共変量シフト適応に基づくrandom forestsの並列分散学習

若山　涼至[†]　村田　隆英[†]　木村　昭悟[††a)]　山下　隆義[†]
山内　悠嗣[†]　藤吉　弘亘[†b)]

Training Random Forests on Large-scale Distributed Systems Based on Covariate Shift Adaptation

Ryoji WAKAYAMA[†], Ryuei MURATA[†], Akisato KIMURA[††a)],
Takayoshi YAMASHITA[†], Yuji YAMAUCHI[†], and Hironobu FUJIYOSHI[†b)]

あらまし　本論文では，MapReduceの枠組を用いて大規模データからrandom forestsを学習する新しい手法を提案する．Random forestsは，多数の決定木によって構成され，かつそれぞれの決定木を独立に学習することができるため，並列分散処理に非常に適した機械学習手法である．しかし，random forestsの学習をナイーブに並列分散化すると，それぞれの決定木を学習するために利用可能な学習データが少量となるため，しばしば過学習を引き起こす．本論文で提案する手法は，この過学習の問題を，以下の三つの要素を導入することで解決する．(1) 全てのワーカノードで共通にもつrandom forestsである共有RFを導入する．(2) 各ワーカノードのMap処理で共変量シフト適応に基づく転移学習を利用することにより，それぞれのワーカノードが保持する学習データに共有RFを適応させ，高い分類性能を獲得する．(3) 転移学習によって得られたrandom forestsをマスタノードに集約するreduce処理で，分類性能の向上に寄与しない決定木を削除することにより，分類性能を大幅に落とすことなく，分類時の計算コストを削減する．実験により，提案手法が分類性能を犠牲にすることなく高速な学習を実現できることを示す．

キーワード　Random forests，並列分散処理，MapReduce，転移学習

1. まえがき

統計的機械学習 [1]～[3] は，高い汎用性から自然言語処理，音声処理や画像処理など広い分野で用いられている．そのタスクは，教師あり学習である分類・回帰から，教師なし学習であるクラスタリングまで，多岐にわたる．統計的機械学習において，学習器の構築に利用するデータ量は重要であり，一般に，データ量を多くすることで学習器の性能は向上する．近年のインターネットやSNSの普及により学習データの収集が容易になり，大規模な学習データを容易に収集できるようになったため，学習に十分な時間をかければ，高い性能をもつ学習器が得られるようになった．しかし，学習データ量の増大は，学習のための処理時間を増大させる問題を引き起こす．しかし，単一の計算機による処理時間の大幅な短縮は困難であるため，複数の計算機やGPU等を用いる並列分散処理が一般的に用いられている．

並列分散処理では，処理するプロセッサを増やし，全体の処理を適切に分割して各プロセッサに割り当てることで，処理時間を大きく削減することが可能である．効率的な並列分散処理を行う標準的な枠組として，MapReduce [4] が広く知られている．MapReduceは，全ての計算機を統括する計算機 (マスタノード) と，実際に処理を行う計算機 (ワーカノード) から構成される．Map処理では，全てのデータをマスタノードへ入力し，マスタノードがワーカノードへデータを割り振る．ワーカノードは割り当てられたデータに処理を行い，処理結果をマスタノードへ返還する．Map処理後，返還された処理結果を統合し，解決すべき問題の答えを出力する．この処理がReduce処理である．これら

[†] 中部大学工学部情報工学科，春日井市
Department of Information Engineering, Chubu University, 1200 Matsumoto-cho, Kasugai-shi, 487–8501 Japan
[††] 日本電信電話株式会社コミュニケーション科学基礎研究所，厚木市
Communication Science Laboratories, NTT Corporation, 3–1 Morinosato Wakamiya, Atsugi-shi, 243–0198 Japan
a) E-mail: akisato@ieee.org
b) E-mail: hf@cs.chubu.ac.jp
DOI:10.14923/transinfj.2015IUP0008

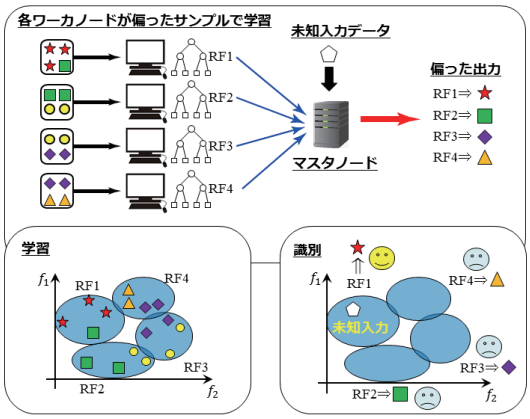

図 1 MapReduce による random forest 学習の問題点．各ワーカノードに割り当てられた学習サンプルのみで決定木を構築するため，サンプルの偏りが大きいと過学習で分類性能が低下する．

Fig. 1 Naive application of random forests to the MapReduce framework degrades the classification performance.

の処理により膨大な量のデータに対し比較的少ない時間で処理を行うことが可能となる．MapReduce では，Map 処理と Reduce 処理の各ステップで並列可能であるため，理論上全ての処理を並列に行うことが可能である．また，MapReduce では，一般の汎用サーバで取り扱うことのできない大量のデータを処理することが可能である．MapReduce の枠組を用いた統計的機械学習手法の並列化も数多くの検討がなされており [5]，ナイーブベイズ・パーセプトロン・ロジスティック回帰などの分類器，k-means・latent Dirichlet allocation (LDA) などのクラスタリング，主成分分析・協調フィルタリングなどの行列分解など，基本的な手法の多くが有力な分散並列ライブラリに実装されている(注1)．

本論文では，MapReduce のフレームワークに適した統計的機械学習手法の一つである，random forests の新しい並列分散学習法を提案する(注2)．

Random forests は，Breiman らによって提案された，複数の決定木で構成されるマルチクラス分類器を構築する，アンサンブル学習アルゴリズムである [8]．Random forests では，決定木の学習のためのサンプル選択にブートストラップサンプリングを取り入れることで過学習を抑制するとともに，決定木の各中間ノードにおける分岐関数の構築にランダム特徴選択 [9] を導入することで，特徴ベクトルの次元数にかかわらず高速に学習することが可能である．また，random forests は独立した複数の決定木で構成されるため，その学習は MapReduce を含む並列分散処理との整合性が非常に高い．この特性から，MapReduce の枠組による random forests 及びその改良手法の並列分散高速化が各種提案され [10]～[12]，有力な分散並列ライブラリにも実装されている．

しかし，学習時間を削減することを重視すると，各ワーカノードに割り当てられる学習データ（分割学習データ）の量が少なくなり，分割学習データの分布に大きな偏りが発生する可能性がある．分布に偏りがある学習データで random forests の学習を行うと，過学習により分類性能が低下することが知られている [12]．Rio ら [12] らは，マスタノードに全ての学習データが入力される場合を想定し，サンプル数が少ないクラスのサンプルを複製するとともに，各ワーカに割り当てるサンプルを適切に設定することで，過学習を回避する手法を提案しているが，分散ファイルシステム上での並列分散処理など，学習データが既に各ワーカノードに分散配置されている状況では適用できない．

これに対し，本論文では，Map 処理・Reduce 処理のそれぞれに独自の改良を行うことで，この過学習の問題を回避しつつ，並列分散処理による高速な学習を実現する手法を提案する．まず，Map 処理を実行する前に，学習データとは別にあらかじめ用意した共有データを用いて random forests (共有 RF) を構築し，この共有 RF を全てのワーカノードであらかじめ共有する．共有データは，各分割学習データと同様の分布をもつとは限らないため，転移学習を利用することにより，共有 RF を各分割学習データに適応させる．これら共有 RF と転移学習の導入により，過学習の問題を回避する．更に，Reduce 処理において，分類精度の向上に寄与しない不要な決定木を削除することにより，分類性能を落とすことなく，分類時の処理コストを削減する．

本論文の以降の構成は以下のとおりである．まず **2.** では，本論文で用いる random forests の転移学習手法である transfer forest [13] について述べる．続いて **3.** で，MapReduce の枠組による random forests の分散並列学習の提案手法を説明する．分散並列学習の利用シーンとして，二つの典型的なモデルを検討し，

(注1)：Apache Mahout: List of algorithms, http://mahout.apache.org/users/basics/algorithms.html

(注2)：本論文では，国内研究会 [6] 及び [7] にて報告した内容に基づき，提案手法の具体的な処理をより詳細に記述するとともに，幾つかの追加実験により提案手法の有効性を検証した結果を報告する．

それぞれについての分散並列学習手法を提案する．**4.** にて，評価実験を行い，提案手法の有効性を示し，**5.** にて本論文をまとめるとともに，今後の課題について議論する．

2. Random forests の転移学習

本章では，提案手法で用いる random forests の転移学習手法である transfer forest [13] について述べる．転移学習とは，事前に用意されたサンプルや学習結果を利用して，異なる問題に対する学習を効率化する手法であり，数多くの手法が提案されている [14]～[16]．Transfer forest では，転移学習の一形態である共変量シフト適応 [17], [18] を random forests に導入した手法である．共変量シフト [19] とは，与えられた入力に対する出力の生成規則が学習時とテスト時で不変であるものの，入力の分布が学習時とテスト時で異なる，という状況を指す．本説で説明する transfer forest は，この共変量シフト下で適切な random forests の学習を実現する手法である．

Transfer forest では，従来の転移学習と同様に，既に得られているサンプルや分類器を事前ドメインとし，それらを利用して目標ドメインの学習を行う．このとき，事前ドメインのサンプル（事前サンプル）は大量に存在し，目標サドメインのサンプル（目標サンプル）は事前サンプルと比べて少数であると仮定することが一般的である．また，事前サンプルを用いて事前学習した random forests (事前 RF) は構築済みであるとする．Transfer forest では，事前サンプル・事前 RF・目標サンプルを用いて，目標ドメインに適した分類器 (目標 RF) を学習する．このとき，事前サンプルは共変量シフト適応により目標ドメインの学習への有効性で重み付けされ，重みの低い事前サンプルは目標ドメインの学習への影響が小さくなる．

図 2 に transfer forest の学習の概要を示す．Transfer forest の学習は以下の四つのステップで構成される．

Step 1: サブセットの生成　事前サンプル集合 $\mathcal{S}^{(S)}$ と目標サンプル集合 $\mathcal{S}^{(T)}$ から同数のサンプルを重複を許して選択し，目標 RF を構築するための学習用サンプルのサブセット $\mathcal{S}^{(L)}$ を生成する．サブセット $\mathcal{S}^{(L)}$ 内の各サンプル s_k の重要度 λ_k の初期値を，事前サンプル集合から選択されたサンプルについては 0，目標サンプル集合から選択されたサンプルについては 1 とする．

Step 2: 決定木の構築　以降に示す三つのサブステップにより，Step 1 で生成されたサブセット $\mathcal{S}^{(L)}$ から，1 本の決定木 t を構築する．

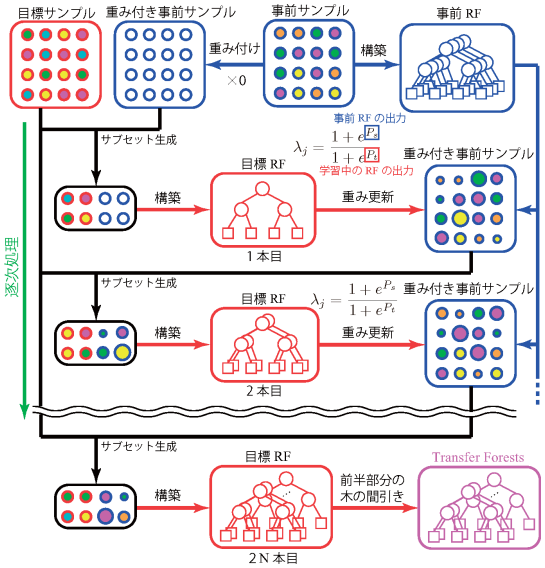

図 2　Transfer forest
Fig. 2　Transfer forest.

Step 2.1: 決定木の初期状態　決定木 t の初期状態では，一つのノードのみをもち，学習用サンプルサブセット $\mathcal{S}^{(L)}$ の全てのサンプルをこのノードに割り当てる．

Step 2.2: 決定木構造の構成　決定木 t の各葉ノードについて，左右二つの子ノードを生成し，以下の式 (1) に従って，ノードに割り当てられた学習用サンプル s_k を左右の子ノードのいずかに割り当てる．

$$d(s_k;\theta_j) = \mathrm{sgn}(f(s_k;\theta_j)), \tag{1}$$

ここで，j はノードを同定するためのインデックス，$\mathrm{sgn}(\cdot)$ は括弧内が正のときに 1，負のときに -1 を出力する符号化関数，$f(s;\theta)$ はパラメータ θ をもつ分岐関数であり，割り当てられた学習用サンプルをより良く分類するように，パラメータ θ を適切な方法で決定する．$d(s_k;\theta_j)$ が 1 ならばサンプル s_k を左の子ノードへ，-1 なら右の子ノードへ割り当てる．

Step 2.3: 決定木の末端ノードの更新　現在注目する葉ノード j が所定の深さに到達する，若しくは，その葉ノードに到達したサンプルの正解クラスラベルが全て同じとなったとき，この葉ノード（以下，末端ノードと呼ぶ）j に到達した学習サンプル集合 $\mathcal{S}_j^{(L)}$ の各サンプル s_k，その正解クラスラベル，及び重要度 λ_k を用

いて，末端ノード j がもつ各クラス $c \in \{1, 2, \ldots, C\}$ の確率分布 $p(c|j;t)$ を以下の式で計算する．

$$p(c|l;t) \propto \sum_{k, s_k \in \mathcal{S}_l^{(L)}} w(c_k) \lambda_k \delta_{c_k, c}, \tag{2}$$

ここで，$\delta_{i,j}$ はクロネッカーのデルタである．また，$w(c)$ はクラスラベルごとのサンプル数の偏りを緩和するために与えられる係数であり，サンプル数の少ないクラスラベルには大きい値が，逆にサンプル数の多いクラスラベルには小さい値が与えられる．

Step 3: 重要度の更新　Step 2 で構築した決定木 t を目標 RF $\mathcal{T}^{(T)}$ に加える．事前 RF $\mathcal{T}^{(S)}$ と更新された目標 RF $\mathcal{T}^{(T)}$ を用いて，事前サンプル集合 $\mathcal{S}^{(S)}$ に含まれる各サンプル s_k の重要度 λ_k を以下の式 (3) により更新する．

$$\lambda_k = \frac{1 + \exp\left\{\sum_{t \in \mathcal{T}^{(S)}} p(c_k|j(s_k,t);t)\right\}}{1 + \exp\left\{\sum_{t \in \mathcal{T}^{(T)}} p(c_k|j(s_k,t);t)\right\}}, \tag{3}$$

ここで，$j(s,t)$ は決定木 t でサンプル s が到達した末端ノードである．目標サンプル集合の各サンプルについての重要度はいずれも 1 とする．

Step 4: 決定木群の構築　以上の Step 1 から Step 3 を，あらかじめ指定した目標 RF の本数の 2 倍になるまで繰り返す．Step 2 及び 3 で示したように，転移学習における事前サンプルの重要度 λ_k は，決定木が追加されるたびに目標 RF と事前 RF との違いを正しく表現できるように更新される．そのため，学習の序盤での重要度 λ_k の信頼性は低く，目標サンプルの学習に適合していない可能性がある．そこで本論文では，信頼性の高い学習後半部分の決定木だけを採用することで，学習初期の悪影響を低減する．

3. 提案手法

本章では，MapReduce の枠組で random forests を学習する提案手法について説明する．以降では，分散並列学習の利用シーンとして，分散ファイルシステムとマルチコアサーバの 2 通りのモデルを想定し，それぞれのモデルに対応した random forests 学習の Map 処理を提案する．

3.1 分散ファイルシステムモデル

本節で説明する分散ファイルシステムモデルでは，各ワーカノード $i \in \{1, 2, \ldots, N\}$ は分散ファイルシステムを構成するノードの一つでもあり，random forests を学習する際に用いる学習データ $\mathcal{S}^{(T)}$ が既に各ワーカノードに分散配置されていることを仮定する．このモデルは MapReduce の枠組における典型的な利用シーンの一つであり，各ワーカノード i は手元にある学習データである分割学習データ $\mathcal{S}_i^{(T)}$ のみを利用して random forests を学習する．特に，学習データ全体が非常に大規模となる場合には，データを転送するコストを最小限に抑えることができる本モデルは望ましい形態の一つである．しかし，学習データは既に各ワーカノードに分散配置されているため，一般には，各ワーカノードに配置された分割学習データの分布に偏りが生じている可能性を否定できない．Random forests を学習する場合，学習データの分布に偏りが発生すると，過学習により分類性能が低下することが知られている [12]．

そこで本モデルでは，ワーカノードがもつ分割データとは別に，全てのワーカノードで共有するデータである共有データを導入する．一般に，共有データは各分割データと同じような分布をもつとは限らないため，各ワーカノードでは，共有データを事前ドメイン，分割データを目標ドメインとする transfer forest を用いることで，過学習を抑制しつつ分割データに適合した random forests を学習することが可能となる．Random forests の転移学習手法として，本論文では transfer forest を用いるが，以降に示すアルゴリズムからも推察されるとおり，別の手法を代用することも可能である．Transfer forest では，事前ドメインに相当する共有データから共有 RF と呼ばれる random forests をあらかじめ学習しておく必要があるが，この共有 RF の構築は共有データに対して一度だけ実行すれば十分であり，類似するドメインの分類問題に対して繰り返し利用することが可能である．各ワーカノードで構築した目標 RF は，全てマスタノードに送られる．

図 3 に分散ファイルシステムモデルでの Map 処理の概要を，Algorithm 1 にそのアルゴリズムを示す．

3.2 マルチコアサーバモデル

もう一つの典型的な利用シーンであるマルチコアサーバモデルでは，マスタノードに一度全ての学習データが入力され，それを分割してワーカノードに配置する．このモデルでは，分散ファイルシステムモデルと異なり，マスタノードが配置先のワーカノードを指定できることから，分割学習データのサンプルの分布に偏りが発生することは少ない．しかし，サンプルの分布が偏らない場合においても，ワーカノード数

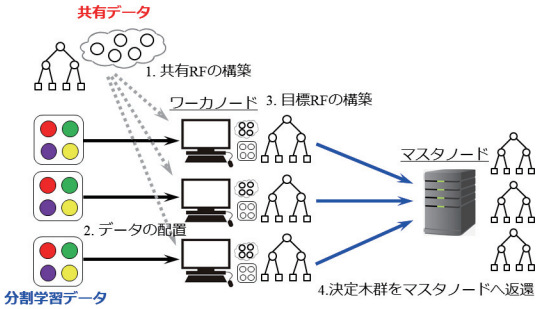

図 3　分散ファイルシステムモデルでの Map 処理
Fig. 3　Map stage for distributed file systems.

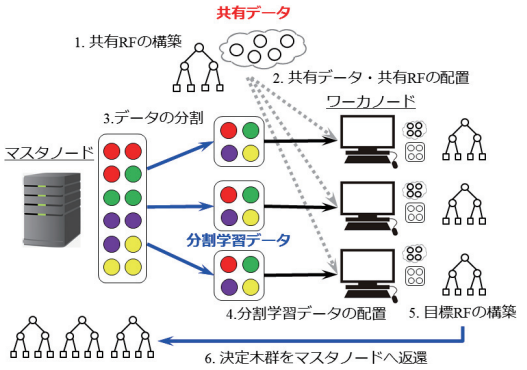

図 4　マルチコアサーバモデルでの Map 処理
Fig. 4　Map stage for multi-core servers.

Algorithm 1 分散ファイルシステムモデル Map 処理

Input: 学習データ $\mathcal{S}^{(T)} = \{\mathcal{S}_1^{(T)}, \mathcal{S}_2^{(T)}, \ldots, \mathcal{S}_N^{(T)}\}$，共有データ $\mathcal{S}^{(S)}$

– **事前処理** –

（1）共有データ $\mathcal{S}^{(S)}$ から事前 RF $\mathcal{T}^{(S)}$ を構築．

（2）各ワーカノード i に，共有データ $\mathcal{S}^{(S)}$，共有 RF $\mathcal{T}^{(S)}$，分割学習データ $\mathcal{S}_i^{(T)}$ を配置．

– **Map 処理** –

（3）各ワーカノードで，分割学習データ $\mathcal{S}_i^{(T)}$，共有データ $\mathcal{S}^{(S)}$ 及び事前 RF $\mathcal{T}^{(S)}$ を用いて，transfer forest により決定木群 $\mathcal{T}_i^{(T)}$ を並列に構築．

（4）各ワーカノードで構築した決定木群 $\mathcal{T}_i^{(T)}$ をマスタノードへ返還．

Output: 各ワーカノードで構築した決定木群 $\mathcal{T}^{(T)} = \{\mathcal{T}_1^{(T)}, \mathcal{T}_2^{(T)}, \ldots, \mathcal{T}_N^{(T)}\}$

Algorithm 2 マルチコアサーバモデル Map 処理

Input: 学習データ $\mathcal{S}^{(T)}$，共有データ $\mathcal{S}^{(S)}$

– **事前処理** –

（1）共有データ $\mathcal{S}^{(S)}$ から事前 RF $\mathcal{T}^{(S)}$ を構築．

（2）各ワーカノード i に共有データ $\mathcal{S}^{(S)}$，共有 RF $\mathcal{T}^{(S)}$ を配置．

データ $\mathcal{D}_w$ をマスタノードに入力．

– **Map 処理** –

（3）マスタノードが学習データ $\mathcal{S}^{(S)}$ をノード数分の分割学習データ $\{\mathcal{S}_1^{(T)}, \mathcal{S}_2^{(T)}, \ldots, \mathcal{S}_N^{(T)}\}$ に分割．

（4）各ワーカノード i に分割学習データ $\mathcal{S}_i^{(T)}$ を配置．

（5）各ワーカノードで，分割学習データ $\mathcal{S}_i^{(T)}$，共有データ $\mathcal{S}^{(S)}$ 及び事前 RF $\mathcal{T}^{(S)}$ を用いて，transfer forest により決定木群 $\mathcal{T}_i^{(T)}$ を並列に構築．

（6）各ワーカノードで構築した決定木群 $\mathcal{T}_i^{(T)}$ をマスタノードへ返還．

Output: 各ワーカノードで構築した決定木群 $\mathcal{T}^{(T)} = \{\mathcal{T}_1^{(T)}, \mathcal{T}_2^{(T)}, \ldots, \mathcal{T}_N^{(T)}\}$

が増えることにより一つのワーカノードに対して割り当てられる学習データの量が少なくなる．Random forests の学習では，サンプル数が少なくなると分類性能が低下することが知られている．そこで，マルチコアサーバモデルにおいても，分散ファイルシステムモデルと同様に共有データを利用することで，全てのワーカノードに必要最低限のデータ量を確保するとともに，transfer forest により目標 RF を構築することで，各分割学習データに適応した random forests を構築できる．図 4 にマルチコアサーバモデルの Map 処理の概要を，Algorithm 2 にそのアルゴリズムを示す．

3.3　各モデルの特長

分散ファイルシステムモデルは，マスタノードによる配置データの調整が不可能なため過学習が起こりやすい．しかし，データの分割・配置の処理を行わないため，データ転送に要するコストが小さい．すなわち，分散ファイルシステムモデルによる random forests の学習は，高効率な並列分散学習であるといえる．逆に，マルチコアサーバモデルは，マスタノードによる分散学習データの配置を調整できるため，過学習が起こりにくく，一般に分散ファイルシステムモデルよりも高い分類性能を得られる．しかし，マスタノードがデータを分割・配置する必要がある．すなわち，マルチコアサーバモデルは，高精度な並列分散処理による学習であるといえる．分類精度と学習効率はトレードオフの関係にあり，解くべき問題や使用できる計算機などによりモデルを変えていくことが重要である．

3.4　Reduce 処理

提案手法の Map 処理で並列に構築した決定木は，分割学習データの配置によっては，分類性能の向上に寄与しない冗長なもの，あるいは分類性能を劣化させる不要なものとなる可能性がある．分類時の計算コストは決定木の数に依存するため，このような冗長若しくは不要な決定木はできる限り除去しておくことが望ましい．この観点から，本提案手法での Reduce 処理では，Map 処理で並列に構築された目標 RF を集約す

Algorithm 3 Reduce 処理

Input: 各ワーカノードで構築され返還された決定木群 $\mathcal{T}^{(T)} = \{\mathcal{T}_1^{(T)}, \mathcal{T}_2^{(T)}, \ldots, \mathcal{T}_N^{(T)}\}$

– Reduce 処理 –

（1）データ $\mathcal{S}^{(I)}$ を用いて，下式により $\mathcal{T}$ 内の各決定木 t のスコアを算出．

$$v(t) = -\sum_{s_k \in \mathcal{S}^{(I)}} \max_{c \neq c_k} p(c|s_k; t).$$

（2）スコアの低い木から順に，指定した本数の木を削除．

Output: 削除されずに残った決定木の集合

るとともに，不要と思われる決定木を削除することで，一つの決定木群を生成する．特に本論文では，事後確率に基づく決定木の削除方法を提案する．Random forests を用いた分類の際に，各サンプルは，random forests を構成する各決定木が出力した事後確率の総和が最大となるクラスに分類される．すなわち，正解クラスラベルの事後確率が高い決定木を数多く残し，不正解クラスラベルの事後確率が高い決定木を削除することで，分類性能を向上させることができる．この考え方に基づき，本論文では，以下のスコア $v(t)$ による決定木 t の削除を考える．

$$v(t) = -\sum_{s_k \in \mathcal{S}^{(I)}} \max_{c \neq c_k} p(c|s_k; t). \tag{4}$$

ここで，$\mathcal{S}^{(I)}$ はスコア算出用のサンプル集合，$p(c|s;t)$ はサンプル s を決定木 t に与えたときのクラス c の事後確率を表す．Reduce 処理は分散ファイルシステムモデル，マルチコアサーバモデル，共に同じ処理であるが，スコア算出に利用するサンプルはに合わせて変更する必要がある．例えば，分散ファイルシステムモデルにおいては，マスタノードはワーカノードが利用したデータを使用できないため，スコア算出にはマスタノードが独自に保持するサンプルを利用する必要がある．一方，マルチコアサーバモデルでは，ワーカノードが利用するデータは一度マスタノードを通過するため，Out-of-Bag (OOB) サンプルなどを利用してスコアを算出することができる．Algorithm 3 に，スコアを用いた決定木削除による Reduce 処理のアルゴリズムを示す．

4. 評価実験

提案手法の有効性を評価するために評価実験を行う．本実験では，分類精度と学習に要する計算量により Map 処理をを評価するとともに，分類精度により Reduce 処理を評価する．実験に用いた計算機のスペックは，以下のとおりである．Intel(R) Xeon(R) CPU E5-2637 v2 (3.50GHz) ×2, 32GB RAM, Windows Server(R) 7 Professional, Microsoft Visual C++ 2012.

4.1 データセット

本実験では，UCI Machine Learning Repository で公開されているデータセットである letter recognition [20] と MNIST を用いて評価を行う．Letter recognition は，サンプル数 20000，クラス数 26，特徴次元 16 のデータセットであり，MNIST は，サンプル数 70000，クラス数 10，28×28 の画像からなるデータセットである．本実験では，letter recognition のサンプルのうち 6666 個，MNIST のサンプルのうち 10000 個をそれぞれ評価データとし，残りを学習の際に用いるデータとする．この学習に用いるデータのうち，特に指定のある場合を除き，letter recognition では 4000 個，MNIST では 10000 個を共有データ，それぞれの残りを分割学習データとして用いる．決定木のパラメータは予備実験により探索し，letter recognition は決定木の総数 50，深さ 15，特徴次元選択回数 15 回，しきい値選択回数 50 回，MNIST では，決定木の総数 30，深さ 15，特徴次元選択回数 50 回，しきい値選択回数 5 回とする．また，以降の評価で示す学習に要した計算時間（学習時間）には，共有 RF を構築するために要した計算時間は含まれていない．これは，一度構築した共有 RF が類似する分類タスクに再利用可能であり，対象とする具体的なタスクごとに共有 RF を再構築する必要がないためである．

4.2 実験 1: Map 処理の評価

本実験では，分散ファイルシステムモデル (DS モデル) とマルチコアサーバモデル (MC モデル) における学習の評価を行う．各モデルでの利用シーンを再現するために，各ワーカノードに割り当てる分割学習データを各モデルで調整する．具体的には，分散ファイルシステムモデルでは，データ分布に偏りがあるものとする．本実験では，データ分布の偏りはクラスラベルにより配置先のワーカノードを決定することで発生させる．一方，マルチコアサーバモデルでは，データ分布に偏りがないものととして，分割学習データをランダムにワーカノードに割り当てる．提案手法 (DF) の比較対象として，共有データを用いない random forests による学習 (RF) を採用する．

Letter recognition データを用いた際の分類性能を

表 1　分類誤差 (letter recognition) [%]
Table 1　Classification error (letter recognition) [%].

	DS モデル		MC モデル	
ワーカ数	RF	DF	RF	DF
1	6.18	5.14	6.19	5.14
2	9.82	5.71	7.78	5.83
5	26.4	8.28	10.6	7.29
10	67.7	9.53	18.3	8.59

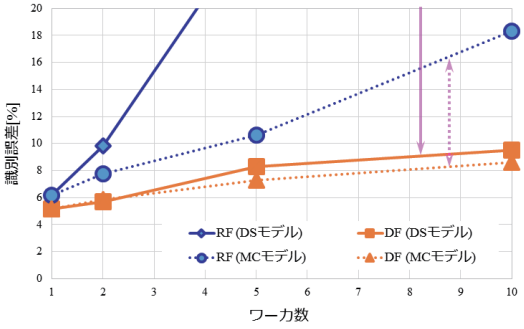

図 5　分類誤差 (letter recognition) [%]
Fig. 5　Classification error (letter recognition) [%].

表 2　分類誤差 (MNIST) [%]
Table 2　Classification error (MNIST) [%].

	DS モデル		MC モデル	
ワーカ数	RF	DF	RF	DF
1	5.31	5.31	5.19	5.81
2	11.2	6.59	5.81	6.33
5	32.1	7.03	6.99	6.67
10	88.4	7.48	7.85	7.23

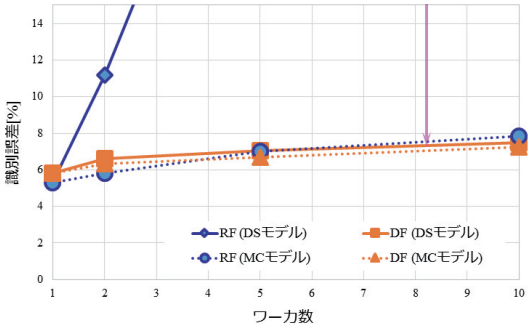

図 6　分類誤差 (MNIST)
Fig. 6　Classification error (MNIST).

表 1 及び図 5 に，MNIST データを用いた際の分類性能を表 2 及び図 6 に，それぞれ示す．実験結果から，分散ファイルシステムモデルについては，特に共有データを利用しない従来手法で大きく分類性能が低下している一方，転移学習を利用する提案手法では，データに分布の偏りが発生した場合においても精度の低下が抑えられている．また，マルチコアサーバモデルについては，データの分割時に分布の偏りが発生しないものの，分割数が多い場合には一つのワーカノードに配置されるサンプル数が少なくなるため，決定木を十分に学習することができず，分類性能が低下する．このような問題に対しても転移学習を利用する提案手法は有効である．

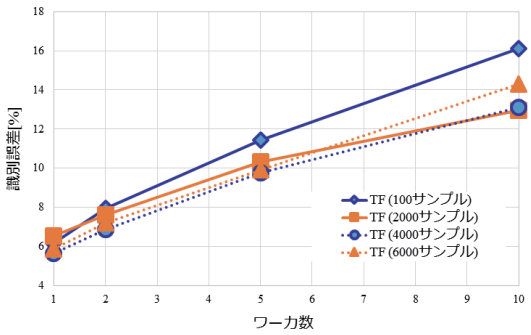

図 7　共有データのサンプル数を変化させた際の分類誤差 [%]
Fig. 7　Classification error for the changed of number of shared samples [%].

続いて，共有データのサンプル数を 100, 2000, 4000, 6000 と変化させた際の分類誤差を計測することにより，共有データが分類性能に与える影響を調査する．Letter recognition データを用いた際の分類性能を図 7 に示す．

提案手法は，転移学習に基づく手法であることから，転移学習で想定している問題設定，すなわち転移元ドメインである共有データのサンプル数が十分に存在し，転移先ドメインである分割学習データのサンプル数が必ずしも十分ではない，という仮定が成立する状況が望ましい．すなわち，共有データのサンプル数を極端に削減すると分類精度の向上は望めなくなる．図 7 の実験結果は，この考察を支持するものである．

4.3　実験 2: Reduce 処理の評価

本実験では，提案する reduce 処理による決定木の選択的削除の効果を，分類誤差により定量評価する．具体的には，各決定木の分類性能の差を顕著にするために，学習データに一定の割合で恣意的に誤ったクラスラベルを与え，誤ったクラスラベルをもつサンプルの割合を変えたときの分類誤差を評価する．このとき，スコア算出には評価用サンプルの半分を利用し，残りを分類誤差の評価に用いる．また，評価用サンプルにはクラスラベルにノイズがないものとする．ノイズの

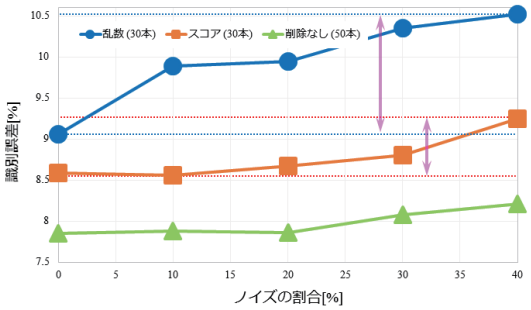

図 8　ノイズの割合による分類誤差 (letter recodnition)
Fig. 8　Classification error for mislabeled training data (letter recodnition).

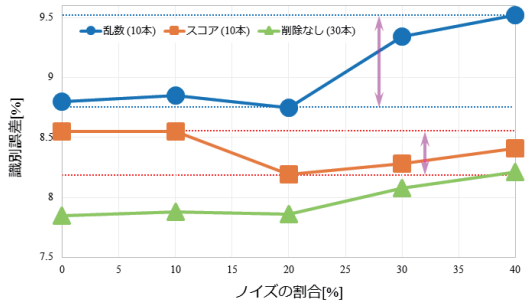

図 9　ノイズの割合による分類誤差 (MNIST)
Fig. 9　Classification error for mislabeled training data (MNIST).

割合を変化させる実験では，ワーカ数を 10，木の総削除本数を 20 とする．提案手法の比較対象として，乱数による決定木の削除を採用する．

結果を図 8 及び 9 に示す．実験結果から，事後確率に基づく提案手法は分類性能を大きく損なうことなく効率的に決定木を削減できていることがわかる．ノイズの割合が少ない場合には，ランダムに決定木を削除しても，事後確率に基づく提案手法と大きな差はないことも見て取れる．

4.4　実験 3: ワーカ数変化による性能の評価

MapReduce による random forests の学習における学習時間と分類誤差を評価する．本実験では，決定木の総数を 100 とし，ワーカ数を 1 から 100 まで変化させ評価する．このとき，データを分割する際に分布の偏りは発生しないものとする．提案手法の比較対象として，実験 1 と同様に，共有データを用いない random forests による学習 (RF) を採用する．

結果を図 10 及び 11 に示す．結果から，ワーカ数を増やすことで，分類性能が低下するものの，学習時間が大幅に減少することがわかる．提案手法では，分類

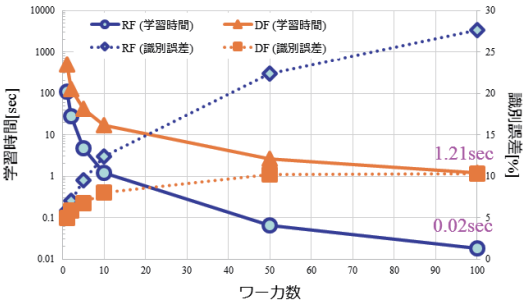

図 10　ワーカ数による学習時間と分類誤差の変化 (letter recognition)
Fig. 10　Trend on classification error and computational cost for number of workers (letter recognition).

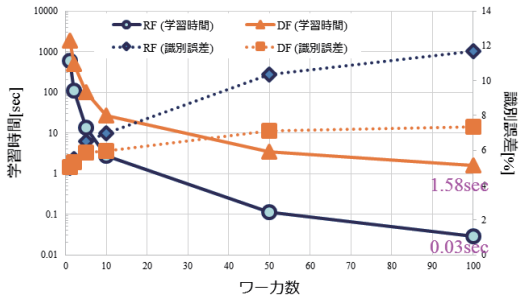

図 11　ワーカ数による学習時間と分類誤差の変化 (MNIST)
Fig. 11　Trend on classification error and computational cost for number of workers (MNIST).

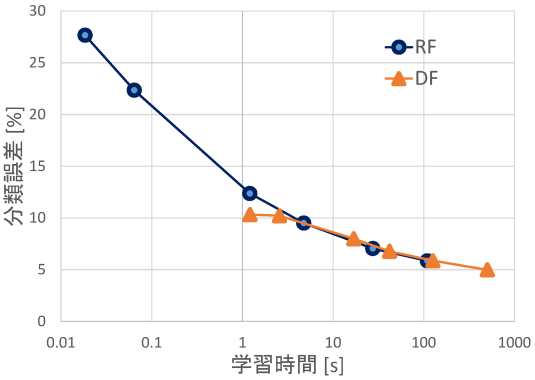

図 12　学習時間に対する分類誤差の変化 (letter recognition)
Fig. 12　Trend on classification error for computational cost (letter recognition).

性能の低下はほとんど見られないが，従来法と比較して 2 倍の決定木を構築する必要があることと，共変量シフト適応に基づく重要度を算出する必要があるため，

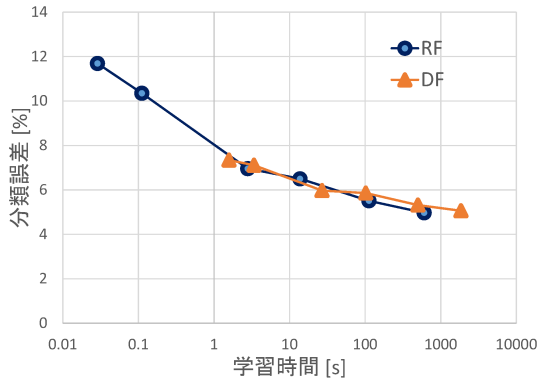

図 13　学習時間に対する分類誤差の変化 (MNIST)
Fig. 13　Trend on classification error for computational cost (MNIST).

従来と比較して計算量が必要となる．しかし，ワーカ数が増えるにつれて学習時間の絶対的な差は小さくなり，十分なワーカ数を確保することで学習時間の差は無視できる程度に小さくなることは見て取れる．

学習時間と分類性能とのトレードオフをより詳細に分析するために，図 10 及び 11 に示したワーカ数による学習時間と分類誤差の変化に関するグラフを，横軸を学習時間，縦軸を分類誤差とする表記に変更した．その結果を，図 12 及び 13 に示す．この結果から，学習時間を多く要する範囲では提案手法と従来手法はほぼ同等の性能であるが，ワーカを数多く用意して学習時間を短縮した際には提案手法が従来手法を上回る性能を示したことが見て取れる．

5. む　す　び

本論文では，MapReduce の枠組を利用して大規模データから random forests を学習する新しい手法を提案した．Map 処理において転移学習を活用することで，並列分散処理に伴うデータ分布の偏りやデータ量の少なさに起因する過学習の問題を回避できるとともに，Reduce 処理において分類性能の向上に寄与しない決定木を削除することにより，分類性能を大きく損なうことなく分類時の計算コストを低減できる．並列分散学習の利用シーンとして，分散ファイルシステムモデルとマルチコアサーバモデルの二つのモデルを想定して，それぞれのモデルに適した MapReduce の処理を提案した．

提案手法では，共有データとスコア算出用データにより分類性能が大きく左右される．共有データとスコア算出用データの選択法を確立することが，今後の研究課題の一つである．また，分散ファイルシステム，マルチコアサーバ，それぞれのモデルにより適合したアルゴリズムとソフトウェア実装を行うとともに，超大規模データを処理するためのデータマネジメント手法の開発も，重要な研究課題である．

文　　献

[1] Y. Freund and R.E. Schapire, “Experiments with a new boosting algorithm,” Proc. International Conference on Machine Learning (ICML), pp.148–156, 1996.
[2] L. Breiman, “Random forests,” Mach. Learn., vol.45, no.1, pp.5–32, Oct. 2001.
[3] Y. LeCun, Y. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol.86, no.11, pp.2278–2324, 1998.
[4] J. Dean and S. Chemawat, “Mapreduce: Simplified data processing on large clusters,” Commun. ACM, vol.51, no.1, pp.107–113, 2008.
[5] C.-T. Chu, S.-K. Kim, Y.-A. Lin, Y.-Y. Yu, G. Bradski, A. Ng, and K. Olukotun, “Map-Reduce for machine learning on multicore,” Proc. Conference on Neural Information Processing Systems (NIPS), pp.281–288, 2006.
[6] 若山涼至，木村昭悟，山内悠嗣，山下隆義，藤吉弘亘，“並列分散処理における共変量シフトを導入した random forests の学習，” 信学技報，PRMU2014-193, 2015.
[7] R. Wakayama, R. Murata, A. Kimura, T. Yamashita, Y. Yamauchi, and H. Fujiyoshi, “Distributed forests for MapReduce-based machine learning,” Proc. IAPR Asian Conference on Pattern Recognition (ACPR), 2015.
[8] L. Breiman, “Bagging predictors,” Mach. Learn., vol.24, no.2, pp.123–140, Oct. 1996.
[9] T.K. Ho, “The random subspace method for constructing decision forests,” IEEE Trans. Pattern Anal. Mach. Intell., vol.20, no.8, pp.832–844, Aug. 1998.
[10] B. Li, Z. Chen, M.J. Li, J.Z. Huang, and S. Feng, “Scalable random forests for massive data,” Proc. Pasific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), pp.135–146, 2012.
[11] J. Han, Y. Liu, and X. Sun, “A scalable random forest algorithm based on mapreduce,” Proc. IEEE International Conference on Software Engineering and Service Science (ICSESS), pp.849–852, 2013.
[12] S. delRio, V. Lopez, J.M. Benitez, and F. Herrera, “On the use of MapReduce for imbalanced big data using Random Forest,” Information Science, vol.285, pp.112–137, 2014.
[13] 土屋成光，弓場　竜，山内悠嗣，山下隆義，藤吉弘亘，“共変量シフトに基づく transfer forest，” 信学技報，PRMU2014-27, 2014.

[14] M. Wang, W. Li, and X. Wang, “Transferring a generic pedestrian detector towards specific scenes,” Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.3274–3281, 2012.
[15] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, “Boosting for transfer learning,” Proc. International Conference on Machine Learning (ICML), pp.193–200, 2007.
[16] J. Pang, Q. Huang, S. Yan, S. Jiang, and L. Qin, “Transferring boosted detectors towards viewpoint and scene adaptiveness,” IEEE Trans. Image Processing, vol.20, no.5, pp.1388–1400, 2011.
[17] 杉山　将，“共変量シフト下での教師付き学習，” 日本神経回路学会誌，vol.13, no.3, pp.111–118, 2006.
[18] M. Sugiyama and M. Kawanabe, Machine learning in non-stationary environments - Introduction to covariate shift adaptation, MIT Press, 2012.
[19] H. Shimodaira, “Improving predictive inference under covariate shift by weighting the log-likelihood function,” J. Statistical Planning and Inference, vol.90, no.2, pp.227–244, 2000.
[20] P.M. Frey and D.J. Slate, “Letter recognition using Holland-style adaptive classifiers,” Mach. Learn., vol.6, no.1, pp.161–182, 1991.

（平成 27 年 10 月 15 日受付，28 年 2 月 19 日再受付，
5 月 6 日早期公開）

若山　涼至

2013 年中部大学工学部情報工学科卒業．2015 年同大学大学院博士前期課程修了．コンピュータビジョン，機械学習の研究に従事．

村田　隆英

2014 年中部大学工学部情報工学科卒業．2016 年同大学大学院博士前期課程修了．コンピュータビジョン，機械学習の研究に従事．

木村　昭悟　（正員：シニア会員）

1998 年東京工業大学工学部電気電子工学科卒業．2000 年同大大学院理工学研究科電気電子工学専攻修士課程修了．同年，日本電信電話 (株) に入社．2007 年東京工業大学大学院理工学研究科集積システム専攻博士課程修了，工博．現在，日本電信電話 (株) コミュニケーション科学基礎研究所メディア情報研究部主任研究員．パターン認識，データマイニング，コンピュータビジョンに関する研究開発に従事．IEEE，ACM SIGMM/SIGKDD/SIGWEB，情報処理学会各会員．

山下　隆義　（正員）

2002 年奈良先端科学技術大学院大学博士前期課程修了，2002 年オムロン株式会社入社，2011 年中部大学大学院博士後期課程修了 (社会人ドクター)，2014 年中部大学講師，人の理解に向けた動画像処理，パターン認識・機械学習の研究に従事．画像センシングシンポジウム高木賞 (2009 年)，電子情報通信学会情報・システムソサイエティ論文賞 (2013 年)，電子情報通信学会 PRMU 研究会研究奨励賞 (2013 年) 受賞．

山内　悠嗣　（正員）

2012 年中部大学大学院博士後期課程修了．2012 年中部大学院博士研究員，2014 年中部大学助手．2010 年独立行政法人日本学術振興会特別研究員 DC2．コンピュータビジョン，パターン認識の研究に従事．

藤吉　弘亘　（正員：シニア会員）

1997 年中部大学大学院博士後期課程修了．1997-2000 年米カーネギーメロン大学ロボット工学研究所 Postdoctoral Fellow．2000 年中部大学講師，2004 年同大准教授を経て 2010 年より同大教授．2005-2006 年米カーネギーメロン大学ロボット工学研究所客員研究員．計算機視覚，動画像処理，パターン認識・理解の研究に従事．2005 年ロボカップ研究賞．2009 年情報処理学会論文誌コンピュータビジョンとイメージメディア優秀論文賞，2009 年山下記念研究賞．2010 年，2013 年，2014 年画像センシングシンポジウム優秀学術賞．2013 年電子情報通信学会情報・システムソサイエティ論文賞．博士（工学）．情報処理学会，IEEE 各会員．