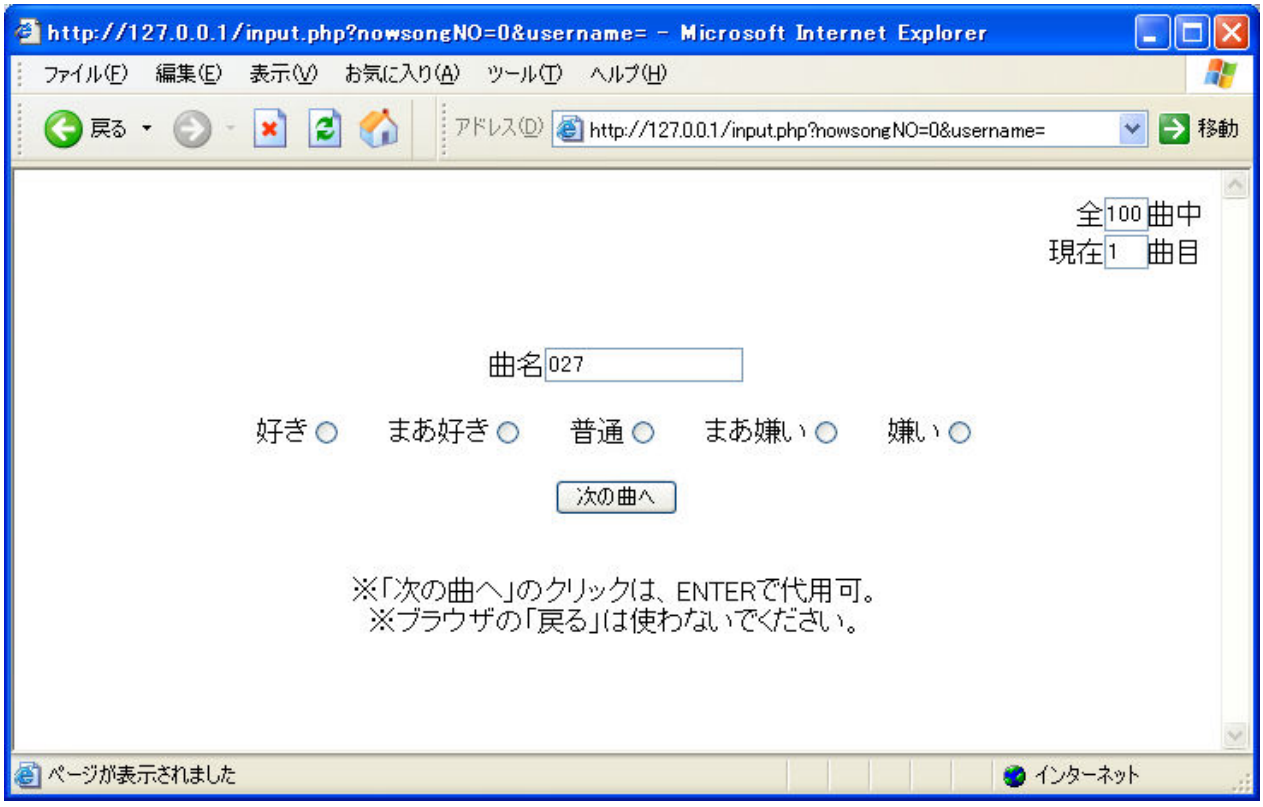

図 2　楽曲の嗜好印象の入力画面

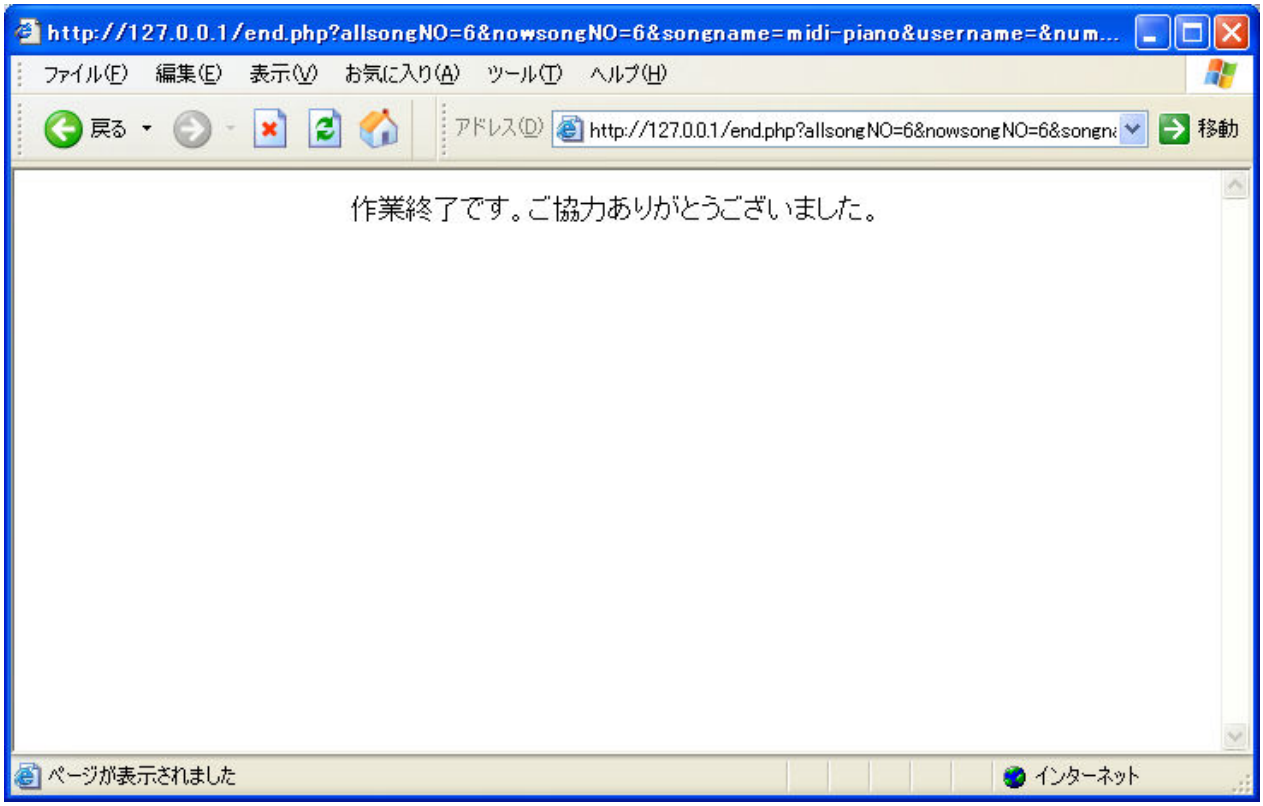

図 3　作業完了画面

なお楽曲の再生順序に気を付ける必要がある。一般にこうしたデータ入力の際に生じる問題として「順序効果」というものがある。これは「今、行った操作・判断が、次に行う操作・判断に少なからず影響する」というものである。順序効果をできる限り防ぐために、楽曲の再生方法としてランダム順を採用した。楽曲をランダムに再生することで、過去の楽曲評価からの影響を最小限にできる。

楽曲の再生順序はランダムであるが、楽曲名とその嗜好印象が書き込まれている CSV の内容は一定の書式に従っている。つまり、被験者同士で楽曲の再生順は異なるが、その再生順で CSV ファイルが出力されることはなく、すべて同様の形式の CSV ファイルが出力される。このようにファイル形式を統一すると後の処理で大きく役立つ。

なお、今回制作した入力フォームの欠点であるが、次のようなものがあげられる。

(1)　見栄えが悪い。レイアウトがシンプルすぎる。

(2)　ブラウザの「戻る」を使用した場合に、作成される CSV ファイルや動作に支障をきたすことがある。

(3)　途中でフリーズしてしまうと、再度最初から操作をしなおさなければならないことがある。

これらについては今後改善予定である。なお、コンテンツの開発には、開発環境提供ソフトとして「XAMPP」、エディタソフトとして「PHP エディタ」を利用した[14][15]。

3章　推薦手法の理論的背景

3-1　データマイニングとは

推薦手法としてデータマイニングという手法に注目した[8][16]。

「マイニング」とは「発掘」という意味であり、大量にあるデータの中から分析に有用なデータを「発掘」し、意味のあるルールを発見することをデータマイニングという。ここでいう「ルール」とは、厳密に言うと、データの項目間に内在する規則性や異なる属性の値に関して成立することをある適切な形で表現したものである。

音楽的特徴とその嗜好との関係は、人間に関係しているだけあって、非常に複雑である。したがって、線形演算的な手法では、人間の嗜好を表現できるとは到底考えられない。しかし、このデータマイニングという手法をもちいれば、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。このようにデータの集まりからその裏にある関係を見つけ出すのに適しているため、本研究での楽曲推薦に有効であると思われる。

3-2　データマイニング手法～記憶ベース推論～

データマイニング手法の一つに記憶ベース推論というものがある[8]。

記憶ベース推論では、新しいデータを入力した際、最も類似しているレコードを探し出し、そのレコードの情報から出力結果を推定する。記憶ベース推論はレコードの形式を問わず、任意の 2 つのレコード間の距離を計算する「距離関数」と、いくつかの近傍レコードから得られた結果を結合し 1 つの答えを得る「結合関数」の 2 つの操作を行う関数が重要となる。

こうした関数は多種多様なデータに対しても定義が可能であるため、記憶ベース推論はほかのデータマイニング手法では適用困難な問題に対しても有効である。また、新たなデータが増えた場合でも新たな操作を必要とせず、推定するときに新たなレコードと既存のレコードの間で距離関数と結合関数による計算をすればよい。

記憶ベース推論は、個人向け推薦コンテンツを提供するのに適用することができる。この方法では、まず人々の選好の履歴を学習し、そして、2 者間の距離を選好の重なり度合いに基づき定義する。つまり同じものを好む人々が近いとされる。実際に推薦を行う際は、結合関数は重み付け多数決によりなされる。このように似たような趣味趣向を持つとして選ばれた中間集団の情報を利用して、特定の人物のその時点の嗜好に適合する商品やサービスを探し出すことができる[8]。

本研究に記憶ベース推論を適用すると、次のような手順を踏むことになる。

(1)　推薦対象者が過去に聴取し、嗜好印象を持つ楽曲を用意する。
(2)　用意した(既知の)楽曲の音楽的特徴を抽出する。
(3)　推薦対象とする未知の楽曲の音楽的特徴を抽出する。
(4)　既知の楽曲と未知の楽曲との距離をそれぞれの音楽的特徴から計算する。

図 4 は上記(4)の完了時点でのイメージを表している。ここから未知の楽曲 X について嗜好印象の推定を行う。

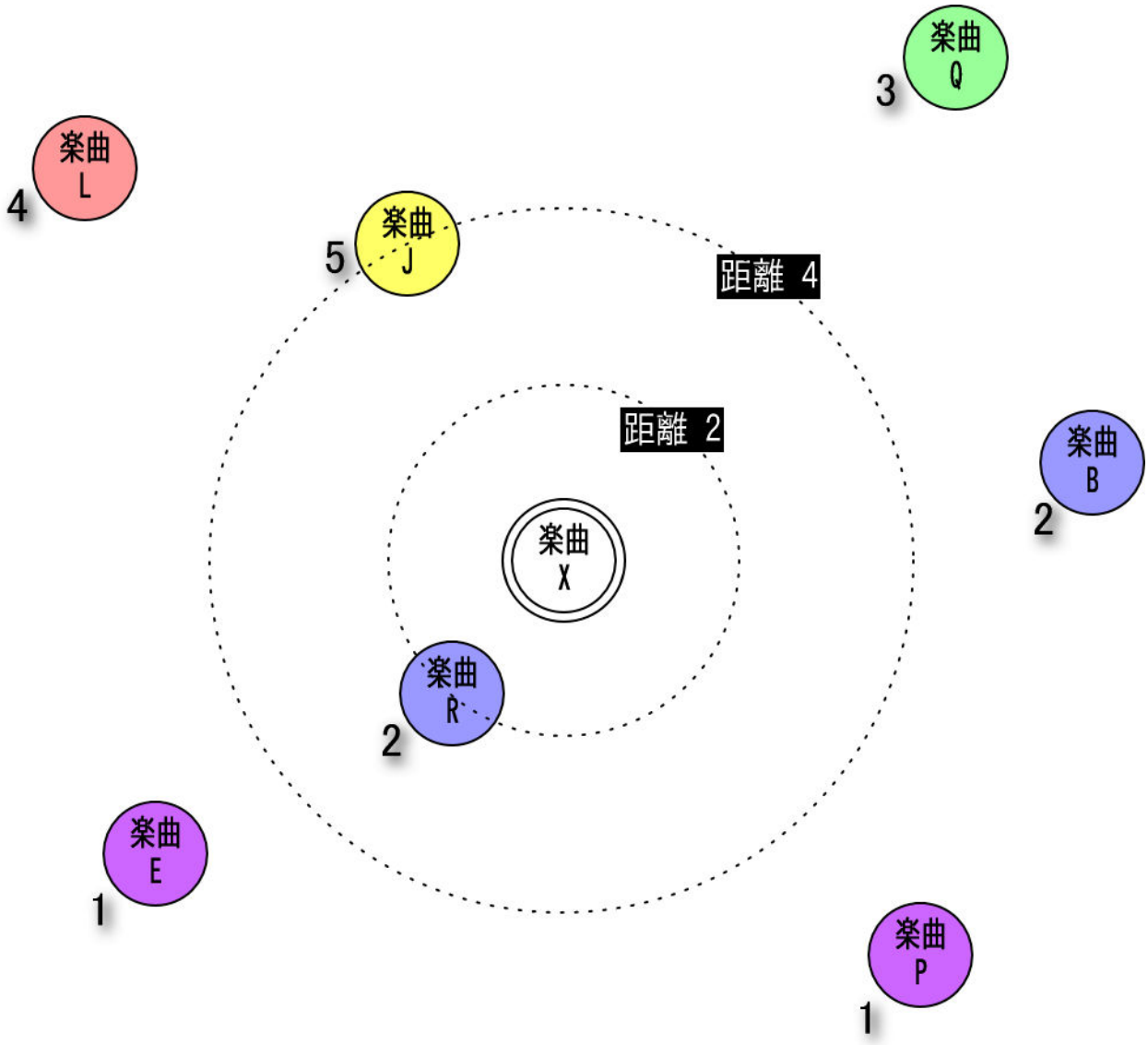

図 4　楽曲推薦での記憶ベース推論

(5)　未知の楽曲から近い順に k 個の既知の楽曲を選択する。

仮に今回は k=2 であるとする。そうした場合、楽曲 X から一番近い楽曲 R と次に近い楽曲 J の 2 楽曲が選択される。このような k 個の近傍のデータから一般的なものを割り当てる方法は k 近傍法と呼ばれる[17]。

(6)　選択した楽曲の嗜好印象とその距離から未知の楽曲の嗜好印象を推定する。

楽曲 R の嗜好印象は 2、楽曲 J の嗜好印象は 5 である。一般的な計算手法として、距離の逆数で重み付けした平均を求める。つまり、楽曲 R の場合は距離 2 の逆数 $\frac{1}{2}$ を、楽曲 J の場合は距離 4 の逆数 $\frac{1}{4}$ を重みとして計算する。以下に計算方法を示す。

$$\frac{\frac{1}{2}\times 2+\frac{1}{4}\times 5}{\frac{1}{2}+\frac{1}{4}}=\frac{9}{3}=3$$

結果として楽曲 X の嗜好印象推定値 3 が得られる。

3-3　嗜好印象の推定値について

前節で提案した楽曲推薦手法を用いて得られた楽曲の嗜好印象の推定値は、楽曲推薦対象者の楽曲に対する嗜好度を表しているということは言うまでもない。楽曲推薦とは「好きであると推定された楽曲を推薦する」ということである。そのため、嗜好印象の推定値を、楽曲を推薦するかしないかの判断材料として用いる。

推定値を楽曲推薦に用いるには次のような処理が必要であると考える。

(1)　推定値を適正な値に補正する。

推定値を適切な値へ補正する。たとえば、本研究では嗜好印象を 5 段階としているため、それらの値に補正する。例えば、対象者の好きな楽曲だけでなく、嫌いな楽曲の傾向を知る場合など、楽曲の分類を行うことから楽曲推薦する場合にはこの方法が望ましい。最も近い値を選択するなど、比較的簡単な方法である。

(2)　推定値に対ししきい値を適用する。

あるしきい値を用意し、推定値としきい値を比較して楽曲推薦を決定する。しきい値の設定が難しいが、適正なしきい値が得られた場合には推薦の性能が大きく向上すると思われる。

本研究ではしきい値処理による方法を採用し、楽曲推薦を行う。

3-4　しきい値を用いた推薦楽曲の決定

提案した推薦手法を用い、ある未知の楽曲の嗜好印象の推定値が得られたとする。こうしたときある基準となる値と比較し、それより大きいならば推薦するといった方法が一般に行われるだろう。この基準となる値はしきい値と呼ばれる。

例えば事前にしきい値を 4 と固定しておき、推定値 3.9 が得られたとする。このとき「しきい値以下だから推薦しない」と判断を行う。しかし、この例のようにしきい値を事前に固定しておく方法は得策とは言えない。なぜならば楽曲推薦は、楽曲推薦対象者の楽曲に対する嗜好印象だけではなく、それ以外の要因について考える必要があるからである。この例の場合、もし対象者の嗜好印象の基準が一般に比べて低かったなら、つまり、実際に感じた嗜好印象に対しそれを過小評価していたのなら、3.9 という推薦値は決して低い値とは言えない。このように、しきい値の設定は特に困難であり、それが適切になされなかった場合、楽曲推薦の性能を大きく損ないかねず、様々な要因を考慮した上で適切なしきい値を設定する必要がある。

本研究では、適切なしきい値を設定するために学習用楽曲を利用してしきい値を動的に変化させて最終的な楽曲推薦のしきい値を決定する。以下にその方法をまとめる。

(1)　学習用楽曲の嗜好印象の推定値を計算する。

学習用楽曲全曲の嗜好印象の推定値を、自身である学習用楽曲から計算する。つまり学習用楽曲が 100 曲の場合を考えると、その 1 曲目を推定するために 2～100 曲目を用い、2 曲目であれば、1 曲目と 3～100 曲目を用いる、といった具合である。これを繰り返し行い、全曲分の推定値を計算する。

(2)　しきい値を 5 から“学習用楽曲の実嗜好印象の平均値”まで 0.02 刻みで変化させ、全学習用楽曲に対し(3)を行う。

最終的なしきい値の最低値は学習用楽曲の実際の嗜好印象の平均値としている。平均値は好きでも嫌いでもない境目であると考えられるため、このようにした。

(3)　嗜好印象の推定値としきい値とを比較し、得点の計算を行う。

上記の「得点」とはしきい値を適用した場合のその適応度を表している。得点が高いとき、そのしきい値は適正であるといえる。得点については以下に詳細をまとめる。

得点は、経験的に得られた整数に、さらに重みとして「学習楽曲内での出現回数の逆数」を掛けたものとしている。たとえば実嗜好印象が「1」である楽曲が学習曲内に 20 曲含まれている時、この逆数の 1/20 を嗜好印象「1」での重みとする。この重みにより補正を行う。

- 嗜好印象の推定値 ≧ しきい値　の場合

実嗜好印象が「3」ならば、「 5×(嗜好印象「3」での重み)」の得点を得る。
実嗜好印象が「4」ならば、「 7×(嗜好印象「4」での重み)」の得点を得る。
実嗜好印象が「5」ならば、「 9×(嗜好印象「5」での重み)」の得点を得る。

しきい値より推定値が大きいならば、その楽曲の実嗜好印象は高くならなければならない。そのため「3」、「4」、「5」では得点を得て、「1」や「2」ならば得点を得ない。

- 嗜好印象の推定値 ＜ しきい値　の場合

実嗜好印象が「1」ならば、「 5×(嗜好印象「1」での重み)」の得点を得る。
実嗜好印象が「2」ならば、「 3×(嗜好印象「2」での重み)」の得点を得る。

しきい値より推定値が小さいならば、その楽曲の実嗜好印象は低くならなければならない。そのため「1」や「2」では得点を得て、「3」、「4」、「5」ならば得点を得ない。

全楽曲に対してこの得点の計算を行う。全楽曲の得点の和をそのときのしきい値の得点とする。

(4)　最大の得点になるときのしきい値を最終的な楽曲推薦のしきい値とする。

4章　実験

4-1　実験概要

前章で提案した推薦手法による楽曲推薦の性能を検証する。被験者は用意された楽曲 100 曲を聴取し、その楽曲に対してそれぞれの嗜好印象を評価する。評価の入力は制作したウェブブラウザのフォームから行う。

実験対象者は東海林研究室の 4・5 学年の学生 11 名と、研究室以外の学生 2 名の計 13 名である。なお、4 学年の学生については一斉に実験を行い、実験を開始してから完了するまでの実験時間を測定した。それ以外の学生は任意の時間に行ったが、4 学年の学生を含め、被験者全員に対して実験の制限時間は設定していない。

評価する楽曲として、RWC 研究用音楽データベースの楽曲を用いた[18]。 RWC 研究用音楽データベースは、研究目的に利用できる様々な楽曲を集めたデータベースである。研究用として用いるのなら学会などの対外発表でも利用でき、著作権の問題もない。実験では、数多くあるデータベースの中から邦楽 80 曲、洋楽 20 曲のポピュラー音楽 100 曲(RWC-MDB-P-2001 No.1～100)を利用した。すべて異なるアーティストの楽曲ではなく、同一のアーティストにより楽曲も複数含まれている。ジャンルとしては、JPOP、ダンスミュージック、ロック、ソウル、HIPHOP、メタル、フォーク、引き語りなど様々である。研究用に制作された楽曲なので、未知の楽曲であることもあり、被験者の嗜好印象の収集用にはとても有効であると思われる。

また、実験の目的である時間短縮を図るために全 100 曲の特定のフレーズの約 15 秒だけを抽出して実験に利用した。特定のフレーズというのは、いわゆる楽曲の「サビ」部分である。サビとは、主に楽曲の中で最も盛り上がる部分と定義されている。楽曲の嗜好印象を知るためには、その楽曲の一番特徴的な部分、つまりサビを聴取するのが最善策だと思われる。

さらに、実験に際し、楽曲がどのような特徴を持っているのかを事前に評価した音楽的特徴テンプレートを用意しておいた。テンプレートは詳細度の違う 1 と 2 の 2 種類を用意し、この違いによる楽曲推薦の性能を比較する。この評価を適切に行えるかどうかが、楽曲推薦の性能を左右する。

表 1　音楽的特徴の評価項目について

項目名	評価内容			得点の使用数値	
	注目する特徴	低得点	高得点	テンプレート1	テンプレート2
性別	アーティストは	男性	女性	0，1	0，1
言語	言語は	日本語	英語	0，1	0，1
テンポ	テンポは	遅い	早い	0，1	0，1，2
明暗	曲調は	暗い	明るい	0，1	0，1，2
年齢	歌声は	若い	若くない	0，1	0，1，2

表 1 は評価項目とその内容についてまとめたものである。項目については以下で説明する。

(1)　性別

特に若い人の間では、男性の楽曲が好きな女性や、女性の楽曲が好きな男性など、性別が嗜好印象に表れやすいと思われる。

(2)　言語

「英語は歌詞が分からない」などの理由で、洋楽を嫌う人もいるはずである。逆に「英語は格好良い」と感じる人もいる。

(3)　テンポ

テンポは曲のジャンルと大きく関係する。一概には言えないが、「ロック」なら早い、「フォーク」なら遅いなどである。なお、「ジャンル」を評価項目として用意しなかったのは、その要素が多くなり、推薦手法の適用が難しくなると考えられるからである。

(4)　明暗

楽しい曲が好きな人もいれば、落ち着いた曲が好きな人もいる。明暗もジャンルを表す要素となる。

(5)　年齢

この項目は、声の感じ方を示している。実際の年齢ではなく、声の年齢である。若々しい声が好きな人もいれば、その逆の人もいるだろう。

最後に、被験者より得られたデータに推薦手法を適用する方法としては、R 言語により作成したプログラムと、Weka と呼ばれるデータマイニングソフトを用いた[19]。

4-2　実験結果

実験から得られたデータに、提案した推薦手法を適用した結果を示す。被験者の中から 6 名を選択し、R 言語のプログラムでの k 近傍法としきい値処理による楽曲推薦を行った。実嗜好印象を持つ楽曲 100 曲のうち、99 曲を学習用楽曲として用い、残りの 1 曲に対して推薦の判断処理を行い、それをすべての組み合わせで行う。そして推薦された楽曲が実際に高い嗜好印象であるかを比較する。表 2ー表 7 にその結果をまとめた。

なお、分析の入力データである楽曲の音楽特徴量はテンプレート 1 とテンプレート 2 の両方を用いた。

表 2　被験者 1

テンプレート1		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	2	4.3%	28.3%
	2	11	23.9%	
	3	18	39.1%	39.1%
	4	15	32.6%	32.6%
	5	0	0.0%	
計		46	100.0%	

テンプレート2		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	2	3.7%	22.2%
	2	10	18.5%	
	3	24	44.4%	44.4%
	4	18	33.3%	33.3%
	5	0	0.0%	
計		54	100.0%	

表 3　被験者 2

テンプレート1		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	8	25.8%	35.5%
	2	3	9.7%	
	3	10	32.3%	32.3%
	4	5	16.1%	32.3%
	5	5	16.1%	
計		31	100.0%	

テンプレート2		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	5	19.2%	34.6%
	2	4	15.4%	
	3	3	11.5%	11.5%
	4	7	26.9%	53.8%
	5	7	26.9%	
計		26	100.0%	

表 4　被験者 3

テンプレート1		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	1	1.7%	35.6%
	2	20	33.9%	
	3	20	33.9%	33.9%
	4	15	25.4%	30.5%
	5	3	5.1%	
計		59	100.0%	

テンプレート2		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	4	7.1%	37.5%
	2	17	30.4%	
	3	20	35.7%	35.7%
	4	11	19.6%	26.8%
	5	4	7.1%	
計		56	100.0%	

表 5　被験者 4

テンプレート1		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	3	7.1%	45.2%
	2	16	38.1%	
	3	12	28.6%	28.6%
	4	6	14.3%	26.2%
	5	5	11.9%	
計		42	100.0%	

テンプレート2		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	6	11.1%	40.7%
	2	16	29.6%	
	3	17	31.5%	31.5%
	4	8	14.8%	27.8%
	5	7	13.0%	
計		54	100.0%	

表 6　被験者 5

テンプレート1		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	0	0.0%	7.3%
	2	4	7.3%	
	3	3	5.5%	5.5%
	4	30	54.5%	87.3%
	5	18	32.7%	
計		55	100.0%	

テンプレート2		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	1	1.5%	9.2%
	2	5	7.7%	
	3	11	16.9%	16.9%
	4	22	33.8%	73.8%
	5	26	40.0%	
計		65	100.0%	

表 7　被験者 6

テンプレート1		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	4	6.3%	19.0%
	2	8	12.7%	
	3	22	34.9%	34.9%
	4	19	30.2%	46.0%
	5	10	15.9%	
計		63	100.0%	

テンプレート2		推薦楽曲数	全推薦楽曲数に対する割合	
実嗜好印象	1	3	8.3%	27.8%
	2	7	19.4%	
	3	10	27.8%	27.8%
	4	10	27.8%	44.4%
	5	6	16.7%	
計		36	100.0%	

被験者全員に言えることではないが、全推薦楽曲数に対する割合については、実嗜好印象が 4 や 5 の割合は、1 や 2 の割合よりも高くなっている。またテンプレートの違いによるその割合の変化は、被験者 2 と被験者 5 を除いてほとんどみられない。

被験者 1 はどちらのテンプレートにおいても実嗜好印象が 5 の楽曲が推薦されていないが、これは 100 曲のうち実嗜好印象が 5 である楽曲がなかったためである。

次に、クロスバリデーションによる平均二乗誤差(実際の嗜好印象とその推定値から計算)を表にまとめた。

表 8　R プログラムでの k 近傍法における平均自乗誤差

	テンプレート1	テンプレート2	差(テンプレート2-1)
被験者1	0.8400	0.8000	-0.0400
被験者2	1.2633	1.1200	-0.1433
被験者3	0.9067	0.9067	0.0000
被験者4	1.0967	1.0533	-0.0433
被験者5	0.9133	1.0200	0.1067
被験者6	1.0333	1.0900	0.0567
平均	1.0089	0.9983	-0.0106

表 8 は R 言語によるプログラムでの平均自乗誤差をまとめたものである。平均自乗誤差の被験者 6 名の平均はそれほど悪いとは言えない。また、テンプレートによる平均自乗誤差の違いは、被験者 2 については減少、被験者 5 については増加しており、他の被験者と比較するとその変化が大きい。先程の全推薦楽曲数に対する実嗜好印象の高い楽曲を推薦する割合との影響を見てみると、被験者 2 の場合はそれが大幅に増加しているのに対し(表 3)、被験者 5 は大きく減少している(表 6)。

比較のため Weka により k 近傍法を適用した結果を次に示す。

表 9　Weka での k 近傍法における平均自乗誤差

	テンプレート1	テンプレート2	差(テンプレート2-1)
被験者1	0.7889	0.5584	-0.2305
被験者2	1.1667	0.8521	-0.3146
被験者3	0.8434	0.5890	-0.2544
被験者4	0.9850	0.6930	-0.2920
被験者5	0.8004	0.6508	-0.1496
被験者6	0.9612	0.7578	-0.2034
平均	0.9243	0.6835	-0.2408

表 9 は同じ被験者の Weka による平均自乗誤差である。これと表 8 を比較すると、Weka での k 近傍法の方がより良い性能であることがわかる。特にテンプレートの違いによる平均自乗誤差の変化が大きく、音楽的特徴を詳細に表わしているテンプレート 2 の方が良い結果を示した。

k 近傍法が楽曲推薦のための手法として適しているかどうかを検証するために、データマイニング手法の一つで、推定や予測に適しているニューラルネットワークによる手法を用い楽曲推薦を行った。Weka によりその手法を適用した結果を示す。

表 10　Weka でのニューラルネットワークにおける平均自乗誤差

	テンプレート1	テンプレート2	差(テンプレート2-1)
被験者1	0.8505	0.7810	-0.0695
被験者2	1.2513	1.1706	-0.0807
被験者3	0.9163	0.8718	-0.0445
被験者4	1.0338	1.0254	-0.0084
被験者5	0.8972	0.8523	-0.0449
被験者6	1.0372	0.9751	-0.0621
平均	0.9977	0.9460	-0.0517

表 10 は推薦手法の適用結果である。表 11 の k 近傍法と表 10 のニューラルネットワークを比較すると、テンプレート 1、2 両方において k 近傍法の方が良い結果を示した。

最後に被験者の実験時間の測定結果を表 9 としてまとめた。

表 11　実験時間

	実験時間（分）
被験者1	10
被験者2	8
被験者3	8
被験者4	7
被験者5	13
被験者6	15
平均	10.17

このように実験時間の平均は約 10 分となり、実験時間が短縮されている。

4-3　考察

まず楽曲推薦には k 近傍法が適しているということがいえた。これは Weka での k 近傍法とニューラルネットワークによる推薦手法の適用結果の比較において、k 近傍法の方が平均自乗誤差が小さいこと、そして使用したテンプレートによる違い、つまり音楽的特徴の違いが k 近傍法ではより大きく表れていることからいえる。

また、R プログラムと Weka での k 近傍法の比較では Weka がより良い結果を示したことから、用いられているアルゴリズムが異なっているものと思われる。

テンプレートによる違いは、特に Weka での結果に大きくみられ、楽曲推薦の性能には音楽的特徴が大きく関係し、詳細に特徴を表現することでその性能は向上すると考えられる。

なお、しきい値に関してはまずまずの結果が得られたため、しきい値の設定方法は有効であったと思われる。しかし、学習楽曲での実嗜好印象が一定の印象に偏る場合には、良い結果が得られないため、更なる向上が図れると思われる。

最後に実験時間についてだが、時間はかなり短縮されていると思われる。「ユーザへの負担」の点では目標を大きく達成したのではないだろうか。

5章　結論

本研究ではデータマイニング手法である記憶ベース推論のｋ近傍法を嗜好印象の推定値の計算手法として用い、推定値のしきい値処理による楽曲推薦システムを提案した。さらにR言語による楽曲推薦プログラムを作成し、実験により得た実嗜好印象データと楽曲の音楽的特徴を用いて、提案した手法による推薦性能の検証を行った。

実験の結果、作成したプログラムでは提案手法のｋ近傍法による高い推薦性能は認められなかったが、Wekaを用いた結果では楽曲推薦に適した手法であることがわかる。

今後の課題としては、R言語でのプログラムにおける推薦性能のさらなる向上があげられる。今回作成したプログラムの推薦性能がWekaのそれに遠く及ばず、あまり良い結果が得られなかった理由には両方で用いられるｋ近傍法のアルゴリズムの違いがあったと思われる。幸いなことに、Wekaはオープンソースであるため、それを解読し詳細のアルゴリズムを理解することができれば、性能向上が見込めるだろう。また、推薦のためのしきい値の設定方法には、改善の余地があるといえる。

もう一つの今後の課題として、テンプレート（音楽的特徴）の抽出の検討がある。実験により音楽的特徴を詳細に表わすことで高い性能を得られることがわかった。しかし、本研究で行った楽曲の音楽的特徴の評価は我々の個人主観によるもので、やや曖昧であるといえる。テンポや曲調などを分析し評価できるシステムを用いることで推薦性能を向上させることができると思われる。

6章　付録

6-1　データ入力用 PHP スクリプトのソースコード

ーindex.phpー

```
<html><body>
<center>

<form action="input.php" method="get">


<input type="hidden" name="nowsongNO" value="0"><!--現在の曲数を 0 としておく。次ページで$_GET['nowsongNO']
で使用。非表示-->
<br>
<big>ユーザネーム </big><input type="text" name="username"> <!--ユーザネームのボックス。次ページ
$_GET['username']。-->
<br>
<br>
<input type="submit" value="Enter">

</form>

<br>
※ユーザネームを入力してください。「ユーザネーム_result.csv」というファイルが作成されます。

</center>
</body></html>
```

ーinput.phpー

```
<?php


//もし 1 曲目(初回表示)なら、result.csv へ曲名の配列だけ出力。
if($_GET['nowsongNO'] == 0)
{
 $songname = file("hirosource/temp.csv");
 $allsongNO=count($songname); //temp.csv から全楽曲数$allsongNO を取得。
 $fp = fopen(("hiroresult/".$_GET['username']."_result.csv"),"w"); //result.csv に曲名のみを出力。
 for($i = 0; $i < $allsongNO; $i++)
 {
   fputs($fp,$songname[$i]);
 }
 fclose($fp);
}


//もし 1 曲目(初回表示)なら、ランダム順を発生後、order.csv を作成。
if($_GET['nowsongNO'] == 0)
```

```
{
 $i_max = round(rand(50,100));

 for($i = 0; $i < $allsongNO; $i++)
 {
  $order[$i] = $i;
 }

 for($i = 0; $i < $i_max; $i++)
 {
  $x = round(rand(0,($allsongNO - 1)));
  $y = round(rand(0,($allsongNO - 1)));

  $tmp = $order[$x]; $order[$x] = $order[$y]; $order[$y] = $tmp;
 }

 $fp = fopen("hirosource/".$_GET['username']."_order.csv","w");
 for($i = 0; $i < $allsongNO; $i++)
 {
   fputs($fp,$order[$i]);
   fputs($fp,"¥n");
 }
 fclose($fp);

}


//現在の曲名を取得。
$data_order = file("hirosource/".$_GET['username']."_order.csv");
$num = trim($data_order[$_GET['nowsongNO']]); //order.csv から nowsongNO 番目に処理する楽曲番号$num を取得。
$data_result = file(("hiroresult/".$_GET['username']."_result.csv"));
$nowsongname = trim($data_result[$num]); //num 番目の楽曲名を現在の曲名$nowsongname として取得。
$allsongNO = count($data_result);


//次のページのアドレス設定。全曲の印象入力が終わったら end.php へ。それ以外は本ページへリンク。
if( $allsongNO == $_GET['nowsongNO']+1 )
{
 $nextadd="end.php";
}
else
{
 $nextadd="input.php";
}


//印象を result.csv へ保存する。
if($_GET['nowsongNO'] != 0)
{
 $data_result = file(("hiroresult/".$_GET['username']."_result.csv"));
 $data_result[$_GET['num']] = (trim($data_result[$_GET['num']]).",".$_GET['impress']."¥n");
 $fp = fopen("hiroresult/".($_GET['username']."_result.csv"),"w+");
 for($i = 0; $i < $_GET['allsongNO']; $i++)
 {
   fputs($fp,$data_result[$i]);
```

```
 }

 fclose($fp);
}

?>



<html><body>

<form action=<?php echo $nextadd; ?> method="get">

<div align="right">
全<input type="text" size=2 name="allsongNO" value="<?php echo $allsongNO; ?>">曲中
<br>
現在<input type="text" size=2 name="nowsongNO" value="<?php echo $_GET['nowsongNO']+1; ?>">曲目 <!--現曲
数に１を足してボックスへ。次$_GET['nowsongNO']。-->
</div>

<center>

<embed src="<?php echo("http://127.0.0.1/data/".$nowsongname.".swf"); ?>"
       type="application/x-shockwave-flash"
       width="150" height="40"
       autostart="true"
       loop="true" repeat="true">
</embed>

<!-- "http://192.168.1.10/~kano/cgi-bin/data/" -->

<br>
曲名<input type="text" name="songname" value="<?php echo $nowsongname; ?>">
<br>
<br>

好き<input type="radio" name="impress" value="5"> <!--次$_GET['impress']。-->

まあ好き<input type="radio" name="impress" value="4">

普通<input type="radio" name="impress" value="3">

まあ嫌い<input type="radio" name="impress" value="2">

嫌い<input type="radio" name="impress" value="1">

<br>
<br>

<input type="hidden" name="username" value="<?php echo $_GET['username']; ?>">
<input type="hidden" name="num" value="<?php echo $num; ?>">

<input type="submit" value="次の曲へ" >
<br>
<br>
```

```
<br>
※「次の曲へ」のクリックは、ENTER で代用可。
<br>
※ブラウザの「戻る」は使わないでください。
</center>

</form>

</body></html>
```

―end.php―

```
<?php

//最終曲の印象を csv へ出力。
 $data_result = file(("hiroresult/".$_GET['username']."_result.csv"));
 $data_result[$_GET['num']] = (trim($data_result[$_GET['num']]).",".$_GET['impress']."¥n");
 $fp = fopen(("hiroresult/".$_GET['username']."_result.csv"),"w+");
 for($i = 0; $i < $_GET['allsongNO']; $i++)
 {
   fputs($fp,$data_result[$i]);
 }

 fclose($fp);

?>

<html><body>
<center>
作業終了です。ご協力ありがとうございました。
</center>
</body></html>
```

6-2　R 言語で作成した楽曲推薦プログラムのソースコード

```
library(EMV)

source<-read.table("C:/training.csv",sep=",",header=TRUE)

data<-matrix(0,100,10)
r<-matrix(0,100,5)
rr<-matrix(0,100,1)
rrr<-matrix(0,100,1)
real<-matrix(0,100,1)
e<-matrix(0,100,1)
diff<-matrix(0,100,3)



for(i in 1:100)
```

```
{
        for(j in 1:5)      data[i,j]<-source[i,j]
        data[i,5+source[i,6]]<-1
}

for(i in 1:100)
{
        help1<-data[i,6]
        help2<-data[i,7]
        help3<-data[i,8]
        help4<-data[i,9]
        help5<-data[i,10]

        data[i,6]<-NA
        data[i,7]<-NA
        data[i,8]<-NA
        data[i,9]<-NA
        data[i,10]<-NA

        result<-impute.knn(as.matrix(data),k=3)

        data[i,6]<-help1
        data[i,7]<-help2
        data[i,8]<-help3
        data[i,9]<-help4
        data[i,10]<-help5

        r[i,1]<-result[i,6]
        r[i,2]<-result[i,7]
        r[i,3]<-result[i,8]
        r[i,4]<-result[i,9]
        r[i,5]<-result[i,10]

        rr[i]<-1*result[i,6]+2*result[i,7]+3*result[i,8]+4*result[i,9]+5*result[i,10]
        real[i]<-1*data[i,6]+2*data[i,7]+3*data[i,8]+4*data[i,9]+5*data[i,10]

        diff[i,1]<-real[i]
        diff[i,3]<-rr[i]

        e[i]<-sqrt((real[i]-rr[i])^2)

}



pre<-matrix(0,100,100)
areal<-matrix(0,100,100)


for(i in 1:100)
{
        analy<-matrix(0,99,10)
        k<-1
        for(j in 1:100)
        {
```

```
                if(i != j)
                {
                        analy[k,]<-data[j,]
                        k<-k+1
                }
        }

        k<-1
        for(j in 1:99)
        {
                help1<-analy[j,6]
                help2<-analy[j,7]
                help3<-analy[j,8]
                help4<-analy[j,9]
                help5<-analy[j,10]

                analy[j,6]<-NA
                analy[j,7]<-NA
                analy[j,8]<-NA
                analy[j,9]<-NA
                analy[j,10]<-NA

                aresult<-impute.knn(as.matrix(analy),k=3)

                analy[j,6]<-help1
                analy[j,7]<-help2
                analy[j,8]<-help3
                analy[j,9]<-help4
                analy[j,10]<-help5

                if(i == j)        k<-k+1

                pre[i,k]<-1*aresult[j,6]+2*aresult[j,7]+3*aresult[j,8]+4*aresult[j,9]+5*aresult[j,10]

                areal[i,k]<-1*analy[j,6]+2*analy[j,7]+3*analy[j,8]+4*analy[j,9]+5*analy[j,10]

                k<-k+1
        }
}


frequ<-matrix(0,100,5)

for(i in 1:100)
{
        ave<-0.0
        for(j in 1:100)
        {
                if(i != j)
                {
                        if(areal[i,j] == 1)        frequ[i,1]<-frequ[i,1]+1
                        if(areal[i,j] == 2)        frequ[i,2]<-frequ[i,2]+1
                        if(areal[i,j] == 3)        frequ[i,3]<-frequ[i,3]+1
                        if(areal[i,j] == 4)        frequ[i,4]<-frequ[i,4]+1
                        if(areal[i,j] == 5)        frequ[i,5]<-frequ[i,5]+1
```

```
                ave<-ave+areal[i,j]
        }
}
ave<-ave/99.0


line<-5
max<--100
mline<-5
while(line >= ave)
{
        eff<-0
        for(j in 1:100)
        {
                f<-matrix(0,5,1)
                if(i != j)
                {
                        f[1]<-frequ[i,1]
                        f[2]<-frequ[i,2]
                        f[3]<-frequ[i,3]
                        f[4]<-frequ[i,4]
                        f[5]<-frequ[i,5]

                        if(areal[i,j] == 1)         f[1]<-f[1]-1
                        if(areal[i,j] == 2)         f[2]<-f[2]-1
                        if(areal[i,j] == 3)         f[3]<-f[3]-1
                        if(areal[i,j] == 4)         f[4]<-f[4]-1
                        if(areal[i,j] == 5)         f[5]<-f[5]-1

                        if(pre[i,j] >= line)
                        {
                                if(diff[j,1] == 3) eff<-eff+5*(1.0/f[3])
                                if(diff[j,1] == 4) eff<-eff+7*(1.0/f[4])
                                if(diff[j,1] == 5) eff<-eff+9*(1.0/f[5])
                        }
                        else
                        {
                                if(diff[j,1] == 1) eff<-eff+5*(1.0/f[1])
                                if(diff[j,1] == 2) eff<-eff+3*(1.0/f[2])
                        }
                }
        }

        if(eff >= max)
        {
                max<-eff
                mline<-line
        }

        line<-line-0.02

}

diff[i,2]<-0
if(diff[i,3] >= mline)      diff[i,2]<-1
```

```
}


comp<-0
for(i in 1:100)    comp<-comp+e[i]
comp<-comp/100


diff

true
```

7章　参考文献

[1]　岩井千明「インターネット音楽配信サービスの現状と問題点」

[2]　暦本純一「Universal Playlist: 利用者の嗜好に動的に適合するメディア再生機構」

[3]　「デジタル音楽リスナーを狙った検索サービス登場:ニュース - CNET Japan」<http://japan.cnet.com/news/media/story/0,2000056023,20062764,00.htm>

[4]　帆足啓一郎、上月勝博、菅谷史昭「楽曲配信サービスを支える音楽情報検索技術」

[5]　梶克彦、平田圭二、長尾確「状況と嗜好に関するアノテーションに基づくオンライン楽曲推薦システム」

[6]　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」

[7]　「データマイニング手法調査報告」<http://mikilab.doshisha.ac.jp/dia/research/report/2005/0712/001/report20050712001.html>

[8]　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 営業、マーケティング、CRM のための顧客分析』

[9]　U.リゲス、石田基広『R の基礎とプログラミング技法』

[10]　「RjpWiki」<http://www.okada.jp.org/RWiki/>

[11]　「PHP の使い方 【PHP 講座】」<http://www.sakura-pc.jp/php/02010000.shtml>

[12]　「初心者用 PHP 入門」<http://www.standpower.com>

[13]　大藤幹『プチリファレンス HTML』

[14]　「apache friends - xampp for Windows」<http://www.apachefriends.org/jp/xampp-windows.html>

[15]　「PHP エディタ - フリーの windows 用 php 統合開発環境 - phpspot」<http://phpspot.net/php/phpeditor.html>

[16]　石井一夫『図解よくわかるデータマイニング』

[17]　「k 近傍法 - Wikipedia」<http://ja.wikipedia.org/wiki/K%E8%BF%91%E5%82%8D%E6%B3%95>

[18]　「RWC 研究用音楽データベース」<http://staff.aist.go.jp/m.goto/RWC-MDB/index-j.html>

[19]　「weka-jp.info」<http://www.weka-jp.info>

謝辞

最後に、本論文を作成するにあたり、多くの人にお世話になり、ここで感謝の気持ちを表したい。

はじめに、本研究において、指導して下さった東海林准教授に感謝したい。そして、お互い助け合った研究室の仲間に感謝する。