Vol.2012-DBS-155 No.16
2012/11/19

MapReduceによる確率的勾配降下法を用いた広告クリック率予測の実践

後藤 康路[2,a)]　油井 誠[1,b)]　横山 昌平[3]　小島 功[1]　石川 博[3]

概要：本論文では，KDDcup 2012 track2 の商用検索エンジンの大量検索ログからの広告クリック率予測タスクを MapReduce 処理系である Hadoop 上で確率的勾配降下法 (Stochastic Gradient Descent) を用いて解いた事例を示し，大規模機械学習を実践的システムに適用したことにより得られた知見を示す．本論文の核となる貢献は，確率的勾配降下法による大規模なオンライン学習を Hive/Pig 上でそれぞれ実現した事例を示し，課題となる問題とその対処法などを明らかにすることにある．また，Hadoop 操作系として代表的な Hive と Pig について，それぞれの特徴，言語体系の違いによる学習器の実装への影響，性能差を述べる．

1. まえがき

検索エンジン事業者からブログやオンラインゲーム，SNS といったコンテンツ事業者まで，多くのインターネットサービス事業者が検索連動型広告やコンテンツ連動型広告によるクリック報酬（Pay-Per-Click）を主要な収入源としている．こうした事業者において広告クリック率（Click-Through-Rate（CTR））は収益を決定づける因子であるため，蓄積されたユーザデータや検索ログを利用して CTR を向上する研究開発が重要課題として取り組まれている [1], [2], [3], [4]．広告クリック率を正確に推定することで，インターネットサービス事業者は，広告の単価や表示順を最適化することができる．

KDD Cup 2012 の track 2 では，こうした産業界の需要に則した実践的なタスクとして，中国 Tencent 社の商用検索エンジン（soso.com）の検索ログを利用した検索エンジン広告のクリック率推定タスクが設定されている [5]．この Tencent 社から提供されているデータセットは, 149,639,105 レコードの Training データセットと 20,297,594 レコードの Test データセットから成り，各レコードが検索エンジンとユーザのインタラクション（セッションと呼ばれる）を表現する．図 1 に関係を示すように, Training データの各レコードは, (AdID, DisplayURL, AdvertiserID, KeywordID, TitleID, DescriptionID, UserID, QueryID, Depth, Position, #Impression, #Click) の 12 属性から成り，あるユーザ（UserID）が特定のセッションにおいて，AdID の広告に #Impression 回接触し，そのうち#Click 回広告をクリックしたことを表現する．検索語（QueryID）および広告の KeywordID，TitleID，DescriptionID は外部キーとして表現されており，hash 値によって数値化されたトークンの集合が外部キーによって参照される．Test データの構成は，Training データから#Impression, #Click を除いたものであり，残る 10 個の属性が特徴ベクトルを構成する．図 1 の training テーブルと user テーブルを結合して質的変数を二値素性に分解すると，特徴数は 54,686,452 である．

このタスクの課題は，正確な広告クリック率の推定器を開発することだけでなく巨大なデータを取り扱う点にある．補助データ構造を含めた Training データのサイズは TSV 形式で約 12GB であり，データサイズ，Training/Test インスタンス数のいずれも他の機械学習評価データセット [6] と比べ一桁大きい．このため，学習の収束時間や必要メモリ量から LIBLINEAR[7] 等の単一計算機で動作する機械学習ツールをそのまま適用することが困難であり，データ量に対してスケールする学習手法が必要である．

本論文では，KDDcup 2012 track2 の大量検索ログからの広告クリック率予測タスクを MapReduce 処理系である Hadoop 上で確率的勾配降下法（Stochastic Gradient Descent）を用いて解いた事例を示し，大規模機械学習を Hadoop で実施したことにより得られた知見を示す．本論文の核となる貢献は，確率的勾配降下法による大規模なオンライン学習を Hive/Pig 上でそれぞれ実現した事例を示し，課題となる問題とその対処法などを明らかにすること

[1] 産業技術総合研究所情報技術研究部門
[2] 静岡大学大学院情報学研究科
[3] 静岡大学情報学部
[a)] gs12015@s.inf.shizuoka.ac.jp
[b)] m.yui@aist.go.jp

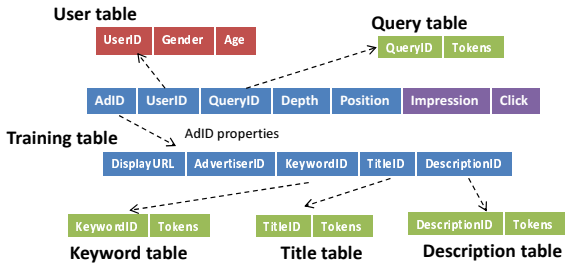

図 1　KDD Cup 2012 track2 のデータセットの構成
Fig. 1　The dataset composition of KDD Cup 2012 track 2

にある．これまでにも MapReduce 上で確率的勾配降下法を利用した取り組みは報告されている [1], [8] が，本論文では，既存手法の問題点を明らかにし，より関係演算に適した形でスケーラブルに機械学習を行う手法を提案する．

2. 関連研究

2.1 MapReduce による機械学習の並列処理

機械学習を MapReduce を利用して並列処理する手法はこれまでにも幾つか提案されている．

Chu らは，Statistical Query Model で記述可能な機械学習のバッチ学習アルゴリズムは MapReduce による並列処理が可能であり，多くの機械学習手法が Statistical Query Model で表現可能であるとしている [9]．実際に論文 [9] のアルゴリズムが Apache Mahout[10] に実装されている．Statistical Query Model は和の形（Summation Form）に変形できるため，各々の項を複数の計算ノード（mapper）に振り分けて処理をさせ，最後に reducer に集約して和をとればよい．この手法は機械学習を集約演算の一種と捉えることで，部分集約 [11] による並列化を行う手法と言える．

バッチ学習アルゴリズムは，多数のイテレーションを必要とするため，入出力が分散ファイルシステムを介す MapReduce とは相性が悪い．同一データを入力として繰り返し計算が行われるアルゴリズムのために，Spark[12] や Twister[13] は，中間結果を分散ファイルシステムではなくメモリ内に保持する in-memory の MapReduce 手法を提案している．これらの手法は中間データがメモリ内に収まる場合には効率的に動作するが，それぞれメモリに分割したデータが収まることが想定されているため，巨大データへの対処という点でデータ量に対するスケーラビリティには課題が残る．

2.2 オンライン学習の並列処理

バッチ学習アルゴリズムは学習の収束に多数のイテレーションが必要であり，無共有型（shared-nothing）で計算機クラスタの実行には課題が残っていた．これに対して，近年ではトレーニングインスタンスごとにモデルを逐次更新していく確率的勾配降下法（Stochastic Gradient Descent）や Passive-Aggressive を利用したオンライン学習 [14], [15] を大規模な機械学習に利用するという動きがある．オンラインアルゴリズムはパラメタが連続的に更新されるため，MapReduce との相性が悪いが，Algorithm 1 に示すように Iterative Parameter Mixing[8] により epoch 毎にパラメタを交換をすることで，訓練例をすべてメモリに保持することなく短い収束時間で大規模な学習を行うことができる．これに対し，epoch ごとのパラメタを交換を行わない学習手法は Parameter mixing[16] と呼ばれる．Parameter mixing に対して Iterative Parameter Mixing は分類器の識別性能ならびに学習の収束速度が速い [8] という利点がある．

Algorithm 1: Iterative Parameter Mixing を利用した Stochastic Gradient Descent アルゴリズム

$w = \emptyset$
$t = 0$
repeat
  $S = \{D^1, D^2, ..., D^K\}$ — *shuffle training data S into K partitions.*
  for $j = 1 \ldots K$ **do** {run on mappers.}
    $w_0^j = w_{t-1}$ — *mix parameters for each epoch.*
    for $i = 1 \ldots |D^j|$ **do** {For each instance, update weights based on gradient.}
      $w_i^j = w_{i-1}^j - \gamma^t \nabla F_{D_i^j}\left(w_{i-1}^j\right)$
  for $f = 1 \ldots |w|$ **do** {For each feature in the model, aggregate on reducers}
    $w^{t+1}(f) = \frac{1}{K}\sum_{j=1}^{k} w_{|D^j|}^j(f)$ — *calculate average of feature weights.*
  $t = t + 1$
until *converged*;

2.3 機械学習の関係データベースへの統合

機械学習を（関係）データベースのユーザ定義関数（User Defined Function（UDF））またはユーザ定義集約関数（User Defined Aggregate Function（UDAF））として，データベース内に閉じた形で分析処理を行うものとして代表的な研究に Madlib[17] と Bismarck[18] がある．

ロジスティック回帰，SVM，パーセプトロンをはじめ線形識別モデルをとる機械学習手法の多くが，勾配降下法を利用した凸最適化で解けることが分かっており，Bismarck[18] は，確率的勾配降下法と Parameter Mixing を利用した機械学習のための統一フレームワークを提案している．Bismarck では勾配の計算は UDAF として実装されている．Algorithm 1 に示したように，(Iterative) Parameter mixing による機械学習は集約処理の一種として捉えることができる．集約処理は結合法則と交換法則を満たす（commutative かつ associative である）とき，平均を求める計算 $avg(V)$ は $\frac{sum(V)}{count(V)}$ の形に変換することで部分集約による並列化が可能である [11].

Madlib や Bismarck はマルチプロセッサを有する単一計算ノードで動作する関係データベース向けの実装となっており，学習速度面でデータ量に対するスケーラビリティは MapReduce に基づく機械学習手法と比べ制限される．これに対し，本論文で用いる Hive のユーザ定義集約関数は，部分集約を，複数台の計算機を利用して，Combiner と MapReduce によって効率的に実行することができる [19].

3. MapReduce によるスケーラブルな機械学習

スケーラブルな機械学習を実現するための代表的なものに，確率的勾配降下法などを利用したオンライン学習を用いるアプローチと複数の（弱）予測器を統合するアンサンブル（ensemble）学習を利用するアプローチがある [1].

オンライン学習を用いるアプローチは，全訓練例をメモリに保持することなく，訓練例ごとに逐次モデルを更新していくため，必要メモリ量の面でデータ量に対してスケーラビリティがある．またオンライン学習には，学習収束速度がバッチ学習に比べて早いという特徴があり，こうした特徴から大規模の機械学習に適している．

アンサンブル学習は，独立（あるいは逐次的）に学習した複数の（弱）分類器を学習モデルを融合させて汎化能力（未学習データに対する予測性能）を向上させて，過学習を避ける手法である．独立に学習可能であるという embarrassingly parallel の特徴から，大規模なデータセットの訓練を並列実行する目的でも有効である．

本稿で議論する KDDCup 2012 track2 のタスクは，各セッションレコードごとのユーザプロファイルや広告のタイトルや記述を特徴ベクトルとしてクリック確率を予測という点で回帰分析タスクである．一方で，各インプレッションごとにクリックしたかどうかの二値クラス分類問題として捉えることも可能である．本稿では予測 CTR 値を事後確率 $p(y = +1|x)$（ただし，x は特徴量，y はクリックしたかどうかの 0/1 ラベル）として捉え，事後確率を特徴量を線形に回帰するためのモデルであるロジスティック回帰で求める．ロジスティック回帰分析を実装するにあたり，アンサンブル学習によるアプローチとオンライン学習によるアプローチをそれぞれ代表的な Hadoop 操作系である Hive と Pig で実装した．

アンサンブル学習によるアプローチでは，予測精度に優れるバッチ学習を並列に実行するユーザ定義関数を pig 上に実装した．弱学習器に与える訓練データはブートストラップ法によって生成し，データはバッチ学習でメモリ内に保持可能なように分割した．

オンライン学習によるアプローチではデータを分割せずに，Iterative Parameter Mixing を利用した確率的勾配降下法によるロジスティック回帰を Hive 上のユーザ定義関数として実装した．Pig/Hive も基本的には関係演算に基づく実行モデルをとるが，言語レベルで大きな違いがある．3.1 節と 3.2 節で，Pig/Hive それぞれで課題となる問題と対処法，操作系などの違いを議論する．

3.1 アンサンブル学習を利用した Pig によるアプローチ

Lin ら [1] は，Pig における確率的勾配降下法とアンサンブル学習を用いたロジスティック回帰分析を提案している．トレーニングにおいて，データセットからサンプリングを行う．サンプリングによってデータセットを分割し，並列に弱学習を行う．予測の際には，複数のモデルの予測結果から，アンサンブル学習を行う．二値判定問題では $\frac{(m+1)}{2}$ 個以上の弱学習器が誤判定を起こすと，多数決によりアンサンブルの判定は誤りとなるが，誤り確率を抑えることができることが経験的に知られている．手法としては，回帰問題において適用が容易なバギングを用いる．入力に対して各モデル毎に予測を行い，Voting を行うことで，スケーラビリティを維持したまま予測精度の向上を図る．しかし，論文 [1] で利用されている予測結果の Voting は入力の特徴量毎の重みを考慮しない．また，予測時に大量のモデル毎の結果を計算する必要があるという課題がある．モデル数を M、テストインスタンス数 N とすると，予測のアンサンブルでは予測に $O(M \times N)$ の時間計算量が必要となる．テストインスタンス数 N は大きいため，モデルのアンサンブルと比較して予測のサンサンブルは計算量が大きい．

そこで，提案手法ではモデルのアンサンブルを行い，精度の向上と予測時の計算量の削減を図る．モデルのアンサンブルでは，特徴量毎に各モデルが計算した重みに対して，重みの正負による Voting を行い，Voting の結果を支持した重みの平均値を取る．全ての特徴量に対してアンサンブルを行い，単一のモデルを作成する．モデルのアンサンブルを行う Pig スクリプトは次のとおりである．

```
model = load '$MODEL' as
       (modelid: int, feature: chararray, weight: float);
model_g = group model by feature;
ens = foreach model_g generate 0 as modelid,
              group as feature,
              WeightVoteAvg(model.weight);
store ens into '$ENS';
```

4.2 節でモデルのアンサンブルと予測結果のアンサンブルの予測精度の違いを議論する．

3.2 関係演算に基づく Hive を利用したアプローチ

Pig がアンサンブル学習によりデータ量に対するスケーラビリティを確保したのに対し，Hive による実装ではオンライン学習の並列処理によりスケーラビリティを確保した．本節では，Bismarck[18] に採用されている UDAF を利用した確率的勾配降下法の実装で問題となる点を述べ，UDTF による勾配の計算手法を提案し，関係演算により適した形で機械学習を行う上での知見を述べる．

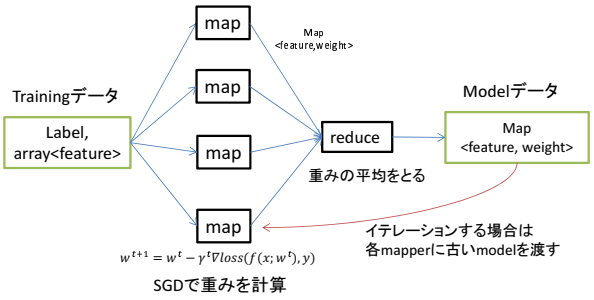

図 2 MapReduce による並列 training のデータフロー
Fig. 2 Data flow of parallel training with MapReduce

3.3 機械学習のデータフロー

ここでは，一般的な機械学習のデータフローを関係演算で処理する観点から入出力フォーマットを検討する．機械学習の学習器は，(label,vector<feature>) を各インスタンスとする訓練（training）データを入力として，特徴ごとの重み map<feature, weight>をモデルとして出力する．予測器は，map<feature, weight>をモデルとして入力にとり，テストデータの各インスタンス vector<feature> ごとに label または probability を出力する．

3.3.1 UDAF を利用する場合の reducer におけるボトルネック

Algorithm 1 を MapReduce で処理する場合には，図 2 に示すようなデータフローとなる．当初，我々は Bismarck の手法を踏襲し，各 mapper で確率的勾配降下法で計算した重みを reducer で平均する学習器を Hive でサポートされているデータ型である map<feature,weight>を出力として返す Hadoop の UDAF として実装した．

```
create table model as
select trainLogisticUDAF(features, label [, params]) as weight
from training
```

しかし，最終的に単一の reducer がスカラー値として重みを返すために，特徴数が多い場合に reducer でメモリ不足が発生する問題が発生した．KDDCup 2012 track 2 は特徴数が 54,686,452 と大きいため，重みの入出力が問題となる．UDAF ではリレーションではなくスカラー値を結果として返す必要があるため，大規模な機械学習には向かないと言える．これを対処する方法としては，3.1 節の手法と同様にバギングによるアンサンブル学習を導入して出力モデルのサイズを抑えることが考えられるが，4.3 節で検証するとおり，予測精度が犠牲となる可能性がある．

3.3.2 UDTF を利用した map-only 実装

上記のデータ量に対するスケーラビリティの問題に対処するために，より関係演算に適した形でモデルを relation<feature, weight>として返すことを考える．リレーションをトレーニング結果として返すために，User-Defined Table-generating Function（UDTF）を利用する．

入力タプルに対する出力が，UDF（User Defined Function）では One-to-One mapping され，UDAF では Many-to-One マッピングされるのに対して，UDTF では One-to-Many マッピングが可能である．Hive の実装では，入力タプルごとに $process(tuple_{in})$ 関数が呼び出され，入力が終わった段階で $close$ 関数が呼び出される．任意のタイミングで $forward(tuple_{out})$ を呼ぶことでタプルが結果リレーションに出力される．UDTF は，MapReduce の入力 split サイズに応じた mapper で，map-only のタスクとして並列に実行される．Hive による実装では，入力サイズを指定した複数の mapper を並列に訓練させ，それぞれの UDTF の呼び出しでは close のタイミングでモデルを出力するようにしている．UDAF では入力サイズが大きくなると reduce での集約処理でメモリ不足になる問題があったが，UDTF を利用した場合，データ量に対するスケーラビリティに限界が生じることはない．

Parameter Mixing による学習を行う Hive 問合せを以下に示す．

```
create table model as
select
  feature,
  CAST(avg(weight) as FLOAT) as weight
from
 ( select
     TrainLogisticSgdUDTF(features,label,..) as (feature,weight)
   from train
  ) t
group by feature;
```

なお，ここで各 mapper でのトレーニングの実行結果は feature によって shuffle されて，MapReduce によって並列に集約処理が行っている．

3.3.3 Iterative Parameter Mixing による学習

2.2 節に述べたように，Parameter Mixing と比べて，Iterative Parameter Mixing は学習速度と識別性能の点で優れる．MapReduce で Iterative Parameter Mixing を利用する場合には，図 2 に示したように epoch ごとに古い重みモデルを各 mapper に渡す必要があり，そのモデルの受け渡しにもデータ量に対するスケーラビリティに課題が残っていた．

確率的勾配降下法によるトレーニングで必要となるのは，各行の素性ごとに対応する古いモデルの重みである．そこで，我々の手法では古い重みについて各訓練例 ins の特徴ベクトル vector<feature>に対応するベクトル vector<weight>を relation<ins, vector<weight>> として生成し，内部結合してトレーニングの各インスタンスごとに特徴量の重みベクトルを渡すことで，スケーラビリティの問題に対処した．Iterative Parameter Mixing による学習を行う Hive 問合せを以下に示す．

```
select
   TrainLogisticIterUDTF(t.features, w.wlist, t.label, ..)
     as (feature, weight)
from
   training t join feature_weight w on (t.ins = w.ins)
```

以上にように，UDTF と関係演算に適したデータフローを考えることで，データ量に対してスケーラブルな大規模な機械学習を SQL の枠組みの上で実現可能である．

4. 評価実験

ここでは，Pig/Hive を用いてそれぞれ構築した学習手法の評価を KDDCup 2012 track 2 のタスクで評価する．

4.1 KDDCup 2012 track 2 のクリック率推定タスクでの評価

KDDCup 2012 track2 の広告クリック率推定タスクの提出フォーマットは，各セッションごとの予測 CTR（predicted CTR（pCTR））値であり，予測結果の精度評価指標は ROC 曲線下の面積（Area Under Curve（AUC））[20] によって行われる．AUC の値は，広告をクリックしたことを意味する正例と広告をクリックしなかったことを意味する負例をランダムに 1 個ずつ選んだとき，正例の予測値が負例の予測値以上になる確率を意味し，つまり，予測 CTR 値によって正例、負例の分類が正しく行われる確率を意味する．AUC は 0 から 1 までの値をとり，完全な分類が可能なときの面積は 1 で，ランダムな分類の場合は 0.5 となる．

4.2 Pig によるアンサンブル学習の予測精度の評価

モデルのアンサンブルの有効性を示すために予測精度に関する評価実験を行う．KDDCup 2012 track2 のデータセットを 500 個のデータセットに分割し，LIBLINEAR の L2-regularized Logistic Regression[21] を用いて 500 個のモデルを生成する．生成されたモデルに対して，モデルのアンサンブルを行った場合と予測結果のアンサンブルを行った場合のそれぞれについて，CTR を計算する．CTR の計算には，予測結果に対する平均値と，Voting による分類結果を支持した値の平均値を用いる．モデルのアンサンブルについても，同様に重みの平均値を取った場合の CTR も計算する．評価指標としては AUC を用いる．表 1 に実験結果を示す．

表 1 より，予測結果のアンサンブルよりも，モデルのアンサンブルの結果の方が精度が高くなる傾向にあった．また，モデルのアンサンブルの手法について，平均を取るよりも，Voting による分類結果を支持する重みの平均を取る方が精度が高くなる傾向にあった．これは，平均を取る重みを Voting 結果によってフィルタリングすることで，分割によるデータセットの偏りに左右されにくいアンサンブルが行われたと考えられる．

分割数の違いによるアンサンブル学習の精度差についても検証を行う．モデルのアンサンブルの 2 つ手法について，分割数を増やした場合の AUC の値を比較する．分割数が，500，1000，5000 の場合の，各手法毎の AUC の値を表 2 に示す．表 2 より，平均のみによるアンサンブルと比較して，Voting と平均によるアンサンブルの方が高分割数に対する精度の低下に強いと言える．

表 1 モデルのアンサンブルと予測結果のアンサンブルによる CTR 予測精度の比較

手法	AUC
モデルのアンサンブル (Voting+平均)	0.742452
モデルのアンサンブル (平均)	0.737929
予測結果のアンサンブル (平均)	0.734723
予測結果のアンサンブル (Voting+平均)	0.734545
単一のモデル	0.688233

表 2 分割数の変化に対する CTR 予測精度

分割数	単一モデル	Ensemble(平均)	Ensemble(Voting+平均)
500	0.688233	0.737929	0.742452
1000	0.681164	0.730641	0.741087
5000	0.651293	0.713578	0.733853

4.3 Hive を利用した Iterative Parameter Mixing による予測精度の評価

本節では，Iterative Parameter Mixing によって予測精度がどれだけ向上するかを評価する．4.2 節ではバッチ学習とアンサンブル学習を組合せによって大規模な機械学習を行ったが，表 3 に示すように確率的勾配降下法と Iterative Parameter Mixing の組合せによる Hive を利用した手法は予測精度の点でも Pig 版を上回る結果となった．

ここで，表 3 の pig batch (emsemble) は 4.2 節の最も良い学習手法，hive sgd (ensemble=5) は確率的勾配降下法を利用したオンライン（弱）学習器 5 つを利用してバギングを行った手法，hive sgd (param-mix) は確率的勾配降下法で求めたモデルを Paramater Mixing により統合した手法，hive sgd (iter=N) は Iterative Parameter Mixing と確率的勾配降下法を利用した手法をそれぞれ示す．N はイテレーション数を示す．

表 3 Iterative Parameter Mixing との他手法の CTR 予測精度の比較

手法	AUC
pig batch (emsemble)	0.742452
hive sgd (ensemble=5)	0.727432
hive sgd (param-mix)	0.738181
hive sgd (iter=3)	0.749916
hive sgd (iter=4)	0.752031
hive sgd (iter=5)	0.753602

hive sgd (ensemble=5) と hive sgd (param-mix) の比較から，今回の実験においては学習の入力とするトレーニングデータは多い方が望ましく，アンサンブル学習による汎化性能の顕著な向上は見られなかった．Iterative Parameter Mixing とオンライン学習の組み合わせは弱学習器をアン

サンブルさせるよりも，識別性能に優れることがあることを示した．hive sgd (param-mix) は hive sgd (iter=1) 等しい．このことからイテレーションを重ねることで，オンライン学習がバッチ学習に対して精度で勝ることがあることが分かる．

5. むすび

本稿では確率的勾配降下法による大規模なオンライン学習を Hive/Pig 上でそれぞれ実現した事例を示し，より関係演算に適した形でスケーラブルに機械学習を行う手法を提案した．これまでに提案されてきた UDAF に基づく実行形態にはデータ量に対するスケーラビリティに限界があることを示し，本論文で提案した UDTF と関係演算に適したデータフローを実施することで，データ量に対してスケーラブルな大規模な機械学習を関係データベースの上で実現可能であることを示した．

AUC を最大化することは，各インスタンスを Click 数の正例と（Impression 数-Click 数）の負例に分類して，その Pairwise ランキング損失を最小化することにあり，テストデータの予測 CTR 値の順序付け（ランキング）を正しく行うことが目標である．つまり，予測 CTR 値自体の精度は問われていないため，順序付けのためのランク学習 [22] の大規模機械学習への適用が今後の検討課題である．

謝辞 本研究の一部は，科学研究費若手研究 (B)（課題番号：24700111）ならびに基盤研究 (A)（課題番号：24240015）の支援による．ここに謝意を表す．

参考文献

[1] Lin, J. and Kolcz, A.: Large-Scale Machine Learning at Twitter, *Proc. SIGMOD*, pp. 793–804 (2012).
[2] Chandramouli, B., Goldstein, J. and Duan, S.: Temporal Analytics on Big Data for Web Advertising, *Proc. ICDE*, pp. 90–101 (2012).
[3] Chen, Y., Pavlov, D. and Canny, J. F.: Large-scale Behavioral Targeting, *Proc. KDD*, pp. 209–218 (2009).
[4] Yan, J., Liu, N., Wang, G., Zhang, W., Jiang, Y. and Chen, Z.: How much can behavioral targeting help online advertising?, *Proc. WWW*, pp. 261–270 (2009).
[5] KDD Cup 2012, Track 2: http://www.kddcup2012.org/c/kddcup2012-track2.
[6] LIBSVM: http://www.csie.ntu.edu.tw/˜cjlin/libsvm/.
[7] Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R. and Lin, C. J.: LIBLINEAR: A Library for Large Linear Classification, *J. Mach. Learn. Res.*, Vol. 9, pp. 1871–1874 (2008).
[8] Hall, K. B., Gilpin, S. and Mann, G.: MapReduce/Bigtable for Distributed Optimization, *Neural Information Processing Systems Workshop on Leaning on Cores, Clusters, and Clouds* (2010).
[9] Chu, C. T., Kim, S. K., Lin, Y. A., Yu, Y., Bradski, G. R., Ng, A. Y. and Olukotun, K.: Map-Reduce for Machine Learning on Multicore, *Proc. NIPS*, pp. 281–288 (2006).
[10] Apache Mahout: http://mahout.apache.org/.
[11] Larson, P.-Å.: Data Reduction by Partial Preaggregation, *Proc. ICDE*, pp. 706–715 (2002).
[12] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S. and Stoica, I.: Spark: Cluster Computing with Working Sets, *Proc. HotCloud*, p. 10 (2010).
[13] Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J. and Fox, G.: Twister: a runtime for iterative MapReduce, *Proc. HPDC*, pp. 810–818 (2010).
[14] Bottou, L.: Large-Scale Machine Learning with Stochastic Gradient Descent, *Proc. COMPSTAT*, pp. 177–187 (2010).
[15] Crammer, K., Dekel, O., Keshet, J., Shwartz, S. S. and Singer, Y.: Online Passive-Aggressive Algorithms, *J. Mach. Learn. Res.*, Vol. 7, pp. 551–585 (2006).
[16] Mann, G., McDonald, R. T., Mohri, M., Silberman, N. and Walker, D.: Efficient Large-Scale Distributed Training of Conditional Maximum Entropy Models, *Proc. NIPS*, pp. 1231–1239 (2009).
[17] Hellerstein, J. M., Ré, C., Schoppmann, F., Wang, D. Z., Fratkin, E., Gorajek, A., Ng, K. S., Welton, C., Feng, X., Li, K. and Kumar, A.: The MADlib analytics library: or MAD skills, the SQL, *Proc. VLDB*, Vol. 5, No. 12, pp. 1700–1711 (2012).
[18] Feng, X., Kumar, A., Recht, B. and Ré, C.: Towards a unified architecture for in-RDBMS analytics, *Proc. SIGMOD*, pp. 325–336 (2012).
[19] White, T.: *Hadoop: The Definitive Guide*, O'Reilly Media (2009).
[20] Bradley, A. P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms, *Pattern Recognition*, Vol. 30, No. 7, pp. 1145–1159 (1997).
[21] Lin, C. J., Weng, R. C. and Keerthi, S. S.: Trust Region Newton Method for Logistic Regression, *J. Mach. Learn. Res.*, Vol. 9, pp. 627–650 (2008).
[22] Sculley, D.: Large Scale Learning to Rank, *Proc. NIPS Workshop on Advances in Ranking* (2009).