社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

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

若山　涼至† 　木村　昭悟†† 　山下　隆義† 　山内　悠嗣† 　藤吉　弘亘†

† 中部大学 〒 487-8501 愛知県春日井市松本町 1200
†† 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 〒 243-0198 神奈川県厚木市森の里若宮 3-1
E-mail: † {rw@vision., yamashita@, yuu@vision., hf@}cs.chubu.ac.jp, ††akisato@ieee.org,

あらまし　データの大規模化は，統計的機械学習において高い識別性能を得るために重要な要件の 1 つであるが，学習時間を増加させる問題がある．しかし複数の計算機や GPU を用いて並列分散処理を適切に用いることができれば，学習時間を大幅に削減できる．本研究では，効率的に並列分散処理を行うモデルとして MapReduce を採用し，MapReduce モデルに適した識別器の 1 つである Random Forests の並列分散学習法を提案する．Map 処理では，並列分散学習に共有データを導入し，転移学習を行うことで各ワーカノードに割り当てられた学習データあ少量であっても高い識別性能を獲得する．Reduce 処理では決定木を削除することで識別時の計算コストを削減する．提案手法では，各ワーカノードに分布の偏りが強いデータが与えられた場合においても，識別性能の低下を防ぐことが可能である．
キーワード　ランダムフォレスト，転移学習，並列分散処理，機械学習，MapReduce

Training of Random Forests Using Covariate Shift on Parallel Distributed Processing

Ryoji WAKAYAMA†, Akisato KIMURA††, Takayoshi YAMASHITA†, Yuji YAMAUCHI†, and Hironobu FUJIYOSHI†

† Chubu University, 1200, Matsumoto , Kasugai, Aichi, 487-8501 Japan
†† Communication Science Laboratories NTT Corporation, 3-1, Morinosato Wakamiya Atsugi, Kanagawa, 243-0198 Japan
E-mail: † {rw@vision., yamashita@, yuu@vision., hf@}cs.chubu.ac.jp, ††akisato@ieee.org,

Abstract　Machine learning with big data improves a classification performance but increases computatinal cost for learning. Parallel distributed processing on multiple processors GPUs is often used to reduce processing time. This paper exploits MapReduce, an efficient framework for parallel distributed processing and proposes a novel method for training Random Forests by using the MapReduce framework. At the Map job stage, each worker trains a Transfer Forest with shared data to enhance classification performance. At the Reduce job stage, a reducer removes unreliable decision trees constructed at the Map stage, in order to reduce the computational cost of testing. The proposed method can retain the classification performance, even though unbalanced training samples are assigned to each worker.
Key words　Random forests, Transfer learning, Parallel distributed processing, Machine learning, MapReduce

1. はじめに

統計的学習法 [1] [2] [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識別・回帰，教師なし学習であるクラスタリングなど多岐にわたる．これらのタスクを解くために，統計的学習法では学習データを利用して学習器を構築する．統計的学習法において，学習器の構築に利用するデータ量は重要であり，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，十分に時間をかければ高い性能を持つ学習器が得られるようになった．データ量が増加することで，機械学習では難しい問題を解くことを可能としているが，処理時間が増加する問題がある．しかしこのようなビッグデータを処理するための処理コストは膨大であり，単一の計算機による処理は困難

である．この問題を解決するために複数の計算機や 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ためのモデルとして MapReduce [4] が提案されている．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 処理ではデータをマスタノードへ入力し，マスタノードはワーカノードへデータを割り振る．このとき，ワーカノードが更に細かく分割し他の複数のワーカノードに配置するなど階層を増やして分割を行う場合もある．ワーカノードは得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，マスタノードは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り，リソースを用意できれば 3 ペタバイトサイズのデータの処理を 1ヶ月で行うことが可能である [4]．

本研究では，MapReduce のフレームワークを用いた Random Forests の学習法について提案する．統計的学習法の一つである Random Forests は，独立した複数の決定木により構成されるアンサンブル学習法である．決定木構造の識別器のため，高速に識別することが可能であり，独立した複数の決定木を構築するため並列化が容易である．この特性から，Random Forests は並列分散処理による学習によく用いられている [5] [6]．MapReduce により Random Forests を学習することで学習時間を大幅に削減することが可能であるが，ワーカノードに割り当てられるデータの分布に大きな偏りが発生する場合やデータ量が少ない場合には過学習により識別性能が低下する問題がある．そこで本研究では，Map 処理においてワーカノードに共有データを与えることで分布の偏りによる過学習を抑制する．このとき，共有データが各ワーカノードに与えられたデータの学習に悪影響を及ぼさないために，転移学習を利用する．さらに，Reduce 処理において決定木の削除を行うことで，識別時の処理コストを削減する．

2. Transfer Forest [7]

本研究では，転移学習を利用して並列分散処理による学習を行う．本章では，提案手法で用いる Random Forests の転移学習法である Transfer Forest 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，複数の手法が提案されている [8] [9] [10]．以下で Random Forests に共変量シフトに基づく転移学習を導入した Transfer Forest について説明する．Random Forests は，2001 年に Breiman により提案された，複数の決定木構造を持つマルチクラス識別器を構築するアンサンブル学習アルゴリズムである．Random Forests では決定木に Bagging [11] と同様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し，Random Feature Selection [12] を取り入れることで特徴ベクトルの次元数に関わらず高速に学習することが可能である．

2.1 事前ドメインと目標ドメイン

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識別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，事前サンプル数は大量に存在し，目標サンプルは事前サンプル数と比べて少数であるとする．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．本研究における転移学習は，事前サンプルと事前 RF，目標サンプルを用いて目標ドメインに適した識別器 (目標 RF) を学習する問題である．転移学習の際，事前サンプルは共変量シフト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．

2.2 Transfer Forest の学習

図 1 に Transfer Forest の学習の概要を示す．Transfer Forest の学習は以下の 4 つのステップで行われる．

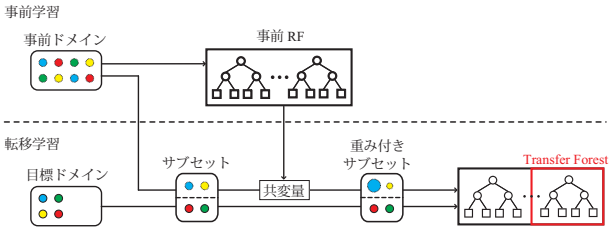


図 1　Transfer Forest

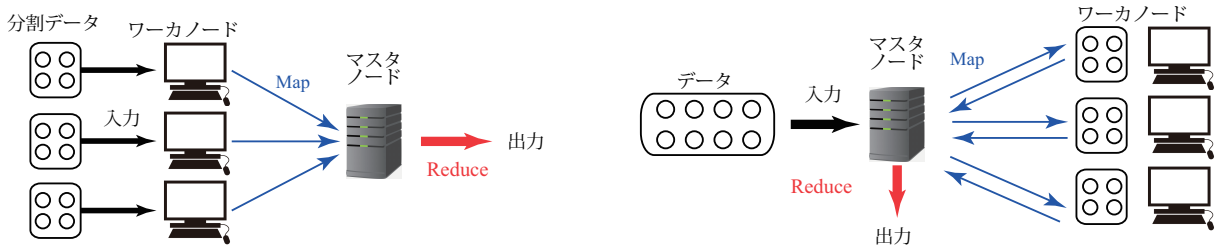
Step 1: サブセットの生成 事前サンプルと目標サンプルから同数のサンプルを選択し，目標 RF を構築するためのサブセットを生成する．

Step 2: 決定木の構築 Step 1 で生成されたサブセットから 1 本の決定木を構築する．ここで，サブセット内の事前サンプルは共変量 λ により重み付けされる．また，1 本目の決定木を構築する場合は事前サンプルの重みを 0 とする．決定木は式 (1) に従って分岐する．

$$d(s;\theta_j) = [f_{\phi_j}(s) \geqq \tau_n] \tag{1}$$

ここで j はノード番号，$[\cdot]$ は 0-1 の符号化関数，f_ϕ は分岐関数であり，この関数の反応値に閾値処理することで左右の分岐を決定する．$d(s;\theta_j)$ が 0 ならば左の子ノードへ，1 なら右の子ノードへ分岐する．Transfer Forest では，共変量を用いて目標サンプルの学習に有効でない事前サンプルの影響が小さくなるように事前サンプルの重み付けを行う．この重みはサンプル毎に異なるため，式 (2) により末端ノードのヒストグラム h を生成する．

$$h_c = \sum_{s \in \mathcal{S}_{ln}} i_{c(s)} \lambda_s \delta[c(s), c] \tag{2}$$



(a) マスタノード非通過型　　(b) マスタノード通過型

図 2　MapReduce のモデル

ここで $c(s)$ はサンプル s のクラスラベルを，$\mathcal{S}_{ln}$ は末端ノードが持つサンプル，λ_s はサンプル s の共変量を表し，δ はクロネッカーのデルタ関数である．i は，クラスラベル毎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: 共変量の更新 Step2 で構築した決定木を Transfer Forest に加え更新する．更新された Transfer Forest と事前 RF により，事前サンプル $\mathcal{S}^{so}$ の共変量 λ を更新する．k 番目の事前サンプル $\mathcal{S}^{so}_k$ の共変量 λ は式 (3) により求められる．

$$\lambda_k = \frac{1 + e^{P^{so}(\mathcal{S}^{so}_k)}}{1 + e^{P^{ta}(\mathcal{S}^{so}_k)}} \tag{3}$$

ここで，P^{so} は事前 RF の事後確率，P^{ta} は学習中の目標 RF の事後確率である．

Step 4: 決定木群の構築 以上のステップを繰り返し，指定した目標 RF の本数の 2 倍の数まで繰り返す．転移学習における事前サンプルの選択基準である λ は，初期状態では 0 が与えられ，決定木が追加される度に Transfer Forest と事前 RF の違いを正しく表現できるよう更新される．そのため，学習の序盤に構築された決定木における λ 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，本研究では信頼性の高い後半部分の決定木だけを利用することで，学習初期の悪影響を低減する．

Algorithm 1 に，Transfer Forest の学習のアルゴリズムを示す．

Algorithm 1 Transfer Forest の学習

Require: 事前サンプル集合 $\mathcal{S}^{so}$，目標サンプル集合 $\mathcal{S}^{ta}$，事前 RF

1. 事前処理
   クラスラベル c に対する係数 i_c の生成
   共変量の初期化　$\lambda_k = 0.0$
2. 目標 RF の構築
   for $n = 1$ to $2N$ **do**　　//N：目標 RF の決定木の本数
   i. 分岐ノードの生成
   ii. 末端ノードの確率ヒストグラム h の生成
   for $s = 1$ to S **do**　//S：学習サンプル数
   $h_c = \sum_{s \in \mathcal{S}_{ln}} i_{c(s)} \lambda_s \delta[c(s), c]$　　//$c()$：クラスラベル
   end for
   iii. 確率ヒストグラム h の正規化
   iv. 共変量の更新
   $\lambda_k = \frac{1 + e^{P^{so}(\mathcal{S}^{so}_k)}}{1 + e^{P^{ta}(\mathcal{S}^{so}_k)}}$　//P^{so}：事前 RF の事後確率　//P^{ta}：目標 RF の事後確率
   end for

Return $T_{(N+1)}, T_{(N+2)}, ..., T_{2N}$　//T_n：目標 RF の決定木

3. 提案手法による Map 処理

本研究では，MapReduce による Random Forests の学習法について提案する．本章では，MapReduce のモデルとして，図 2 に示すマスタノード非通過型とマスタノード通過型の 2 つを提案し，各モデルに対応した Map 処理について説明する．

3.1 マスタノード非通過型

マスタノード非通過型では，データはワーカノードへ直接入力される．そのため，データを分割・配置するコストが不必要でり，処理コストが低くなる．特に，大規模データの処理においては，データを分割・配置することは非常に大きな処理コストを発生させるため，このモデルの導入で大幅な処理時間の削減が期待できる．しかし，マスタノードによるデータ調整が不可能であるために，各ワーカノードに割り当てられたデータの分布に偏りが生じる場合がある．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識別性能が低下することが知られている．

そこでマスタノード非通過型では，ワーカノードが持つデータとは別に，全てのワーカノードで共有可能なデータを導入することによりデータの分布の偏りを抑制する．また，共有データからワーカノードが持つデータの学習に有効なデータを利用するために，Transfer Forest を用いて転移学習により学習を行う．この時，共有データを転移学習の事前サンプル，ワーカノードが持つデータを目標サンプルとして学習する．また，共有データからあらかじめ事前 RF が構築されているものとする．分布の偏りのない共有データを利用して各ワーカノードで学習を行うことで，各ワーカノードに与えられたデータに強い分布の偏りが発生しても過学習を抑制することが可能である．ワーカノードは Transfer Forest により構築した目標 RF をマスタノードへ返還する．図 3 にマスタノード非通過型の Map 処理の概要を，Algorithm 2 にマスタノード非通過型における Map 処理のアルゴリズムを示す．

3.2 マスタノード通過型

マスタノード通過型では，マスタノードへ一度データが入力され，それを分割しワーカノードへ配置する．マスタノード非通過型では，データ配置時に分布の調整を行うことができないために，分布の偏りが発生し過学習を引き起こす問題があるが，マスタノード通過型においては，マスタノードが配置先のワーカノードを指定可能であることから，サンプルの分布に偏

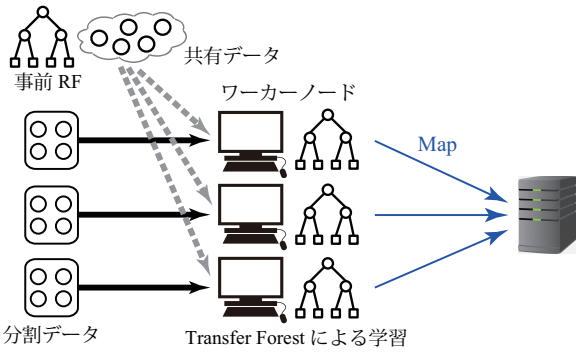


図 3　マスタノード非通過型における Map 処理

Algorithm 2 マスタノード非通過型における Map 処理

Require: 分割データ $\mathcal{D}_w$，共有データ $\mathcal{D}_s$

共有データ $\mathcal{D}_s$ から事前 RF を構築

Map 処理

1. ワーカノード w_i にそれぞれ分割データ $\mathcal{D}_{w_i}$ を配置
2. 各ワーカノードに事前 RF，共有データ $\mathcal{D}_s$ を与える
3. Transfer Forest により決定木群を並列に構築
4. 各ワーカノードで構築した決定木群をマスタノードへ返還

りが発生することは少ない．そのため，共有データを導入しない場合においても各ワーカノードが過学習を起こす可能性は低く，マスタノード非通過型と比較すると高い識別性能を得ることが可能であるが，データを分割・配置するためのコストが必要となる．しかし，データの分布が偏らない場合においても，ワーカノード数が増えることにより 1 つのワーカノードに対して学習に必要最低限なデータ量を確保することが難しくなる．Random Forests の学習では，サンプル数が少なくなると識別性能が低下する問題がある．そのため，ワーカノード数を増やすことで学習時間を大幅に削減できるが，データを重複しないように配置する場合には識別性能の低い識別器が構築されてしまう．そこで，マスタノード通過型においてもマスタノード非通過型と同様に共有データを利用する．共有データを導入することにより，全てのワーカノードに一定数のデータが与えられ，学習に必要最低限のデータ量を確保することが可能である．マスタノード通過型も非通過型と同様に，Transfer Forest により目標 RF を構築し，構築した決定木群をマスタノードへ返還する．図 4 にマスタノード通過型の Map 処理の概要を，Algorithm 3 にマスタノード通過型における Map 処理のアルゴリズムを示す．

Algorithm 3 マスタノード通過型における Map 処理

Require: データ $\mathcal{D}$，共有データ $\mathcal{D}_s$

共有データ $\mathcal{D}_s$ から事前 RF を構築

データ $\mathcal{D}$ をマスタノードへ入力

Map 処理

1. マスタノードがデータ $\mathcal{D}$ を $\mathcal{D}_{w_i}$ に分割
2. ワーカノード w_i にそれぞれ分割データ $\mathcal{D}_{w_i}$ を配置
3. 各ワーカノードに事前 RF，共有データ $\mathcal{D}_s$ を与える
4. Transfer Forest により決定木群を並列に構築
5. 各ワーカノードで構築した決定木群をマスタノードへ返還

3.3　各モデルの特徴

マスタノード非通過型は，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に処理コストが小さい．そのため，マスタノード非通過型による 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，マスタノード通過型はマスタノードによる配置データの調整が可能であるため，過学習が起こりにくくいためマスタノード非通過型よりも高い識別性能を得ることが期待できる．しかし，マスタノードがデータを分割・配置するために処理コストが高い．このことから，マスタノード通過型は高精度な並列分散処理による学習であることがいえる．また，両モデルにおいて共有データによる転移学習を利用することで識別性能は向上する．識別性能と処理コストはトレードオフであり，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．

4.　提案手法による Reduce 処理

マスタノードは Reduce 処理として，Map 処理で並列に構築された目標 RF を集約し，1 つの決定木群を生成する．並列分散処理で過剰に決定木を構築した場合には識別時に余分なコストを生み出す．そのため，決定木を削除することで識別時の計算コストを軽減する．

決定木の削除には，乱数を用いて削除する方法や，スコア算出用サンプルによりスコアを算出し決定木を削除する方法を用いる．乱数による決定木の削除では，Random Forests のランダム性を損なうことなく決定木を削除することができ，決定木を少ない処理コストで削除することが可能である．ワーカノード毎に保持するサンプル数が異なる場合や，サンプルにノイズが多く含まれるような場合には，決定木の性能に差が出やすいため，スコアによる決定木の削除を行うことで効率よく精度の高い決定木を残すことができる．本研究では，スコアは以下により算出する．

$$Score(t) = -\sum_{i=1}^{I} \max_{c \neq c_i} h(\mathbf{v}_i, c, t) \tag{4}$$

ここで，I はスコア算出用のサンプル数，$\mathbf{v}_i, c_i$ は，それぞれサンプルの特徴ベクトル，クラスラベルを表し，h は決定木 t が出力するクラス c の事後確率を表す．スコア関数は問題設定により適した関数を設計すること必要がある．Reduce 処理はマスタノード通過型，非通過型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 MapReduce のモデルに合わせて変更する．例えば，マスタノード非通過型においては，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．マスタノード通過型においては，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，全てのデータを利用してスコア算出することが可能である．Algorithm 4 に，スコアを用いた決定木削除における Reduce 処理のアルゴリズムを示す．

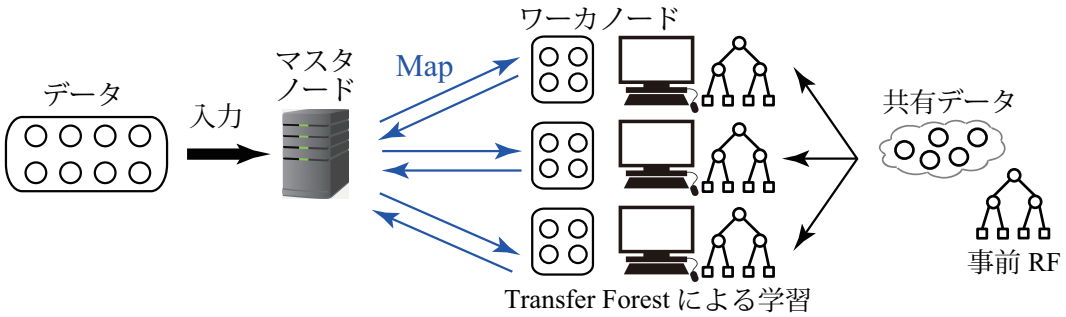


図 4　マスタノード通過型における Map 処理

Algorithm 4 提案手法による Reduce 処理

Require: 複数の決定木群 $\mathcal{T}_{w_i}$

各ワーカ w_i から決定木群 $\mathcal{T}_{w_i}$ を受け取る

Reduce 処理

1. 返還された複数の決定木群 $\mathcal{T}_{w_i}$ を一つの決定木群 $\mathcal{T}$ へ統合
2. データ $\mathcal{I}$ を用いて下式により決定木 t のスコア算出

$$Score(t) = -\sum_{i=1}^{I} \max_{c \neq c_i} h(\mathbf{v}_i, c, t)$$

3. スコアの低い木から順に指定した本数の木を削除

5. 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，Map 処理の評価のためにデータの配置時に分布の偏りが発生した場合の識別誤差を評価し，Reduce 処理の評価のために，乱数による決定木の削除とスコアによる決定木の削除による識別性能の変化の調査を行う．また，並列数による処理速度と識別精度の推移を測定する．

5.1 データベース

本実験では，UCI Machine Learning Repository letter recognition [13] と MNIST を用いて評価を行う．letter recognition は 20000 個のサンプルを持ち，クラス数 26，特徴次元 16 のデータセットである．MNIST は 70000 個のサンプルをもち，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 6666 個，MNIST の 10000 個を評価用サンプルとし残りを学習用サンプルに用いる．事前サンプルの数は letter recognition では 4000 個，MNIST では 10000 個とする．決定木のパラメータは予備実験により探索し，letter recognition は本数 50，深さ 15，特徴次元選択回数 15 回，閾値選択回数 50 回，MNIST では，本数 30，深さ 15，特徴次元選択回数 50 回，閾値選択回数 5 回とする．

5.2 実験 1：Map 処理の評価

本実験では，マスタノード非通過型 (MN 非通過型) とマスタノード通過型 (MN 通過型) における学習の評価を行う．マスタノード非通過型では，データ分布の調整が不可能であるために，データ分布に偏りがあるものとする．マスタノード通過型では，データ分布の調整が可能であるため，データ分布に偏りがないものとする．本実験において，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．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) と，共有データを用いた転移学習 (TF) を比較する．実験結果を表 1,2 に示す．実験結果から，マスタノード非通過型において，共有データを利用しない学習法では大きく識別性能が低下している．しかし，共有データを利用することで，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を抑制している．

表 1　識別誤差 (letter recognition)[%]

	MN 非通過型		MN 通過型	
ワーカ数	RF	TF	RF	TF
1	6.19	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	67.72	9.53	18.3	8.59

表 2　識別誤差 (MNIST)[%]

	MN 非通過型		MN 通過型	
ワーカ数	RF	TF	RF	TF
1	5.31	5.31	5.19	5.81
2	11.2	6.59	5.81	6.33
5	32.12	7.03	6.99	6.67
10	88.47	7.48	7.85	7.23

5.3 実験 2：Reduce 処理の評価

本実験では，学習サンプルにおけるクラスラベルのノイズの割合を変化させて学習した決定木を削除したときの識別誤差と決定木の削除本数を変えたときの識別誤差を評価する．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，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．ノイズの割合を変化させる実験では，ワーカ数を 10，木の削除本数を 20 とする．比較手法として，乱数による決定木の削除とスコアによる決定木の削除を比較する．結果を図 5,6 に示す．実験結果から，ノイズを含むような場合にはスコア算出により効率的に決定木を削除することが可能である．ノイズの割合が少ない場合には，乱数による決定木の削除とスコア算出による決定木の削除との差が小さいため，乱数により決定木を削除することで計算コストを小さくすることが可能である．

5.4 ワーカ数による学習時間と識別性能の推移の測定

MapReduce による Random Forests の学習における学習時間と識別誤差を評価する．本実験では，木の本数を 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．比較対象として共有データを用いない Random Forests による学習

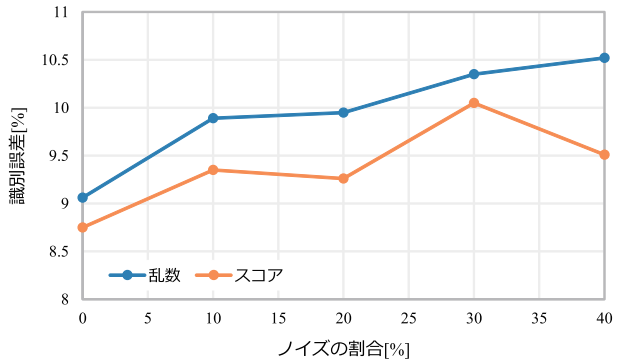


図 5　ノイズの割合による識別誤差 (letter recognition)

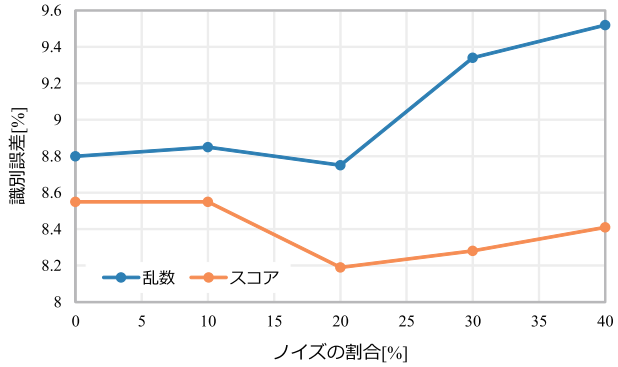


図 6　ノイズの割合による識別誤差 (MNIST)

(RF) と，共有データを用いた転移学習 (TF) を比較する．結果を図 7,8 に示す．結果から，ワーカ数を増やすことで，学習時間は減少するが識別性能は低下する．転移学習による決定木の構築では，2 倍の決定木を構築する必要があることと，共変量を算出するため従来の Random Forests よりも学習時間が増加している．しかし，ワーカ数が増えるにつれて学習時間の差は小さくなり，十分なワーカ数を確保することで十分に小さい学習時間で決定木を構築することが可能である．

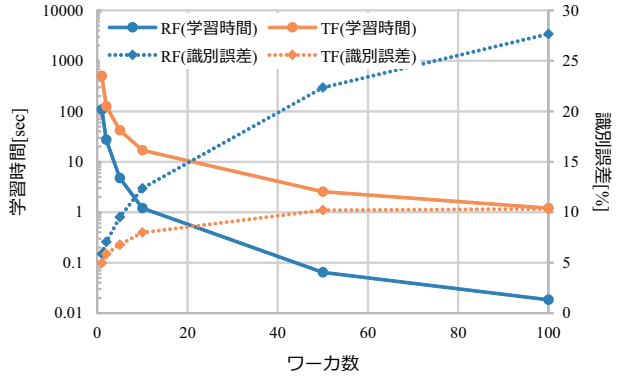


図 7　ワーカ数による学習時間と識別誤差の変化 (letter recognition)

6. ま と め

本研究では，Transfer Forest による並列分散環境の学習方法を提案した．Map 処理としてマスタノード非通過型，マスタノード通過型の 2 つの MapReduce のモデルにおける学習法を提案し，それぞれが持つ特徴について述べた．マスタノード非通過型におけるデータの分布に偏りが発生し過学習する問題

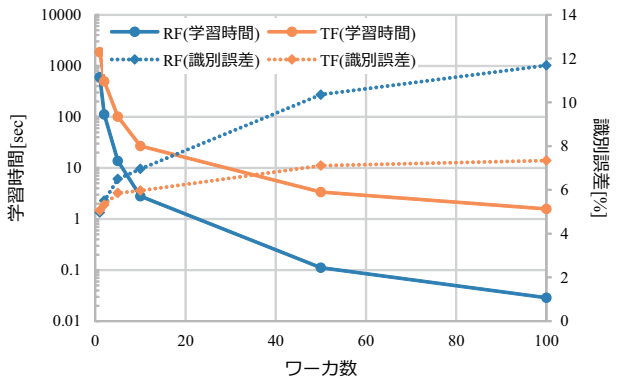


図 8　ワーカ数による学習時間と識別誤差の変化 (MNIST)

や，データ分割によりデータ量が不足し識別性能が低下する問題に対して共有データを利用することが有効であることを示した．また，Reduce 処理として決定木を削除する方法を提案し，効率的に決定木を削除することで，識別時の計算コストを削減した．提案手法において，共有データとスコア算出用データにより識別性能が大きく左右される．そのため今後は，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で，より高精度で効率的な並列分散学習を目指す．

文　献

[1] Y. Freund, and R.E. Schapire, Experiments with a New Boosting Algorithm, International Conference on Machine Learning, pp.148-156, 1996.
[2] L. Breiman, Random Forests, Machine Learning, vol.45, pp.5-32, 2001.
[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based Learning Applied to Document Recognition, Proceedings of the IEEE, vol.86, no.11, pp.2278-2324, 1998.
[4] J. Dean, and S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, Communications of the ACM, vol.51, no.1, pp.107-113, 2008.
[5] B. Li, X. Chen, M.J. Li, J.Z. Huang, and S. Feng, Scalable Random Forests for Massive Data, Advances in Knowledge Discovery and Data Mining, pp.135-146, Springer, 2012.
[6] J. Han, Y. Liu, and X. Sun, A Scalable Random Forest Algorithm Based on Mapreduce, Software Engineering and Service Science (ICSESS), 2013 4th IEEE International Conference on, pp.849-852, 2013.
[7] 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，vol.114, pp.31-36, 2014.
[8] M. Wang, W. Li and X. Wang, Transferring a Generic Pedestrian Detector Towards Specific Scenes. CVPR, 2012.
[9] W. Dai, Q. Yang, G. Xue and Y. Yu, Boosting for Transfer Learning, ICML, 2007.
[10] J. Pang, Q. Huang, S. Yan, S. Jiang and L. Qin, Transferring Boosted Detectors Towards Viewpoint and Scene Adaptiveness, IEEE Transactions on Image Processing, vol. 20, No. 5, pp.1388-1400, 2011.
[11] L. Breiman, Bagging Predictors, Machine Learning, vol.24, no.2, pp.123-140, 1996.
[12] T.K. Ho, The Random Subspace Method for Constructing Decision Forests, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol.20, no.8, pp.832-844, 1998.
[13] P.W. Frey, and D.J. Slate, Letter Recognition Using Holland-style Adaptive Classifiers," Machine Learning, vol.6, no.2, 1991.