特集号
招待論文

Jubatusの機能を利用した二者択一型不動産賃貸物件推薦サービスの開発

中野 猛[†1]　下垣 徹[†2]　橋本 拓也[†3]　渡邉 卓也[†4]

[†1] (株) リクルートテクノロジーズ　[†2] (株) NTTデータ　[†3] Acroquest Technology (株)
[†4] エヂリウム (株)

本稿では，オンライン機械学習を実現するためのミドルウェアであるJubatusのユースケースとして，不動産賃貸物件を題材とし，利用者の嗜好を反映させながら絞り込みを行い最終的に物件を推薦するサービスを開発した．このサービスでは物件の属性情報を基にMDS（多次元尺度構成法）を用いて探索空間を構築する．利用者の操作に応じ，その空間上で二分探索を基本的発想とする幾何的絞り込み，およびJubatusの分類器を利用した絞り込みを行い，探索空間を狭めていく．さらに3次のBスプラインによるcurve fittingを行い，絞り込みの収束点を予測した上で，利用者の嗜好に合致する物件一覧を提示する．

1. はじめに

1.1　オンライン機械学習の活用

近年の計算機の能力向上に伴い，機械学習を積極的に活用したサービスを実現するための取り組みが盛んに行われている．機械学習は音声認識や画像認識などのノイズ除去や特徴の抽出に使われるほか，ある商品の購入者はこの商品も購入しているといった推薦（協調フィルタリング）にも利用されている．

機械学習を利用する多くのケースではバッチ処理により学習させる．そこで得た結果を取り出し，オンラインでの評価に利用するのが一般的な適用方法である．しかしながら，バッチ処理による学習では入力に必要とする大量のデータを保持する必要がある，または処理するための大規模な環境が必要になるなど，処理にあたっての種々のコストが大きいことや，バッチ処理に一定の時間を必要とするため，学習結果をサービスに反映させるまでのタイムラグが存在する，といったさまざまな課題がある．

そこで，学習をオンラインで行うことで，それらの問題を解消するオンライン機械学習が注目される．オンラインで学習させることで，その場限りの利用者のアクションを学習結果に反映させることができ，短いタイムスパンで利用者個別にフィードバックをかけることができる．

1.2　Jubatusについて

Jubatus [1]はNTTソフトウェアイノベーションセンタと（株）Preferred Infrastructureが共同で開発したオンライン機械学習を実現するためのミドルウェアである．大量のデータを処理するための基盤技術としては，Googleが発表した論文をベースにOSSとして開発されたHadoop [3]があり，これをベースにした機械学習ライブラリであるMahout [4]が存在する．しかしHadoopはバッチ処理ベースのシステムであるため，結果を得るまでに一定の処理時間を必要とする．適切なアクションを適切なタイミングで行うためには，よりリアルタイム性が求められるシーンがあり，Jubatusはそのために開発された．2011年に最初のバージョンがリリースされて以降，機能拡張が進められており，現在では「分類」「推薦」「回帰」「グラフ処理」「異常値検知」「クラスタリング」といった機能を備えている．

Jubatusの分類器はJubatusの初版リリースから実装されている機能である．これに対する検証の結果，利用するにあたってのCPU使用率やメモリ消費量などのフットプリントが小さいことや，学習を行った際に比較的安定して動作するものであることを確認できたことから[2]，本稿で紹介するサービスではこれを利用している．

本稿の内容のうち，Jubatusを利用した処理については3.2.3節で記述している．また，Jubatusの分類器のシステム中での位置付けやプロセスの管理方法については

4.2 節と 4.3 節で述べる．

1.3 不動産の賃貸物件提示サービスにおける事業課題

実際にオンライン機械学習を適用するユースケースとして，不動産の賃貸物件を提示するサービスについて考える．賃貸物件をインターネット上で検索できるサービスがいくつか提供されているが，不動産会社に直接足を運んで会話しながら物件を探すのとは異なるため，あいまいな選好しか持たない場合，既存の検索サービスでは効率的に探せない利用者が一定数いるものと考えられる．また，賃貸物件を探す利用者は，会員登録を行わずに検索を行うことが多く，利用者のプロファイルを事前に入手したり，過去の行動履歴を分析したものを検索結果に反映させることが難しい．このような利用者に対して効果的にアプローチすることが望まれる．さらに，昨今のスマートフォン環境の成熟に伴い，スマートフォンで賃貸物件を探す利用者が増えている．スマートフォンからのアクセスでは細かい希望条件を逐一入力することが PC 環境に比べると困難を伴うため，スマートフォンでのユーザビリティの改善が課題として挙げられる．

こうした課題に対応可能なサービス実用化の目処をつけることを目標とし，数人の開発者により，予備調査期間を除きサービス開始まで約 1 カ月半の期間で設計・実装したのが本サービスである．ただし，サービス開始後も継続的に改良は行っており，本稿の記述はサービス終了時点での版を対象としている．

2. 基本構想

2.1 基本的な処理フロー

今回開発した二者択一型不動産賃貸物件推薦サービスの基本的な処理フローは，提示物件の探索空間より特徴的な 2 つの物件を選択して**図 1** のように利用者に提示し，利用者より選択結果を得て Jubatus に学習させ，選択結果および Jubatus の学習結果に基づいて探索領域の絞り込みを行うというものである．一定回数の試行の後，利用者の嗜好を反映した物件の検索条件を提示する．その際，残り試行回数の明示による利用者の負担軽減を考慮し，探索空間を二分探索的に探索することによる 8 回以内での収束を目指した．それを実現するアルゴリズムについては，できるだけ例外状況が原理的に発生しないような頑健かつ保守性の高い方式となるよう，設計を行った．

図 1 開発した物件提示サービス「" あなた好み " の条件をレコメンド」

2.2 探索空間の構築

対象物間の関係を低次元空間（実用としては 2 または 3 次元）における点の布置で表現する手法として multidimensional scaling（MDS）[5] がある．本サービスでは，賃貸物件の属性情報を，MDS によって 2 次元空間に落とし込んで探索空間とした．物件間の特徴空間上の距離は 2 次元空間上でも保存されるため，似ている物件は近くに配置されるという特性がある．この特性によって，3.2.2 節で説明する，探索領域の幾何的操作による絞り込みで不要な物件を切り離すことが可能となった．

3. 処理方式

3.1 物件データの変換法

本サービスでの学習・探索に利用する物件データは以下の 4 属性からなる（利用者への表示用データにはより多くの属性が含まれる）．

- 賃料
- 面積
- 築後年数
- 駅からの所要時間

この物件データにつき，リアルタイム処理に適さない

以下の2つの処理を前処理として行っている．

3.1.1　分類器向けの変換

Jubatusの分類器へ投入する物件データの各属性について，ダミー変数を導入する変換を行う．これは，Jubatusの分類器はperceptron [13]およびその亜種であり，投入されるデータは基本的に線形分離可能である必要がある一方，物件データは線形分離可能ではないためである．たとえば，賃貸物件の賃料について個々の利用者の嗜好に合致するか判定する場合を考えたとき，それが線形分離可能であるためには，利用者が，ある額以上はいくらでもよく，それ未満には興味がないといったような嗜好を持つ必要があるが，これが現実的な嗜好を表現しているとは言いがたいであろう．

そこで，まず各属性についてダミー変数を導入し，フラグ化する．たとえば，物件の賃料が12.8万円の場合，「賃料12万円から14万円」を表す変数の値が1，ほかは0となるよう変換する．これにより，区間幅の粒度で利用者の嗜好に合致するかを判定することが可能となる．

実際の区間幅hはヒストグラム作成時の区間幅決定法の1つであるScott's normal reference rule [6]を用い，以下のように属性値のばらつきに応じた適切な幅に設定する．

$$h = \frac{3.5\,\hat{\sigma}}{n^{1/3}}$$

ここで $\hat{\sigma}$ は属性値の標準偏差，nは物件数である．Scott's normal reference ruleは，正規分布したデータからのランダム標本に対して適用した場合に，密度推定における平均2乗誤差の積分を最小にする．

その後，導入したダミー変数につき，近接k変数（両隣kずつ，実装では2を採用）への減衰を伴う伝播処理を行う．このとき，減衰の仕方を区間中央値からのずれに応じて，左右非対称にする．具体的には，ある属性の値がxのとき，その値の属する区間を表す変数v_iがカバーする区間の最小値をv_i^{min}，最大値をv_i^{max}，中央値をv_i^{med}とすると，各区間の変数の値を，

$$v_j = \begin{cases} 1 & \text{if } i = j, \\ 1 - \dfrac{|i-j| + \mathrm{sign}(i-j)\,\alpha}{k+1} & \text{if } |i-j| \le k, i \ne j, \\ 0 & \text{otherwise,} \end{cases}$$

$$\text{where } \alpha = \frac{x - v_i^{med}}{v_i^{max} - v_i^{min}}$$

により計算する．これは，ある区間を好む利用者はその近接区間もいくらかは好むであろうという自然な予想を表現したものであり，またoverfittingを防ぐための措置でもある．

3.1.2　MDSによる探索空間の生成

上記分類器向け物件データの変換とは独立に，探索用の空間を生成するため，MDSを用い，4属性からなる物件データを2次元空間にマップする．探索空間は駅ごとに，その駅を最寄駅とする物件の集合から生成する．処理は以下の順で行う．

1. 各属性について平均0，分散1に正規化
2. MDSを用いて2次元データに変換
3. 座標を最小値0，最大値1となるように正規化

また，探索空間に付随する情報として，属性ごとの回帰平面の回帰式を求める．そのために，各属性を従属変数，探索空間の座標を独立変数とする重回帰分析を行う．この情報は後に提示物件の属性値の妥当性を検証する際，また検索条件生成時に探索空間上の座標が属性値としてはどういった値に相当するのかを計算する際に用いる．

3.2　二分探索による高速収束方式

本サービスでは，利用者の賃貸物件に関する嗜好について，MDSにより生成した探索空間につき二分探索的手法によって積極的に探索空間を狭めることで，属性値の区間の数nに対して$O(\log n)$の操作で推定することを目標としてアルゴリズムを設計した．

基本的な考え方は次のようなものである．利用者には2物件提示し，どちらか好みの方を選択してもらう．この際に提示する物件は，探索空間上で十分に離れたもの同士とし，選ばれなかった側付近は空間から切り離す．たとえば，0から100の間では，まず20と80の物件を提示し，20が選択されたら50以上を切り離して次は10と40を提示する．

この考え方を複数属性の場合に適用すると，ある属性については希望どおりだが，別の属性については希望と合致しないということがあり得る．その場合にも正しく利用者の嗜好を判別できるよう，提示する物件ペア間の属性の値同士の大小関係について，複数の組合せのものを用意する必要がある．

このとき，全属性についてその組合せの数だけ提示するのは非現実的なため，探索はMDSにより生成した2次元空間上で行う．つまり，提示物件は元の属性空間ではなく探索空間上での位置関係から選定するのであり，詳細については3.2.1節で説明する．また，この提示物件の位置関係に基づく絞り込みを幾何的絞り込みと呼び，3.2.2節で説明する．ただし利用者に提示するのはあくまで物件の属性情報であり，探索空間上の布置の

情報は与えない.

探索空間はあらかじめ一定の大きさのタイルに分割して利用する．本サービスでは近傍探索や探索空間の切り離しなどをタイル単位で行うことで，利用者の操作に対するリアルタイムな応答を実現している．**図2**にタイル分割後の実際の探索空間の例を示す．初期状態において白いタイルが物件の存在するタイル，灰色のタイルが物件の存在しないタイルである.

これらタイルそれぞれについて，そこに含まれる物件がどの程度利用者の好みと合致するかを算出することによる絞り込みも行う．算出は利用者の選択結果を分類器に投入することにより行い，前述の幾何的絞り込みに続いて実施する．これが分類器による絞り込みであり，3.2.3節で説明する.

3.2.1　提示物件の選択方式

提示物件を選択する際には，まず，その際の基準となる軸を求める．軸は探索空間上の物件分布に対して主成分分析を行うことにより求め，第1主成分に対応する軸を長軸と呼ぶ．**図3**におけるように物件分布が楕円形であるとき，長軸は楕円の長軸に相当する.

次に，軸に近接し，また互いに適度に離れた物件ペアを求める．このために，まず軸上から物件選定の基準となる点を選ぶが，実装ではこれを軸方向の物件分布での20%点と80%点とした．この基準点の含まれるタイル中に物件が存在した場合はそれを選択する．もしタイル中に物件がなかった場合には1つずつ外側のタイル（Chebyshev距離で1ずつ離れたタイル）を順に，見つかるまで調べる.

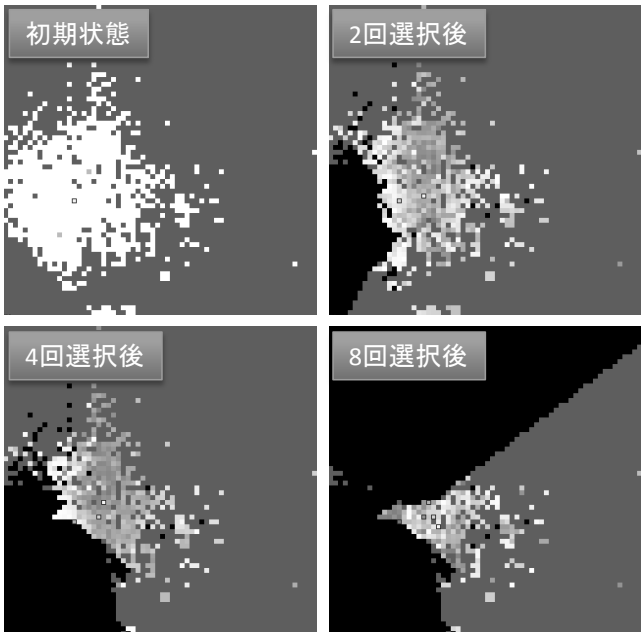

図 2　探索領域の絞り込みの様子

次回提示する物件ペアは，物件分布の重心を通り，長軸に直交する軸から同様に求める．この軸を短軸と呼ぶ．物件の提示は，基本的にこの長軸と短軸を基準に選んだ2ペア4物件を組として行う.

この過程で選ばれる個々の物件は，空間の全体的特徴と離れた属性値を持つことがあり得る．そこで，提示する物件として適切かどうか，属性ごとにその値の大小関係が回帰平面の勾配と一致しているか（共に上りあるいは下り勾配なのか，それとも逆の勾配なのか）検査する．この検査は物件ペア単位で行い，それらの各属性値の差の符号について，回帰平面におけるそれら物件の座標間の属性値の差の符号と一致しているか調べる．その結果一致していなかった物件ペアについては提示対象としない．ただし，提示物件を選ぼうとしているタイルに属する物件が1つだけの場合は検査は行わず，常に提示対象とする.

3.2.2　探索領域の幾何的絞り込み

利用者により，提示した物件のどちらか一方が選ばれた後に，その選択・非選択物件の座標情報を利用して探索領域の絞り込みを行う．この絞り込みは探索領域についてタイル単位で「線を引き」「色を塗る」という幾何的操作により実施する．具体的な手順は以下のようなものである.

1. 長軸・短軸それぞれで非選択となった2物件の座標から，以下の3つの線分を求める
   1.1. 2物件間を結ぶ線分
   1.2. 各物件から，それぞれの基準となった軸に平行に外側に伸ばした線分
2. 線分はBresenhamのアルゴリズム[8]によりタイルに対応させる（量子化）

これら線分と領域全体の外周により，2つの部分

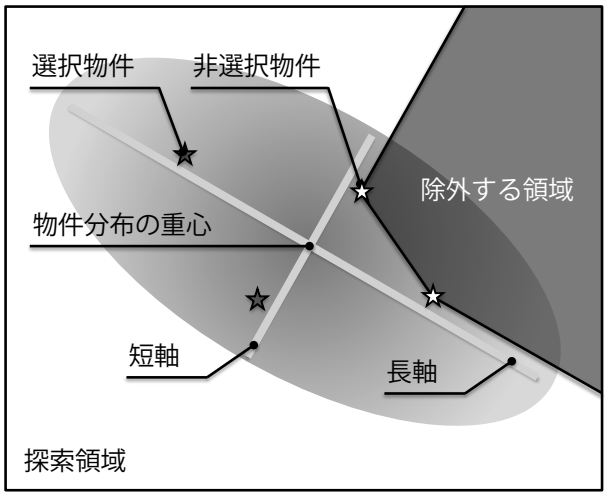

図 3　提示物件選択および幾何的絞り込みの概念図

領域が形成される

3. どちらかの領域を（線分とは異なる「色」によって）flood fillする
4. 以下の方式で「外側」を決定し，切り離す

4.1. 中心点および選択された2物件の座標（3つの座標）のうち，含まれる数の少ない領域を外側とする（多数決）

4.2. 外側に属するタイルおよび線分を構成するタイルを切り離す

4.3. 中心点などが線分上に存在し，外側に位置する点と内側に位置する点が同数となった場合には，線分を構成するタイルのみ切り離す

線分とそれにより形成される領域の例については図3を参照されたい．

ここで実現したいことは，外側の領域に属するタイルを決定し，それらのタイルを切り離すことである．もし非選択物件が必ず軸上に乗っているのであれば，上記線分により形成される外側の領域は必ずconvex（凸）となるので，どの領域が外側なのかを決定するのは容易である．しかし，物件は軸上ではない点からも選ばれるため，外側の領域の形状はconcave（凹）になることもあり，どちらが外側かを単純に決定することはできない．また，軸の中心点が外側の領域に含まれることもあり得るので，中心点から線分への交線を引いて外側を判定するといった方式も採ることはできない．外側の領域の面積の方が大きいこともあり得るので，面積の大小も判定には用いることができない．

一方，上記の線分の引き方によって保証されている性質は，それにより必ず領域全体は二分割される，ということである．よって，まずどちらかの領域をflood fillにより塗りつぶせば，それぞれのタイルが2つのうちどちらの領域に属しているのかを一意に決定することができる．その上で，残したい3つの点（中心点，2つの選択物件の点）がより多く含まれる領域を「内側」と考える多数決によって，その逆側の領域としての「外側」の領域を決定する．

探索領域から切り離すのは,「外側」に属するタイルと，線分を構成するタイルである．線分を構成するタイルを切り離すことで，非選択物件を確実に探索領域から除外することができる．つまり，非選択物件およびその類似物件の履歴を用いた継続的除外を，間接的に実現していることになる．図2において選択後に黒くなっているタイルが探索領域から切り離されたタイルである．

3.2.3　探索領域の分類器による絞り込み

探索領域の絞り込みの際には，Jubatusの分類器の重みベクトル（学習結果）の情報を利用した，幾何的な絞り込みとは異なる原理による絞り込みも行う．分類器には，利用者の選択ごとに，選ばれなかった物件を負例として，選ばれた物件を正例として（その順番で）投入することで学習を行う．この学習により形成された分類器内部の（ダミー変数化された）各属性への重みのベクトルを取得し，その情報を基に切り離すタイルをタイル単位で決定する．なお，分類器のアルゴリズムとしては，passive-aggressive algorithm [14]を利用した．

切り離すタイルを決定するにあたっては，まず探索空間中の各物件の重みを，重みベクトルから決定する．物件の各属性につき，それをダミー変数化した後の値が1となっている区間の，重みベクトル中の対応する区間の重みを集め，その全属性に渡る平均を物件の重みとする．なお，重みベクトルはあらかじめ正規化しておく．

次に，タイルの重みをそのタイルに属する物件の重みから求める．この際の重み付けにはinverse distance weighting (IDW) [15]を用いる．すなわち，タイルの中心座標とタイル中の物件との距離を求め，距離の逆数による物件の重みの重み付き平均をタイルの重みとする．タイルの中心座標がx，各物件の重みがu_iであるとき，タイルの重み$u(x)$は以下の式により求める．

$$u(x) = \sum_{i=0}^{N} \frac{w_i(x)u_i}{\sum_{j=0}^{N} w_j(x)}$$

重み付け関数w_iとしては，実装ではユークリッド距離の逆数を用いている．図2では各タイルの重みを色で表現しており，0は白，負は青，正は赤に，また絶対値が大きいほど濃くなるよう色付けしている．

タイルの切り離しは以下の2条件を満たした場合に行う．

- タイルの重みが負の値である
- タイルの重みが，探索領域における重み分布の値域の下位20%に含まれる

たとえばタイルの重み分布の値域が[-0.5, 1.0]であった場合は，重みが[-0.5, -0.2]に含まれるタイルが切り離しの対象となる．図2において散発的に分布する黒いタイルがこれによって切り離されたタイルである．

3.3　利用者の選択結果に基づく検索条件生成

本サービスでは規定回数（8回）利用者が操作を行った後に，その過程で学習した，利用者の好みだと推定さ

れる物件の一覧を表示する．一覧表示には既存サービスの部品を流用しており，そのため，一般の利用者が賃貸物件を検索する際と同じ粒度で検索条件を指定することにより一覧表示を行っている．学習により推定する利用者の好みの属性の範囲は実数値区間なのに対して，検索条件は離散値で指定するようになっており，また，上限と下限の両方を指定可能な属性と上限のみを指定可能な属性があるなど，属性によって指定可能な条件の特性が異なるので，それに応じた変換を行っている．

3.3.1 基本方式

本サービスにおいては，以下のように基本的には属性値に対する回帰によって検索条件を生成する．まず，探索空間における各属性値の回帰平面にその時点で残存する（物件が属し，かつ切り離されていない）タイルの領域をあてはめ，対応する属性値の範囲を算出する．**図4**はこれによる属性値範囲算出の例である．図4左では図の見やすさを優先し，座標を1次元で表現しているが，実際には，残存するタイルの領域につき回帰平面の法線ベクトル方向の両端の座標を求め，その2次元座標からの重回帰分析を行っている．

その後，可能な場合には，検索条件に該当する物件数が目標とする上限値・下限値の範囲に収まるように，条件を物件分布の重心に向かって狭めていく．これは，推薦サービスとしての実用性を確保するためには利用者にとって適度な数の選択肢を提示する必要があると考えたためである．

より詳細な方式は以下のようなものである．

1. 探索空間から，残存するタイルを集める
2. 上記タイルにつき，回帰平面より求めた属性値の範囲r_1（図4の属性値の範囲に相当）と，タイルに所属する物件の属性値の最小値・最大値を集計することにより求めた範囲r_2を得る
3. r_1を検索条件に変換して該当物件数を求め，該当物件数が0件だった場合は，r_2を検索条件に変換し，該当物件数を求めて終了する
4. r_1の上限・下限値を5%ずつ削り，それを検索条件に変換，該当物件数を求め，以下の処理を行う

4.1. 該当物件数の下限を下回った場合は前回の条件を採用して終了

4.2. 該当物件数の上限を下回った場合はその条件を採用して終了

4.3. 45%ずつまで削っていた場合はその条件を採用して終了

4.4. 削る割合を5%ずつ増加させて繰り返す

r_1は回帰分析によって求めた範囲のため，実際に該当する物件が存在しないことがあり得，その場合にr_2を採用することで物件一覧表示の際には必ず1つ以上の物件を表示するようになっている．

検索条件からの物件数の算出は，初期化時に作成した整列済みのデータ構造を利用した二分探索により行っている．そのため，計算時間は物件数nに対して$O(\log n)$である．

3.3.2 物件数の表示と絞り込み

利用者の満足度を向上させるためには，自分の選択の結果が一回一回着実に物件の絞り込みに繋がっているという感触を与えることが重要と考えられる．そこで，当該駅の全物件数を初期値として，そこから操作ごとに物件数が単調減少するように毎回検索条件を生成し，その条件にあてはまる物件数を画面表示することで学習過程の可視化を行った．

この過程は物件数につき以下の制約条件の下で進行することとした．

- 初期値は当該駅の全物件数
- 最終的な（8回操作後の）物件数は設定された上限値と下限値の間に収まる
- 単調減少する

この制約に従い，単調かつ最終的な物件数に滑らかに接続するような物件数の減少のさせ方を工夫した．

物件数の絞り込みは，基本的には前節の検索条件生成方式における，物件数の上限値・下限値を選択回数に応じて引き下げることで行う．またそれに加えて，前回提示した物件数を上回る件数を提示しないよう，以下の方式に従って処理を進める．

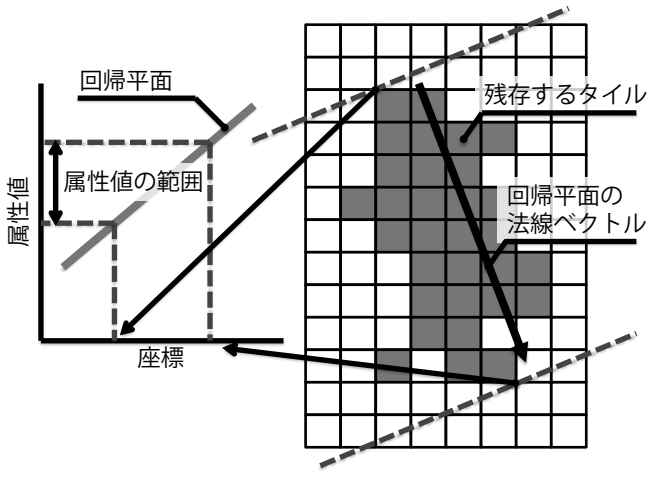

図4　回帰による検索条件生成の概念図

1. 初回は空の検索条件を返す（物件数は当該駅の全物件数となる）
2. 2回目以降は，物件数の上限値・下限値を，選択回数に応じて徐々に引き下げていく

2.1. 各回の範囲の上限につき，以下の式で算出する

$$(\text{全物件数} - \text{最終の上限物件数}) \times \left(\frac{\text{残り選択回数}}{\text{最大選択回数}}\right)^2 + \text{最終の上限物件数}$$

2.2. 下限については，上限から以下の式で算出する

$$\text{最終の下限物件数} \times \frac{\text{上限物件数}}{\text{最終の上限物件数}}$$

3. さらに，検索条件の範囲を狭めた結果の物件数が目標範囲の上限を下回っていても，前回の物件数を上回っている場合は継続して狭める

前回提示した物件数を上回っている場合に継続して狭めるのは単調減少制約を満たすためだが，範囲の下限を下回った場合はやむを得ないので下回る前の値を採用する．

3.3.3 学習の収束点の推定

検索条件を生成するにあたってはその時点で探索空間に残っている物件の分布の重心を中心とした範囲を用いている．学習が十分に収束していれば物件分布の重心が利用者の好みに合致しているはずであるが，ユーザインタフェース上の要請により8回の選択で必ず打ち切るため，物件の分布にもよるが，十分に収束しないまま打ち切りとなる蓋然性が高い．その段階では必ずしも探索を続けていれば収束したであろう点に重心があるとは限らず，そのことが生成される検索条件の精度にも影響するため，それまでの重心の移動履歴から，重心の収束先の点を推定することによって利用者の好む条件をより精度よく生成することを試みた．

推定に先立ち，まず，探索空間上の物件分布の重心の履歴（x, y座標のリスト）を記録しておく．この際の重心については分類器から取得した重みベクトルから算出した各物件の重み（3.2.3節参照）による重み付けを行った上で計算する．

その履歴に対し，最後の（物件一覧表示に用いる）検索条件生成時，3次のBスプライン（cubic B-spline）[9]によるcurve fittingを行う．fittingは，x, y座標それぞれにつき，履歴中の番地（0, 1, 2, ...）をu，座標をvとする$u-v$平面上で行う．これによって生成したスプラインを用い，x, y座標それぞれにつき，uとして履歴の最後の番地+1を指定したときのvの値を外挿することにより収束点を推定する．

表1　バッチ処理用サーバに導入したソフトウェア

ソフトウェア名	版
R	3.0.1
psych	1.3.2

表2　オンライン処理用サーバに導入したソフトウェア

ソフトウェア名	版
Python	2.7
mod_wsgi	3.4
web2py	2.4.6
SciPy	0.9.0
Jubatus	0.4.2

4. システム構成

4.1 全体構成と利用したミドルウェア

本サービスではバッチ処理用のサーバとオンライン処理用のサーバの2つのサーバを用意した．バッチ処理用サーバではPythonスクリプトやRスクリプトを用い，日次バッチ処理にて物件データの分類器向けの変換やMDSによる探索空間の構築を行う．オンライン処理用サーバではWebアプリケーションによりこの処理結果をセッション開始時にロードし，利用者の操作に応じた空間の探索を行う．サーバ側アプリケーションはPython，クライアント側はJavaScriptによる実装とした．両サーバについてはいずれもAmazon EC2 [16]を利用して構築し，OSはAmazon Linux（Amazon Linux AMI 2012.03）である．

それぞれのサーバで導入したソフトウェアパッケージやミドルウェア類のうち代表的なものを**表1**と**表2**に示す．このほか，1つ1つ列挙はしないが，各パッケージの依存パッケージについても導入している．利用したソフトウェアの選定にあたっては，Jubatusを除き，すでに広く利用されており，導入や基本機能についての習得が容易であること，同時に高度な機能が必要な場合には利用可能であることを重視した．

4.2 アプリケーション構成と処理フロー

Webアプリケーションのフレームワークには web2py [12]を利用し，アプリケーション内で用いる各種統計手法用のライブラリとしてSciPy [7]を利用した．また，Apache [10]に mod_wsgi [11]を導入し，web2pyを web server gateway interface（WSGI）を用いてApacheと連携して運用することとした．この構成については**図5**に示す．

利用者が提示された物件を選択する操作を行った場合

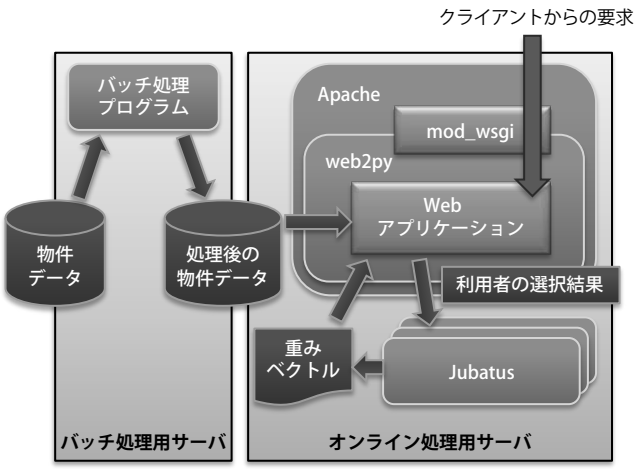

図 5　本システムの構成

の処理フローは以下のようなものである.

1. Apacheによりクライアントからの HTTP リクエストが受け付けられる
2. リクエストが WSGI を通じて web2pyに渡される
3. web2pyによりアプリケーションのハンドラが呼び出される
4. ハンドラでは以下の処理を行う

4.1. 利用者の選択内容を RPC（Remote Procedure Call）により Jubatusの分類器へ投入し，学習させる
4.2. 分類器内部の重みベクトルをファイルに書き出させる
4.3. 上記ファイルを読み込む
4.4. 探索領域の絞り込みを行う（3.2.2 節の幾何的絞り込み, 3.2.3 節の分類器による絞り込みの順に行う）
4.5. 検索条件を生成し，該当物件数を算出する
4.6. 次回提示物件を取得する（3.2.1 節の方式による）

5. 検索条件や次回提示物件などが web2pyを経由し Apacheからクライアントに返される

分類器の重みベクトルを取得する際に，RPCにより直接取得しないのは，現在の Jubatusにその機能がないためである．RPCによりファイルに出力させることはできるので，その内容を読み込んで利用している．読み込み部分は Jubatusの関数をそのまま利用するために C++で記述しており，それを Python の foreign interface を経由して呼び出すことで処理を行っている.

4.3　分類器プロセスの管理方法

本サービスでは Jubatusの分類器プロセス（インスタンス）を各利用者に割り当て，各分類器は利用者個々の操作結果につき，独立に学習を行う．したがって，最大同時セッション数だけの分類器プロセスを同時に起動し，管理しなければならない．アプリケーションは分類器プロセスのプールを用意し，セッション開始時にプールから 1 つのプロセスを選択して割り当てる．セッション終了時にはセッションに割り当てたプロセスを解放するが, この際に学習内容をクリアしてからプールに戻し，再利用する.

本サービスのようにセッションごとに分類器プロセスを割り当てる方式の場合，それによるメモリ不足や性能の低下が問題となり得る. Jubatusの分類器は軽量のため, 1 ノードで数 100 の分類器プロセスを常駐させることができる．また，負荷試験において 100 プロセスを同時に起動した場合でも応答時間の顕著な低下は発生しないことが確認できた.

一方，セッションがプロセスに対応することによる管理上の課題として，明示的に終了しないセッションに関する問題があった．閉じられないままのセッションが累積すると，分類器プロセスは簡単に枯渇してしまう．このため，アプリケーションは定期的にセッションの状況をチェックして，一定時間アクセスのないセッションを破棄し，分類器プロセスを回収するようにした．破棄したセッションに対してアクセスがあった場合は，利用者にタイムアウトが発生した旨を表示して，選択を初めからやり直してもらうものとした.

分類器プロセスが異常終了した，あるいは動作を受け付けなくなったなどの異常系に対しては，いくつかの対策を講じている.

まず，セッション開始時にプロセスを割り当てる際には，そのプロセスが生存していることを確認する．割り当てようとしたプロセスが終了していた場合は, 別の（生存している）プロセスを割り当てるとともに，終了していたプロセスに代わる新たなプロセスを起動する.

プロセスとの RPC 時には常にその成否を確認し，失敗した場合は，そのセッションではそれ以降分類器を用いないで探索領域の絞り込みを進める．利用者には特に通知は行わず，分類器の重みベクトルを利用した処理だけが無効となった状態でその他の処理を進める.

5. おわりに

本サービスは 1 カ月あまりの期間，実際に一般の利用者に対して，POC（Proof Of Concept）という位置付けの下提供された．限られた期間と流入量ではあったものの，通常の検索条件指定による既存サービスをやや上回るコンバージョン率を記録するなどの効果が観測され

た．これは，検索条件を直接指定することなしに，わずか8回の二択で数千件にのぼる多次元データから利用者の好みに合致する数十件にまで絞ることができるという，本サービスの今までにない特徴が評価されたことによるものと考える．

将来的には，現在実装している属性情報から見た最適値の提示にとどまらず，過去データの活用ならびに類似した利用者の挙動の反映や，物件データに付随する画像データおよび解説文の解析をも加味した新機軸の提示など，より新しく，検索サービスレベルの向上に寄与する開発を進めていきたい．また本サービスで採用した絞り込みの方式は幅広い分野に適用可能なものであり，他分野への応用の可能性についても探っていきたい．

参考文献

1) 岡野原大輔，海野裕也，熊崎宏樹，小田　哲：大規模リアルタイム解析エンジン Jubatus の創り方，デジタルプラクティス，Vol.4, No.1, pp.20-28 (2013).
2) 世界征服を目指す Jubatus だからこそ期待する 5 つのポイント，http://www.slideshare.net/hadoopxnttdata/jubatus-5
3) Hadoop, http://hadoop.apache.org/
4) Mahout, http://mahout.apache.org/
5) Cox, T. F. and Cox, M. A. A. : Multidimensional Scaling, Second Edition, Chapman and Hall (2001).
6) Scott, D. W. : On Optimal and Data-based Histograms, Biometrika 66 (3), pp.605-610 (1979).
7) SciPy, http://www.scipy.org/
8) Bresenham, J. E. : Algorithm for Computer Control of a Digital Plotter, IBM Systems Journal 4 (1), pp.25-30 (1965).
9) De Boor, C. : A Practical Guide to Splines, Springer-Verlag (1978).
10) Apache, http://httpd.apache.org/
11) mod_wsgi, http://code.google.com/p/modwsgi/
12) web2py, http://www.web2py.com/
13) Rosenblatt, F. : The Perceptron : A Probabilistic Model for Information Storage and Organization in the Brain, Psychological Review 65 (6), pp.386-408 (1958).
14) Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S. and Singer, Y. : Online Passive-aggressive Algorithms, Journal of Machine Learning Research (2006).
15) Shepard, D. : A Two-dimensional Interpolation Function for Irregularly-spaced Data, Proceedings of the 1968 ACM National Conference, pp.517–524 (1968).
16) Amazon EC2, http://aws.amazon.com/ec2/

中野 猛（非会員）tf0054@r.recruit.co.jp
1976 年兵庫県宝塚市出身．奈良先端大学院大学より（株）リクルートに入社．分社化後は（株）リクルートテクノロジーズに所属，シニアリサーチャー．Hadoop や検索エンジン Solr 導入や，負荷対策ミドルウェアの開発／導入など，オープンソースを中心とした Web 技術の革新を幅広く推進．その間，R25 等 Web サイトのインフラ構築／アプリ開発／運用にも行う．現在はリアルタイム処理の汎用化や M2M に興味を持つ．

下垣 徹（非会員）shimogakit@nttdata.co.jp
京都大学大学院情報学研究科システム科学専攻修士課程修了．（株）NTT データ所属．長年に渡り PostgreSQL を中心としたオープンソースの DBMS に取り組む．本体拡張機能の開発や商用 DBMS から PostgreSQL への移行案件に従事してきた．近年の大規模なデータを処理するニーズに対し，DBMS と Hadoop をはじめとする OSS の並列分散処理技術の両者の特徴を活かした組合せの実現に注力している．

橋本 拓也（非会員）hashimoto@acroquest.co.jp
早稲田大学理工学研究科情報科学専攻卒業．Acroquest Technology（株）所属．Java のトラブルシューティングを中心にしたシステムエンジニアとしての経験を経て，OSS を活用したソフトウェア基盤の検証・導入等に取り組む．

渡邉 卓也（非会員）sodium@edirium.co.jp
2011 年東京大学大学院総合文化研究科博士課程単位取得退学．同年ヱヂリウム（株）を設立，代表取締役社長に就任．言語学調査の統計解析，通信監視システムの設計・実装，経路制御アルゴリズムの設計等に携わる．数理的手法による問題解決全般に興味があり，最近は特に整数と実数の境目の辺りが面白いと思っている．日本ソフトウェア科学会会員．2007 年日本ソフトウェア科学会高橋奨励賞受賞．

投稿受付：2013 年 12 月 18 日
編集担当：吉野松樹（(株）日立製作所）