分類マイクロタスクにおけるタスク順序制御に関する研究

筑波大学
図書館情報メディア研究科
2017 年 3 月
根本 千代之介

目次

第 1 章

はじめに

クラウドソーシングは多数の人間の力を必要とする業務に適したアプローチであり，様々な場面で用いられている．例えば，自然言語処理における正解データの作成や，画像のタグ付けなどにクラウドソーシングは用いられている．クラウドソーシングの中でも，短時間で処理できる作業を扱うものをマイクロタスク型クラウドソーシングとよぶ．

マイクロタスク型クラウドソーシングにおける典型的な作業の一つとして，データの分類がある．例えば，図 1.1 に示すタスクは，災害時における被災家屋の発見を目的としたもので，写真の中央にある家屋が倒壊しているか否か分類する．本論文では，このような「短時間でできる，分類を行うタスク」を分類マイクロタスクと呼ぶ．

本論文では，分類マイクロタスクにおいて，大量のデータ中からある特定のクラスのデータのみを少ないタスク数で獲得したい場合に，タスクの割り当て順をどう制御すべきかという問題について議論する．分類マイクロタスクでは，ある特定のクラスのデータのみを少ないタスク数で獲得したい場合が存在する．そのような場合の例を次に示す．**例（航空写真からの被災家屋の特定）**— 図 1.1 に示した，家屋が倒壊しているか否か分類するタスクでは，被災地支援のために被災家屋を迅速に特定する必要があることから，被災家屋が写っている写真を迅速に獲得できることが望ましい．この場合，「被災家屋が写っている写真」が「ある特定のクラスのデータ」にあたる．なお，以下ではある特定のクラスに属するデータをそのクラスの「正例」とよぶ．

また，上記のような場合以外でも，一般に，大量のデータ中からある特定のクラスのデータのみを少ないタスク数で獲得できれば，必要なタスク数を削減できワーカに支払う報酬を削減できる．

上に述べたような，少ないタスク数で正例を獲得したい場合には，タスク中のデータが持つ特徴を利用してワーカに割り当てるタスクの順序を変更することが有効である．

具体的には，まず，タスク中のデータが持つ特徴からそのデータが正例である可能性を見積もり，その可能性が高い順にデータをランキングする．そしてランキング上位のタスクから先にワーカに割り当てればよい．

例えば，図 1.1 のタスクでは，写真の色の分布やエッジの数といった特徴を利用し，タスクの割り当て順を変更することができる．本論文ではこういった特徴を特徴量フィルタ（以下，単にフィル

図 1.1　分類マイクロタスクの例

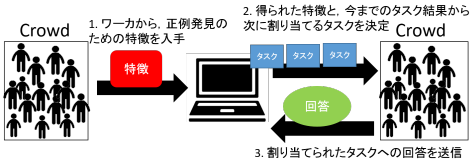

図 1.2　クラウドソースされた特徴に基づくタスク割り当て制御手法

タとよぶことがある）としてモデル化する．特徴量フィルタは，各特徴それぞれに対し作られる関数であり，データを引数とし，そのデータが持つ特徴量に応じて真偽を返す．

しかし，有効な特徴を発見することは必ずしも容易ではない．なぜならば，どの特徴が有効であるかはデータセットごとに異なり，そのデータセットの性質を事前に知ることは容易ではないからである．実際，図 1.1 の被災家屋判定タスクの場合，当初，色に関する特徴が有効であると仮定し，茶色や黄土色などに関するフィルタを作成したものの，それらの中に有効なものはほとんど存在しなかった．また，有効な特徴の中には，それを思いつくのに特定の分野の知識を必要とするものもあり，そうした特徴を発見することは容易ではない．

そこで本論文では，クラウドソーシングと機械処理によって大量の有効そうなフィルタを用意し，その中から有効なフィルタを選別しつつ，選別したフィルタを正例の発見に利用する手法を 2 つ述べる．この手法の全体像を図 1.2 に示す．クラウドソーシングによってフィルタを入手するのは，リクエスタだけでは考えつかない，正例の発見に有効そうなフィルタを数多く入手するためである．実際，クラウドソーシングを通して大量のフィルタを作成し，そのフィルタをデータの分類に利用するという研究がいくつか存在する [3][7]．これらの研究では，ワーカにフィルタを提案させることによって，ワーカのドメイン知識を活かしたフィルタが獲得できたと報告している．

本論文で述べる 2 つの手法はどちらも，ワーカから回答を受け取るたびに，その回答と今まで得られた全ての回答に基づき，次に割り当てるタスクを決定する．一方，これら 2 つの手法は，フィルタを利用して次のタスクを決定する部分のアルゴリズムが異なっている．この 2 つの手法はそれぞれ次のとおりである．

- 統計情報を利用したタスク割り当て手法：Statistics based Filter Select（以下では SFSelect と

よぶ）
この手法では Backward Stepwise Selection[6] と同様に，統計情報を用いて全てのフィルタの中から有効でないフィルタを徐々に取り除いていき，残ったフィルタに多くマッチするデータを選択する．

- 機械学習アルゴリズムを利用したタスク割り当て手法：ML
この手法では，既存の機械学習アルゴリズムを用いて各特徴を重みづけし，その重みを利用して各タスクのスコアを計算する．そして，そのスコアをもとにタスクのランキングを作成し，ランキング上位のタスクからワーカに割り当てる．

シミュレーションでは，2 つの手法について，一定数のタスクを処理した時点での適合率を比較し，それぞれの手法の性質を明らかにする．

本論文の貢献は次のとおりである．

(1) クラウドソーシングによって入手した特徴を用いた学習．本論文では，学習のための特徴をクラウドソースするという問題を扱う．これまでも，特徴をクラウドソースするという点に着目した研究は存在したが，そこでは，既存のデータの正例と負例を比較して特徴を見つける作業をクラウドソースするというアプローチであった．これは，既に学習のためのデータがあるという事を想定している．本論文では，このような仮定を置かず，クラウドの知識や発想を利用しようとするものである．

(2) クラウドソーシングによって入手した特徴の性質とマッチしたアルゴリズム．本論文では，玉石混淆であり，かつ，有効でないものがほとんどであるといった大量の特徴が与えられた時に，統計情報を利用したタスク割り当て方法が，特に初期の段階ではより良い学習を行う事を示した．この手法では，特徴の重みを 1 と 0 に離散化して「すぐに石を捨てる」ことを重視したアルゴリズムを用いている．

(3) 実データを用いた実験．本論文では，実際のニュース記事のデータと，クラウドソースした特徴を利用した実験を行い，特に初期の段階において，統計情報を利用したタスク割り当て手法が優れていることを確認した．

本論文の構成は以下のようになる．まず，第 2 章で関連研究を整理する．第 3 章では特徴のモデル化に用いる特徴量フィルタについて説明する．第 4 章では本論文で扱う問題を定式化する．第 5 章では統計情報を利用したタスク割り当て手法 SFSelect について，第 6 章では機械学習を用いたタスク割り当て手法 ML について述べる．第 7 章ではシミュレーションとその結果を述べる．第 8 章ではシミュレーションの結果について考察する．第 9 章では今後の課題を述べ，第 10 章で本論文をまとめる．

第 2 章

関連研究

本論文は，論文 [8] で提案したアルゴリズムを利用し，実データおよび実際にクラウドソーシングで得られた特徴を用いて行った大規模実験の結果を報告するものである．

多腕バンディット問題 本研究で扱う問題は多腕バンディット問題 [4] と類似している．多腕バンディット問題は知識の「活用」と「探索」がトレードオフの関係にある場合に，一定回数の試行後における利得をいかにして最大化するかという問題である．

本研究で扱う問題とこの問題は，探索と活用のトレードオフが存在するという点で類似している．既に述べたように，少ないタスク数で正例を獲得するには，有効なフィルタとそうでないフィルタを判別することが必要である．しかし，有効なフィルタを判別するために多くのタスクを処理しすぎると，有効なフィルタを活用する回数が減り，正例を発見するのに多くのタスクが必要になる．このような性質は，多腕バンディット問題における「活用」と「探索」のトレードオフと類似している．

一方，本研究で扱う問題とこの問題は次の点で異なる．すなわち，一般に多腕バンディット問題では，1 つの試行につき 1 つのアームの性能のみしか調査できないのに対し，本問題では，1 回の試行，すなわち 1 つのタスクが処理されるとき，複数のフィルタの良しあしを同時に調査できる．このため，多腕バンディット問題の解法を本研究で扱う問題に直接適用することはできない．

群衆による特徴の提案と特徴量の付与 群衆に特徴の提案と特徴量の付与を行わせた研究として Cheng らの研究 [3] や, Zou らの研究 [7] がある. これらの研究は, ワーカに分類の手がかりとなる特徴を提案させ, ワーカが提案した特徴とリクエスタ自身が考案した特徴を組み合わせた分類器を作成するものである. これらの研究と本研究は，どちらもワーカが提案した特徴を利用する点で類似している．

一方，次の点が異なる．すなわち，これらの研究では，分類器の精度向上のために，特徴の提案と特徴量の付与の両方をワーカに行わせているのに対し，本研究では特徴の発見のみをワーカに行わせ，特徴量の付与は機械的な方法を用いて行う．なぜならば，特徴量の付与まで群衆に行わせると，必要なタスク数が膨大になってしまうためである．

情報検索における検索結果のランキング作成 本研究で扱う問題は，情報検索における検索結果のランキング作成と類似している．ランキング作成ではランキングの上位に適合文書がくるよう文書を

並べ替える．これと同様に，本研究では，正例である可能性が高いタスクから順に処理されるようタスクの割り当て順を決定する．一方これらは，得られる結果の多様性を重視するか否かが異なっている．情報検索では一般に，検索結果のランキング上位に適合文書が多く出現することに加え，ランキング上位に多様な種類の文書が出現することが求められる．実際，Wozniak らの研究 [5] では複数のクエリをプーリングしておき，それらを多腕バンディット問題におけるアームとみなし交代で活用することにより，検索結果のランキング上位に様々な種類の適合文書が出現するようにしている．一方，本研究は，正例を獲得することに注目しており，正例の中での多様性を重視していない．ただし，多様性を重視するようアルゴリズムを拡張することも可能である．

特徴選択　統計情報を用いたタスク割り当て手法 SFSelect では，Backward Stepwise Selection と同様に，全てのフィルタの中から劣ったフィルタを徐々に取り除いていく．

一方，Backward Stepwise Selection とは次の点で異なっている．すなわち，一般に，Backward Selection では取り除いた特徴を活用することは無いが，SFSelect では一度取り除かれたフィルタを再び活用する場合が存在する．

第 3 章

特徴量フィルタ

本章では, データが持つ特徴のモデル化に使用する特徴量フィルタを定義する.

特徴量フィルタ f_i とは， データ d_jを引数にとり，d_jが性質 p_i を満たせば 1 を, そうでなければ 0 を返す関数である. 例えば，画像が茶色いか否かを判定するフィルタ $f_{brown}(d_j)$ は, d_j が茶色い画像データであれば 1 を, そうでなければ 0 を返す. 形式的には次のように定義される.

定義 1 (特徴量フィルタ)**.** $D = \{d_1, \ldots, d_n\}$ をデータ集合とし，p_i を判定したい性質とする．このとき，あるデータ $d_j \in D$ が性質 p_i を満たすか否かを判定する特徴量フィルタは，関数 $f_i \colon D \to \{1, 0\}$ である.

$$f_i(d_j) = \begin{cases} 1 & \text{if } d_j \text{ satisfies } p_i \\ 0 & \text{if } d_j \text{ dosen't satisfy } p_i \end{cases}$$

なお，以下では $f_i(d_j) = 1$ のとき，データ d_jはフィルタ f_iにマッチする，とよぶことがある．$f_i(d_j) = 0$ のときはその逆である.

次に空フィルタ f_{emp} を定義する．空フィルタ f_{emp} は全てのデータにマッチする特徴量フィルタであり，次のように定義される.

定義 2 (空フィルタ)**.** 空フィルタ $f_{emp} : D \to \{1, 0\}$ は次である.

$$(\forall d_i \in \mathrm{D})\ f_{emp}(d_i) = 1$$

空フィルタは，SFSelect と ML において，最悪の場合でもランダムな順でタスクを割り当てた場合とほぼ同程度の結果となることを保証するために用いられる.

次に，複数の特徴量フィルタを用いて，特徴量ベクトルを定義する.

定義 3 (特徴量ベクトル)**.** 性質の列 $p_1 \ldots p_m$ に関するデータ $d_i \in D$ の特徴量ベクトル $\boldsymbol{x_i}$ とは次である.

$$\boldsymbol{x_i} = (f_1(d_i), f_2(d_i), \cdots, f_m(d_i))$$

第 4 章

問題定義

本章では，本論文で扱う問題を定式化する．まず，分類したいデータの集合を $D = \{d_1, d_2 \ldots, d_n\}$ とし，D 中の各データがクラス c に属するか判定するタスクの集合を $T = \{t_1, t_2, \ldots, t_n\}$ とする．ここで，各 t_i は d_i がクラス c に所属するか否か分類するためのタスクである (図 1.1).

このとき，本論文における分類タスクの割当て順序制御とは，タスクの処理順序を表す列 T' を動的に作る事である．ここで，T' は T 中の全てのタスクをそれぞれ一度含む列である．例えば，$T = \{t_1, t_2, t_3\}$ の時，$T' = [t_2, t_1, t_3]$ などが考えられる．

T' を**動的に作る**とは，T' を先頭から順に作る際に，時点 k までに割り当てたタスク列 T'_k (最終的な T' の部分列) のタスク結果の集合 $Results_k$ を見て，次のタスク t_p を決定すると言う事である (図 1.2)．また，次に割り当てるタスクを決定する際に，クラウドソーシングと機械処理によって作成した特徴量フィルタを利用する．したがって，本論文で扱う手法は，次の手順になる．

1. 入力として，データ集合 D，タスク集合 T, および特徴量フィルタの集合 $F_{all} = \{f_1, \ldots, f_m\}$ をとる．
2. 時点 0 でのタスク列を $T'_0 = []$ とする．
3. タスクを行ってもらいながら，時点 $k+1$ のタスク列 T'_{k+1} を順次計算する．
4. $T'(= T'_n)$ と，T' に含まれる分類タスクを処理した結果の集合 $Results$（$= Results_n$）を出力する．

上記 (3) において T'_{k+1} を求める際の入出力は次の通りである．

> **入力** ある時点 k までに行われたタスク列 T'_k と，T'_k に含まれる分類タスクを処理した結果の集合 $Results_k$，および特徴量フィルタの集合 $F_{all} = \{f_1, \ldots, f_m\}$.
> **出力** $T'_{k+1} = T'_k + [t_p]$．ただし，t_p は，**特徴量フィルタを用いて**，$T - Set\,(T'_k)$ から選ばれる．Set は列を集合に変換する関数であるとする．

出力される T' の善し悪しは，いかに少ないタスク数で多くの正例 (クラス c に属するデータ) を発見できたかで評価される．つまり，一定のタスク数がこなされたときの適合率で評価される．

以降では，統計情報を利用したタスク割り当て手法 SFSelect と，既存の機械学習手法を利用したタスク割り当て手法 ML を説明する．これらは，上記 (3) において，t_p を選ぶために利用する特徴量フィルタをどう選別するかの部分が異なる．

第 5 章

統計情報を利用したタスク割り当て手法：SFSelect

統計情報を利用した手法 SFSelect では, 時点 k までのタスク処理結果に関する特徴量フィルタの統計情報をもとに，t_p を選択するために利用するフィルタの選別を行う．この手法は，第 6 章で述べる手法と比べると，以下の 2 つの特徴をもっている．（特徴 1）各特徴に 1 と 0 という離散的な重みづけを行う．（特徴 2) 多くの有効なフィルタにマッチするデータを持つタスクを優先する．

5.1 概要

統計情報を利用したタスク割り当て手法 SFSelect のアルゴリズムを Algorithm 1 に示す. SFSelect の入力はタスク集合 T とフィルタ集合 F_{all}であり, 出力はタスク処理結果の列 $Results$ である．

このアルゴリズムにおけるフィルタの選別手法は，Backward Stepwise Selection の一種である．すなわち次のようにしてフィルタを選別する．まず，F_{all} 中のフィルタを F_{active} というフィルタ集合に所属させる．そして，ワーカからの回答をもとに，他のフィルタと比べて相対的に有効でないと一時的に判断されたフィルタを順次 $F_{inactive}$ に移動させる．F_{active} はその時点において有効であるとされているフィルタの集合であり，$F_{inactive}$ はその時点において有効ではないとされているフィルタの集合である．また，マッチするデータを全て見つけたフィルタは F_{finish} というフィルタ集合に移動させる (図 5.1). F_{finish} の詳細は, 5.2 節で説明する.

本アルゴリズムは次のように動作する．まず，F_{all} 中のフィルタを全て F_{active} に所属させ, $F_{inactive}$, F_{finish} は空にする (1 行目, 2 行目) . つまり, 全てのフィルタが有効なものであるとみなす.

そして未処理のタスクのうち, F_{active} に属するフィルタに最も多くマッチするデータを持つタスクを選ぶ (5 行目) . 5 行目の getMaxMatchTasks 関数は, 未処理のタスク集合から, F_{active} に属するフィルタに最も多くマッチするタスクの集合を返す関数である. マッチするフィルタの数が多いデータを持つタスクを優先して割り当てるのは, マッチするフィルタが多いほど, 正例である確率が高い

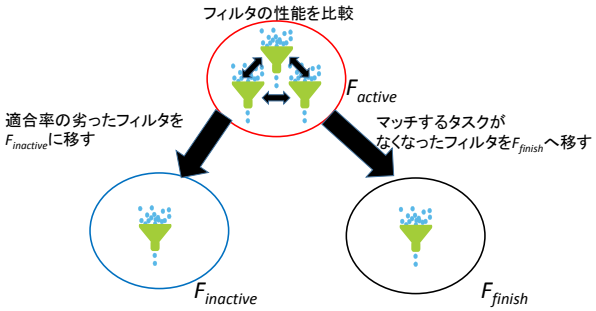

図 5.1　3 つのフィルタ集合とフィルタ選別

表 5.1　$RTable$ の例

フィルタ名	正例	負例
f_1	1	10
f_2	8	1
f_3	4	4
f_4	5	2

と期待するためである. getMaxMatchTask 関数から返された結果からワーカに割り当てるタスクをランダムに 1 つ選択する (12 行目) . そして選択されたタスクをワーカに割り当て, 回答を得る (13 行目) .

ワーカから回答が得られたら, その結果を $Results$ に加え (14 行目) , タスク処理結果保存テーブル $RTable$ の内容を更新する (15 行目). $RTable$ の例を表 5.1 に示す. $RTable$ を利用してフィルタの選別を行い, 再びタスクの処理を開始する.

5.2　フィルタの選別

本章ではフィルタの選別 (16 行目～22 行目) について説明する. フィルタの選別は F_{active} に属するフィルタを対象に, $RTable$ を用いて行う (16 行目) . そして選別の結果, 取り除くフィルタを F_{active} から $F_{inactive}$ に移す. この動作は図 5.1 の左側に示されている.

具体的には, F_{active} に属する 2 つのフィルタの組み合わせ全てについて, 適合率に有意差があるか検定を行う. 17 行目の existSignificantDiff 関数は, 与えられた 2 つのフィルタ f_1, f_2 について Precision(f_1) = Precision(f_2) という帰無仮説の検定を行い，2 つのフィルタの適合率に有意差が有るか否かを返す関数である. この関数は $RTable$ の値をもとに検定を行う. $RTable$ には各フィルタにマッチしたタスクに関して, 正例の個数と負例の個数が保存されており, これを用いてフィルタの適

合率を計算する. 検定の結果, 2 つのフィルタの適合率に有意差が存在するとされた場合, 適合率が劣っている方のフィルタを F_{active}から $F_{inactive}$ に移す (19 行目, 20 行目) . つまり, 適合率の有意に劣ったフィルタを, タスク処理順の変更に用いるフィルタの集合 (F_{active}) から取り除く. 19 行目の getInferiorF 関数は 2 つのフィルタを引数としてとり, 適合率が低い方のフィルタを返す関数である.

仮説検定の手法としては, フィッシャーの正確確率検定とカイ二乗検定を用いた．フィッシャーの正確確率検定は標本数が少ない場合にも正確な検定が行えるが, 標本数が大きくなると計算量が膨大になることが知られている. そのため本手法では, 標本数が少ない場合にはフィッシャーの正確確率検定を用い, 多い場合にはカイ二乗検定を用いた.

5.3　割り当てるタスクがない場合の処理

Algorithm 1 では, ワーカに割り当てるタスクが存在しない場合が生じる. その場合の処理 (6 行目から 11 行目) について説明する. まず, 割り当てるタスクが存在しないとは, F_{active} 中のフィルタにマッチするデータをもつ未処理のタスクが存在しないということである. この場合, F_{active}中のフィルタの適用を終了し, $F_{inactive}$ に属するフィルタを対象に選別を開始する. つまり, 残っているフィルタ集合を対象に, フィルタの選別を再び始める. 具体的には, まず, 図 5.1 の右側のように, F_{active} 中のフィルタを全て F_{finish} へと移す (8 行目) . F_{finish} は, 未処理のタスクのいずれにもマッチしないフィルタの集合である. そして, $F_{inactive}$ 中のフィルタを全て F_{active} に移し (9 行目) , 再びタスクの処理とフィルタの選別を開始する.

Algorithm 1 SFSelect

Input: T, F_{all}

Output: $Results$

1: $F_{active} \Leftarrow F_{all}$
2: $F_{inactive} \Leftarrow \emptyset$, $F_{finish} \Leftarrow \emptyset$
3: $Results \Leftarrow []$
4: **while** $|Results|$!= $|T|$ **do**
5: $candidateTasks \Leftarrow$ getMaxMatchTasks(T, F_{active})
6: /*f_{active} 中のフィルタにマッチするタスクがなくなった場合*/
7: **if** $candidateTasks$.empty? **then**
8: $F_{finish} \Leftarrow F_{finish} \cup F_{active}$
9: $F_{active} \Leftarrow F_{inactive}$ /*残っているフィルタの中で選別を開始する*/
10: next
11: **end if**
12: $task \Leftarrow$ getRandom($candidateTasks$)
13: $result \Leftarrow$ ask($task$) //タスクをワーカに割り当て, 回答を入手
14: $Results$.add($result$)
15: updateTable($RTable$, $Results$)
16: **for all** f_1, f_2 $s.t.$ $f_1 \in F_{active} \wedge f_2 \in F_{active} \wedge f_1 \neq f_2$ **do**
17: /*フィルタの性能をペアワイズで比較*/
18: **if** existSignificantDiff($RTable$, $f1$, $f2$) **then**
19: $inferiorF \Leftarrow$ getInferiorF(f_1, f_2)
20: F_{active}.delete($inferiorF$), $F_{inactive}$.add($inferiorF$)
21: **end if**
22: **end for**
23: **end while**

第 6 章

機械学習を用いたタスク割り当て手法：ML

本章では, 機械学習を用いたタスク割り当て手法 ML について説明する. この手法ではタスクの処理結果をもとに各フィルタに重みづけを行う. そしてフィルタの重みをもとに各タスクのスコアを計算し, スコアが最も高いタスクをワーカに割り当てる.

ML のアルゴリズムを Algorithm2 に示す. このアルゴリズムの入力は, 統計情報を利用した手法のアルゴリズムと同じく, タスク集合 T とフィルタ集合 F_{all} である. 出力はタスク処理結果の集合 $Results$ である.

このアルゴリズムでは, ワーカからの回答を訓練データと見なし, 訓練データをもとに各フィルタに対し重みづけを行う (8 行目). 訓練データは第 3 章で述べた特徴量ベクトルと同じ形式をとり, ワーカからの回答を正解ラベルとして持つ. そして得られた重みを用いて未処理のタスク全てのスコアを計算し, スコアが最も高いタスクをワーカに割り当てる (4 行目, 5 行目, 6 行目). スコアが高いほど, 正例である確率が高いとする. 4 行目の getMaxScoreTask 関数は未処理のタスク全てについてスコアを計算し, スコアが最も高いタスクの集合を返す関数である. なお, スコアが最も高いタスクが複数存在した場合, それらのタスクからワーカに割り当てるタスクをランダムに 1 つ選択する (5 行目).

ワーカから新たな回答が得られると (6 行目), そのデータを訓練データに加え (8 行目) 再学習を行い, 各フィルタの重みを更新する (9 行目). 以上のようなステップを全てのタスクを処理するまで繰り返す.

疑似コード中の $Classifier$ に対応する分類器としてはロジスティック回帰分析を用いた. 具体的には統計解析ソフト R の glm 関数を用いた.

ロジスティック回帰分析を用いると, 各タスクのスコアは次のように計算できる. すなわち, まず, glm 関数に引数として訓練データの集合をわたすと, 各フィルタの重みが返される. このとき, あるタ

Algorithm 2 ML

Input: T, F_{all}

Output: $Results$

1: $TrainingData \Leftarrow \emptyset$
2: $Results \Leftarrow \emptyset$
3: **while** $|Results|$!= $|T|$ **do**
4: $candidateTasks \Leftarrow Classifier$.getMaxScoreTask($tasks$)
5: $task \Leftarrow$ getRandom($candidateTasks$)
6: $result \Leftarrow$ ask($task$) //タスクをワーカに割り当て, 回答を入手
7: $Results$.add($result$)
8: $TrainingData$.add($result$)//得られた回答を訓練用データに追加
9: $Classifier$.learn($results, F_{all}$) //現在までのタスク処理結果を利用し再学習
10: **end while**

スク t において使用するデータを d とするとき, t のスコア $score(t)$ は以下のように計算される.

$$score(t) = w_1 f_1(d) + w_2 f_2(d) + \cdots + w_m f_m(d), \qquad w_i \in R, f_i(d) \in \{0, 1\}$$

ここで, w_iはフィルタ f_iの重みであり, 実数である. 例えば, フィルタ数が 3 でそれらの重みが $(w_1, w_2, w_3) = (2, 1, 1)$ であり, あるタスク t に対応するデータ d ついて $(f_1(d), f_2(d), f_3(d)) = (1, 0, 1)$ であるとき, t のスコアは

$$score(t) = w_1 f_1(d) \ + \ w_2 f_2(d) \ + \ w_3 d_3(d) = 2 * 1 + 1 * 0 + 1 * 1 = 3$$

となる

多腕バンディット問題の観点から見ると，このアルゴリズムは Greedy 法 [6] の一種としてとらえることができる．Greedy 法では，今までの試行の結果最も良いと予想されるアームを常に引き続け，探索は行わない．このアルゴリズムでもこれと同様に，今まで得られたワーカの回答からフィルタを重みづけしてタスクごとのスコアを計算し，スコアが最も高いタスク 1 つを選択する．この際，スコアが次点のタスクを選択することはしない.

第 7 章

シミュレーション

「ニュース記事を分類し，テレビに出演するような職業の人の不祥事に言及したニュース記事を少数獲得する」というタスクを想定し，シミュレーションを行った．このタスクの例を図 7.1 に示す．まず，使用したデータについて説明する．次に，クラウドソーシングを利用したフィルタの入手方法について述べる．そして SFSelect，ML それぞれについてシミュレーションの結果を述べる．なお，いずれの手法もランダムな要素の影響を受けるため，各手法について 10 回ずつシミュレーションを行い，その平均を求めた．

7.1　使用したデータ

2016 年の Yahoo!ニュース [1] の記事のうち，「ライフ」，「地域」，「経済」以外のカテゴリに属する記事 6,684 件を用いた．以上の 3 つのカテゴリに属する記事を除外したのは，それらのカテゴリに属するニュースが正例である可能性はかなり低いと考えたためである．

これらの記事全てに対し，上記のテーマに適合しているか否かという 2 値のラベルをクラウドソーシングによって付与した．なお，記事へのラベル付けでは，各記事についてワーカ 3 人にラベル付けを行わせた．3 人の回答が一致した記事についてはその結果をそのまま採用し，回答が割れた記事については，新たに雇用したアノテータ 3 名にラベル付けを行わせ，その 3 人の多数決の結果を採用した．

その結果，適合と判断された記事は 225 件（全体の約 3.8%），不適合と判断された記事は 6,459 件（全体の約 96.2%）であった．また，得られたデータの信頼性を確認するために，得られたデータについて Fleiss の Kappa 係数を求めたところ，その値は 0.667 となった．

7.2　クラウドソーシングによるフィルタの入手

Yahoo!クラウドソーシング [2] においてタスクを 200 件投稿し，「テレビ番組に出演するような職業の人の不祥事に関する記事」を検索するためのキーワードを入手した．タスクは自由入力式のも

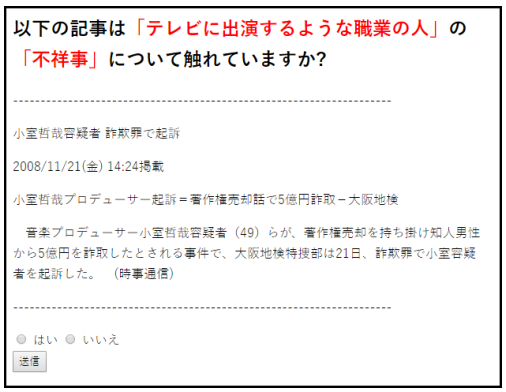

図 7.1　記事の適合性判定タスクの例

表 7.1　クラウドソーシングによって得られたキーワード（抜粋）

疑惑
恫喝
事務所
弁明
元人気子役

のとし，1 タスクにつき 1 個以上のキーワードを入力させた．この結果，キーワードは重複を除いて計 386 件得られた．得られたキーワードの一部を表 7.1 に示す．

次に，得られたキーワードそれぞれについてフィルタを作成した．具体的には，正規表現を利用し，各キーワードについて記事にそのキーワードが含まれていれば 1 を，そうでなければ 0 を返すフィルタを作成した．例えば，「恫喝」というキーワードについてのフィルタを $f_{\text{恫喝}}$ とするとき，記事 d に「恫喝」というキーワードが含まれていれば $f_{\text{恫喝}}(d) = 1$ となるようにした．各キーワードから作成したフィルタの適合率，再現率を表 7.2 に示す．なお，作成したフィルタのうち，選択率が 0% のものは削除した．なぜなら，選択率 0% のフィルタは正例の発見に役立たないためである．選択率が 0% のフィルタを削除した結果，フィルタ数は 286 となった．この 286 個のフィルタをシミュレーションで使用した．

7.3　入手したフィルタの性質

クラウドソーシングによって入手した 286 個のフィルタの性質を調べるため，各フィルタの適合率をもとにヒストグラムを作成した．それを図 7.2 に示す．

図 7.2 より，286 個のフィルタの大半は適合率が 10% 以下である一方で，適合率が 50% を超えているフィルタも少数ながら存在していることが分かる．

表 7.2　シミュレーションで用いるフィルタ（空フィルタと，F 値が高い順に上位 3 件）

	キーワード	選択率	適合率	再現率	F 値
f_{emp}	（空フィルタ）	100%	3.8%	100%	7.3%
$f_{疑惑}$	疑惑	1.4%	56.2%	21.4%	31.0%
$f_{逮捕}$	逮捕	2.3%	21.7%	13.5%	16.6%
$f_{不倫}$	不倫	0.5%	61.1%	8.7%	15.2%

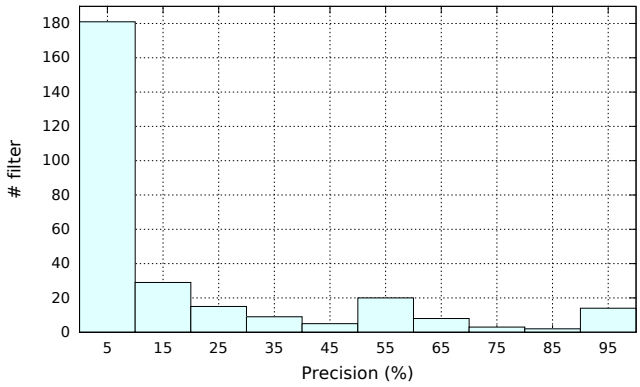

図 7.2　各フィルタの適合率によるヒストグラム

7.4　シミュレーションの結果

図 7.3，図 7.4 にシミュレーションの結果を示す．図 7.3，図 7.4 の横軸は累積タスク数であり，縦軸は再現率である．図 7.3 は全てのタスクの結果を示したものであり，図 7.4 は 500 タスク目までの結果のみを示したものである．また，各手法での Precision@n（n = 100，500，1000，5000）を表 7.3 に示す．n はその時点までに処理された累積タスク数であり，n が小さいときに Precision@n の値が高い程，少ないタスク数で多くの正例を獲得できていることを示す．

図 7.3，図 7.4 より，SFSelect と ML はともにランダムな順にタスクを割り当てる場合（Random）と比べて，少ないタスク数で多くの正例を発見していることがわかる．また，SFSelect と ML では，はじめから 2,000 タスク目付近までは SFSelect の方が高い再現率となっている一方で，約 3500 タスク目以降では ML の方が SFSelect を上回っていることがわかる．また，表 7.3，図 7.4 より，SFSelect と ML の差は，累積タスク数が少ない時点の方が多い時点と比べ，大きくなっていることが確認できる．

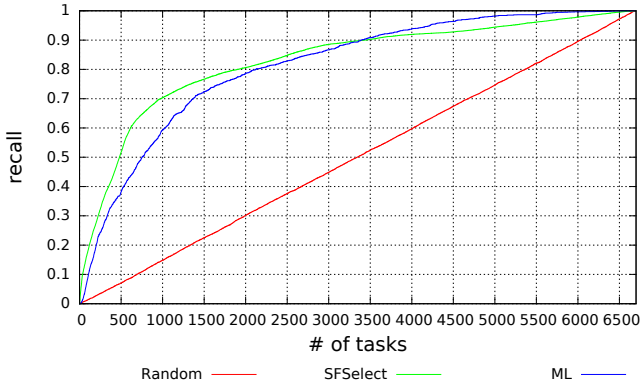

図 7.3　実施された累積タスク数とそれに伴う再現率の推移（全タスク）

図 7.4　実施された累積タスク数とそれに伴う再現率の推移（500 タスク目まで）

表 7.3　各手法での Precision@n

	Random	SFSelect	ML
Precision@50	3.6%	59.5%	12.8%
Precision@100	3.4%	45.0%	22.3%
Precision@500	3.6%	26.0%	19.3%
Precision@1,000	3.7%	17.7%	14.9%
Precision@5,000	3.8%	4.8%	4.9%

第 8 章

考察

シミュレーションの結果，タスク数が少ない段階では SFSelect の方が ML よりも有効であったが，この理由としては次の 2 つが考えられる．まず，SFSelect において，「多くのフィルタにマッチするデータを持つタスク」を優先して選択していることが理由の 1 つとして考えられる．多くの有効なフィルタにマッチするデータを持つタスクを優先することは，多くのフィルタに関する情報を早期に獲得することにつながる．例えば，1 つのフィルタにしかマッチしないデータからは 1 つのフィルタに関する情報しか得られないが，10 個のデータにマッチするデータからは 10 個のフィルタに関する情報が得られる．そのため，SFSelect では早期に多くのフィルタに関する情報が獲得でき，選別がはやく進み，少ないタスク数で多くの正例を獲得できたものと考えられる．

もう 1 つの理由としては，特徴に対する重みづけ方法の違いがあげられる．ML では各特徴に実数値の重みを付与する一方，SFSelect では 2 値の離散的な重みを付与する．すなわち，ML での重みづけと比べ，SFSelect ではより極端な重みづけを行っている．そのため，クラウドソーシングによって収集した，有効なものがごく少数のみ含まれる特徴を用いる場合，より極端な重みづけを行う SFSelect の方が少ないタスク数で特徴の選別が行え，多くの正例を発見できたのではないかと考えられる．

一方で，タスク数が増えた段階では ML の方が有効となったが，この原因は，ML において学習データが増え，各特徴に適切な重みが付けられるようになったためと予想される．ML では各特徴に実数値の重みを付与するため，離散的な重みを付与する SFSelect よりもより詳しく特徴を選別できる．そのため，学習データが増え，各特徴に適切な重みが付与できるようになった段階からは，ML の方が良い結果となったと予想される．

第 9 章

今後の課題

今後の課題としては，以下の 3 点がある．

（1） シミュレーション結果の要因分析．第 8 章では，タスク数が少ない段階では SFSelect が ML を上回ったことの理由と，タスク数が多くなった段階ではその逆になったことの理由について述べた．しかし現段階では，この理由の妥当性を示すのに十分なデータが得られていない．そのため，さらにデータを収集し，分析する必要がある．具体的には，ML における各フィルタの重みの推移や，SFSelect における 3 種類のフィルタ集合がどう変化しているかについて分析する必要がある．

(2) 他手法との比較．本論文では，ワーカに割り当てるタスクの順序を決定する 2 つの手法について述べた．しかし，ワーカに割り当てるタスクの順序を決定する方法は他にも多く考えられる．例えば，多腕バンディット問題の解法である UCB1 やトンプソンサンプリングを本論文で述べた問題用に拡張した手法を用いることも可能である．今後は，今回述べた 2 つの手法だけではなく，こういった他手法との比較も行う必要がある．

(3) 統計情報を用いた手法と機械学習アルゴリズムを用いた手法の組み合わせ．第 8 章で述べたように，処理されたタスク数が少ない段階では統計情報を用いた手法の方が有効であり，処理されたタスク数が多くなってくると機械学習アルゴリズムを用いた手法の方が有効になっている．このため，さらに良い結果を出すためには，タスク数が少ない段階では統計情報を用いた手法を用い，タスク数が増えてきた段階では機械学習アルゴリズムを用いた手法に切り替えることが有効であると考える．ただしその場合，どの程度の数のタスクが処理された時点で手法を切り替えるか決定する必要がある．

第 10 章

おわりに

本論文では，大量のデータ中から少ないタスク数で特定のクラスのデータを獲得したい場合に，いかにしてタスクの割り当て順を制御すべきか，という問題を扱い，クラウドソーシングによって入手した特徴を利用したタスク割り当て制御手法を 2 つ提案した．ニュース記事のデータを用いて実施したシミュレーションの結果，処理されたタスク数が少ない段階では統計情報を利用した手法の方が多くの正例を発見できることが示唆された．今後の課題としては，両手法の切り替えや他手法との比較がある．

参考文献

[1] http://news.yahoo.co.jp/.

[2] http://crowdsourcing.yahoo.co.jp/.

[3] Justin Cheng and Michael S. Bernstein. Flock: Hybrid crowd-machine learning classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW 2015, Vancouver, BC, Canada, March 14 - 18, 2015*, pp. 600–611, 2015.

[4] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, Vol. 6, No. 1, pp. 4–22, 1985.

[5] Cheng Li, Paul Resnick, and Qiaozhu Mei. Multiple queries as bandit arms. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, pp. 1089–1098, 2016.

[6] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning:An Introduction*. MIT Press, 1998.

[7] James Y. Zou, Kamalika Chaudhuri, and Adam Tauman Kalai. Crowdsourcing feature discovery via adaptively chosen comparisons. In *Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2015, November 8-11, 2015, San Diego, California.*, p. 198, 2015.

[8] 根本千代之介, 田島敬史, 森嶋厚行. 分類マイクロタスクにおけるタスク順序制御手法. DEIM2016 第 7 回データ工学と情報マネジメントに関するフォーラム, 2016.

本研究に関連する発表論文

国内研究会論文

- 根本千代之介，田島敬史，森嶋厚行，”分類マイクロタスクにおけるタスク順序制御手法”. 第 8 回データ工学と情報マネジメントに関するフォーラム (DEIM2016) pp.1-8，福岡，2016 年 3 月

謝辞

本研究を進めるにあたり，指導教員である森嶋厚行教授に深く感謝致します．大学学群から 3 年間，熱心に御指導して下さいました．ここまで研究を進めることができたのは森嶋先生のおかげです．ありがとうございました．

また，森嶋研究室に所属する方々にも大変お世話になりました．論文の添削から研究に関する指摘，日々の雑談まで，充実した研究生活が送れたのは皆さんのおかげです．学会参加時の書類作成時等にお世話になりました秘書の篠崎さんや，論文に関するコメントを頂きました池田さんにも感謝致します．

また，副研究指導員を引き受けて下さいました若林助教授に感謝致します．

最後になりましたが，合同ゼミなどでご指導いただきました杉本重雄教授，阪口哲夫准教授，永森光晴講師に，深く感謝いたします．