Vol.2015-DBS-162 No.4
2015/11/26

クレジットカード加入審査結果データセットを用いた複数の機械学習手法の性能比較

池田政人[†1]　新美礼彦[†2]

概要：本研究では，Deep Learning がクレジットカードデータの分析にどの程度有効であるかを確かめるため，UCI Machine Learning Repository から入手可能な Credit Approval Data Set（クレジットカード加入審査結果データセット）を性能評価データとして機械学習の手法の一つである Deep Learning を用いて性能評価実験を行った．他の機械学習の手法であるロジスティック回帰とガウシアンカーネル SVM，線形カーネル SVM，ランダムフォレスト，Xgboost を用いた結果と Deep Learning を用いた結果を比較したところ，ガウシアンカーネル SVM が正解率 0.89 となり一番高い正解率となったが，Deep Learning は 0.87 となり 2 位の正解率になった．トランザクションデータに対する性能評価として実データを元にしたテストデータを利用し，Deep Learning による不正検知モデルを作成し，不正検知率と誤検知率を算出した．

キーワード：機械学習，Deep Learning，クレジットカード

1. はじめに

昨今のインターネットの普及によりインターネットショッピングを利用した商品の購入が増加している．また，コンビニエンスストア等での小額決済も増額しており，クレジットカードを利用する機会が増えたことから今後ますます利用する機会が増加すると考えられる．また，複数枚のクレジットカードを保有しているケースも少なくない．

クレジットカードはその場で現金を必要とせず，商品やサービスの購入・利用決済が可能であるがその反面，カード番号の流出や偽造カードの利用等，不正利用が社会的な問題となっており平成 26 年の金額ベースでの不正利用は 106 億円 1)であり，同年の不正利用を含めた利用額は 40 兆円 2)である．金額ベースでの不正利用発生確率は 0.0265%であり一見低く見えるが，先に述べた通り不正利用額は 106 億円である．

本研究では，Deep Learning 3)を用いたクレジットカード分析モデルの性能を評価するため，Deep Learning による分析モデルと他の機械学習を用いた分析モデルを構築し，結果を考察した．第 2 章では比較に用いた機械学習について述べる．第 3 章では Deep Learning について述べる．また，実験で用いたデータセットについては第 4 章で述べる．第 5 章で実験設定について，第 6 章でその結果について考察する．トランザクションデータを用いた実験については第 7 章にまとめた．実験では不正利用検知率と誤検知率に注目した．

2. 比較に用いた機械学習手法

今回，Deep Learning 以外に実験を行った機械学習手法，統計分析手法を挙げる．

2.1 ロジスティック回帰

ある事象が起こる確率を予測する統計分析手法の 1 つで 2 値分類問題に適しており，予測結果が 0 から 1 の間を取るように目的変数と説明変数から線形的な合成関数を用いて定量的に分析する．

2.2 サポートベクターマシン（SVM）

サポートベクターマシンとは，特徴空間上で線形分離できない際にソフトマージンと呼ばれる線形分離の条件を緩めて線形分離可能にする．SVM のひとつであるガウシアンカーネル SVM は線形分離不可能パターンに強く，データに関する事前知識がない場合に用いられる汎用的なカーネル法である．線形カーネル SVM は，線形であり，サポートベクターマシンにおいて，訓練データが大規模で疎であるデータ（ほとんどの項目が 0 の場合）は線形分離可能であれば特徴空間に写像する必要がないため，カーネル法を用いない線形のサポートベクターマシンとなる 5).

2.3 ランダムフォレスト

決定木による分類器を複数組み合わせてアンサンブル学習を行う際に，決定木の各ノードで選択する属性をランダムに選ぶ手法である 6).

2.4 Xgboost

複数の決定木による分類器をアンサンブル学習において，逐次的に学習され，個々の分類器が苦手とするデータに対して，より分類能力が高くなるように個々の分類器を組み合わせ学習することを勾配ブースティングと呼ぶ．勾配ブースティングの高速な C++の実装である 6) 7) 8).

3. Deep Learning

Deep Learning ニューラルネットワークの発展版であり今までの階層型ニューラルネットワークよりも多くの隠れ層を用いるのが特徴である．また，自己符号器を用いることにより，入力層と出力層の両方に学習させたいデータの正解例を読みこませ，例えば隠れ層を 200 に設定すること

†1 公立はこだて未来大学大学院 システム情報科学研究科
Future University Hakodate, Graduate School of System Information Science
†2 公立はこだて未来大学 システム情報科学部
Future University Hakodate, Faculty of System Information Science

でデータを 200 個の特徴量で表現できるようになる．自己符号器による特徴量の圧縮が繰り返し，最終的には特徴量としてベストなものが残る 3).

Deep Learning を利用してモデルを高速で作成・判定を行うために，様々な実装が提案されている，今回は，リアルタイム処理を Apache Hadoop より 10 倍高速とされる Apache Spark 9)上で Deep Learning が実行可能である H2O の Sparkling Water 10)を利用した．

Sparking Water に実装されている活性関数を表 1 に示す 11).

表 1　Sparkling Water で利用されている活性関数

関数	式
Tanh	$f(\alpha) = \frac{e^{\alpha} - e^{-\alpha}}{e^{\alpha} + e^{-\alpha}}$
Rectified Linear	$f(\alpha) = \max(0, \alpha)$
Maxout	$f(\cdot) = \max(w_i x_i)$, $rescale\ if\ \max f(\cdot) \geq 1$

4. クレジットカードデータセット

Deep Learning と他手法の比較実験には UCI Machine Learning Repository から入手可能である Credit Approval Data Set 12)を Takashi J. OZAKI 氏が欠損値処理を行ったデータセット 13)を用いた．属性は連続値の離散化や，オリジナルの要素を推定不可能にするために加工が行われている．データセットの概要を表 2 に示す.

表 2　データセットの概要

インスタンス数	690
属性数	15+判定結果（+, -）
データ分布	+:307 (44.5%), -:383 (55.5%)
トレーニングデータ数	590
テストデータ数	100

5. 実験環境

実験は統計分野や機械学習で用いられる R 14)を(1)で示す環境と，Sparkling Water は(2)で示す．Deep Learning による実験では Sparkling Water を，他の機械学習による実験では R を用いた．R による実験は MacBook Air で，Sparkling Water による実験は Amazon Web Services の EC2 環境 15)で行った．Amazon Web Services の EC2 環境による実験はどの程度のスペックのインスタンスが分析に適しているかを検討するために処理時間や処理能力の測定を兼ねている．

(1) **R の実行環境**

- MacBook Air 2013 mid
- Mac OS X 10.10 Yosemite
- CPU Intel Core i7 1.7GHz
- メモリ 8 GB
- R 3.2.2

(2) **Sparkling Water の実行環境**

- Amazon Web Services EC2 t2.micro
- Amazon Linux AMI
- CPU Intel Xeon 3.3GHz
- メモリ 1GB
- Spark 1.3.1
- H2O Sparkling Water 0.2.101
- 隠れ層 2 層 (200,200)
- 活性関数（Rectifier）

なお, R の実行についてはインストール直後に入っていないライブラリを都度追加した．ライブラリは下記の通りである．

- e1071
- randomForest
- Xgboost, Matrix

6. 実行結果

4 章にて述べたデータセットを利用し，機械学習の手法で実験を行った．テストデータに対する推定結果について Confusion Matrix の形で表 3 にロジスティック回帰，表 4 にガウシアンカーネル SVM，表 5 に線形カーネル SVM，表 6 にランダムフォレスト, 表 7 に Xgboost, 表 8 に Sparkling Water の順に示す．N,Y は推定する属性を示す．一部のアルゴリズムでは数値属性として推定するため，N,Y の代わりに 0,1 を用いた.

表 3　ロジスティック回帰

	0	1
N	42	8
Y	10	40

正解率 = 0.82

表 4　ガウシアンカーネル SVM

	N	Y
N	42	8
Y	3	47

正解率 = 0.89

表 5　線形カーネル SVM

	N	Y
N	42	8
Y	6	44

正解率 = 0.86

表 6　ランダムフォレスト

	N	Y
N	45	5
Y	9	41

正解率 = 0.86

表 7　Xgboost

	N	Y
N	44	6
Y	8	42

正解率 = 0.86

表 8　Sparkling Water(Deep Learning)

	N	Y
N	41	9
Y	4	46

正解率 = 0.87

7. 考察

表 9 において，正解率の順位を整理する．

表 9　正解率の順位

順位	手法名	正解率
1	ガウシアンカーネル SVM	0.89
2	Sparkling Water	0.87
3	線形カーネル SVM	0.86
3	ランダムフォレスト	0.86
3	Xgboost	0.86
6	ロジスティック回帰	0.82

利用したデータの分布が滑らかであり，木構造を用いるランダムフォレストと Xgboost では上手くアンサンブル学習ができなかったと考えられる．線形分離可能とは言えないデータではないため，ロジスティック回帰と線形カーネル SVM も苦戦したと考えられる．結果として，線形分離不可能パターンに強く，データに関する事前知識がない場合に用いられる汎用的なガウシアンカーネル SVM が一番高い正解率が得られたと考えられる．

Sparkling Water(Deep Learning)の正解率は第2位であったが，他手法と比較して性能に差がないことが確認できた．

Deep Learning はデータ数が多い方が性能が出やすいが，本実験で利用したデータ数が少ないか，多いかという議論については少ないと考えているため，特徴量の圧縮が十分に行われないまま終わってしまったと考えている．隠れ層の値の調整など，パラメータの変更で正解率が変化する他が今回の実験ではデフォルトのパラメータを用いている．使用するデータ数やパラメータチューニングにより性能向上も考えられる．

Amazon Web Services の EC2 インスタンスについては第5章にて述べたスペックにおいて，実行時間の平均は 15 秒程度であり，特にメモリが不足することはなかった．一般的に Deep Learning には大量のメモリが必要となるが，今回の実験で用いたデータの規模であればそれほどメモリが多くなくとも処理が可能であることが確認できた．実行時間に関しては，他の機械学習手法とは実験環境が異なるため，一概には比較できない．

8. トランザクションデータを用いた実験

追加実験として，第 5 章で述べた EC2 インスタンスに構築した Sparkling Water 上でトランザクションデータを用いた不正利用検知モデルの構築と性能評価を行った．Deep Learning のパラメータはデフォルトのものを用いた．実際のクレジットカード利用データからモデル構築し生成したデータ用いて不正利用検知モデルを構築し，不正利用検知率（全不正のうち，どの程度不正を検知できたか）と誤検知率（不正と判断したデータのうち，どの程度不正が含まれていたか）を算出した．データは 2015 年 9 月 1 日から 2015 年 9 月 30 日の 1 ヶ月分であり，1 日あたり 1100 件である．そのうちテストデータ生成時に不正利用とされた件数は 1100 件中最大 8 件である．トレーニングデータは 2015 年 9 月 n 日のデータを利用し，テストデータは 2015 年 9 月 n+1 日のデータを利用し，29 パターンの実験を行った．例えばトレーニングデータとして 2015 年 9 月 1 日のデータを利用し，テストデータとして 2015 年 9 月 2 日のデータを利用する．表 10 に不正利用検知率と誤検知率の平均値を示す．

表 10　不正利用検知率と誤検知率（平均値と分散）

	不正利用検知率（トレーニングデータ）	誤検知率（トレーニングデータ）	不正利用検知率（テストデータ）	誤検知率（テストデータ）
平均値	0.4724	0.4129	0.7693	0.9522
分散	0.0573	0.1274	0.0424	0.0050

不正利用検知率は 1 に近づくにつれ，誤検知率は 0 に近づくにつれ良いとされているが，翌日のデータをテストデータとしているため，日によって結果にばらつきがあった．これはモデルの作成に利用したデータ数が少なく，十分にモデル構築が行えなかったためと考えられる．例えば 1 週間単位や 1 ヶ月単位でモデルを生成すれば決まった曜日や日に同じようなクレジットカードの利用パターンが出現し，

不正利用検知率と誤検知率が向上する可能性がある．

本実験において，先に述べた EC2 インスタンスの Sparkling Water 環境ではメモリが不足したため，実行コマンドの引数においてメモリを 512MB 確保した．実行時間は 14 分 13 秒であった．実メモリ以上のメモリを指定したため，スワップが発生し実行時間が遅くなる原因になったと思われる．

9. まとめと今後の展開

本論文では，Deep Learning を用いたクレジットカード分析モデルの性能を評価するため，他の機械学習によるモデルと比較する実験を行った．実験の結果，ガウシアンカーネル SVM に次ぐ性能を確認できた．Deep Learning ではデフォルトのパラメータを用いているため，学習に使うデータ数と共に検討を行えば，より公正なモデル構築が期待できる結果となった．また，トランザクションデータに対する Deep Learning による不正利用検知モデルの構築を行い，性能を評価したが，不正利用検知率・誤検知率共に良い結果は得られなかった．

現段階では，クレジットカードの利用データについて Sparkling Water(Deep Learning)のデフォルトのパラメータや関数では不正利用の検知に有効であるとは言えない．パラメータや関数を工夫する必要があり今後取り組む．

今回の実験で使用した Deep Learning のライブラリは Spark 上で動いているため，Spark の特徴であるストリーミング処理に対応した不正利用検知のためのアプリケーションについても検討を行う．

謝辞 本研究は株式会社インテリジェント ウェイブとの共同研究の一環であり，テスト用クレジットカード利用データの提供と不正利用検知モデルへのコメントを頂いた．謹んで感謝の意を表する．

参考文献

1) 一般社団法人日本クレジット協会:クレジットカード不正利用被害の発生状況(オンライン), 入手先〈http://www.j-credit.or.jp/information/statistics/download/toukei_03_g.pdf〉(参照 2015-10-14)
2) 一般社団法人日本クレジット協会:クレジットカード動態調査集計結果(オンライン), 入手先〈http://www.j-credit.or.jp/information/statistics/download/toukei_03_c.pdf〉(参照 2015-10-14)
3) 深澤祐援:人工知能 : ディープラーニングとは何なのか？ そのイメージをつかんでみる (1/5) - ITmedia ビジネスオンライン(オンライン), 入手先〈http://bizmakoto.jp/makoto/articles/1507/27/news067.html〉(参照 2015-10-11)
4) Takashi J OZAKI:UCI 機械学習リポジトリのデータ（など）で遊ぶ(3) : クレジットカードの加入審査データ - 銀座で働くデータサイエンティストのブログ(オンライン), 入手先〈http://tjo.hatenablog.com/entry/2015/06/12/190000〉(参照 2015-10-10)
5) 竹内一郎, 鳥山昌幸: サポートベクトルマシン, 機械学習プロフェッショナルシリーズ, 講談社(2015)
6) 元田浩, 津本習作, 山口高平, 沼尾正行: データマイニングの基礎, IT Text, オーム社(2006).
7) 岡谷貴之: 深層学習, 機械学習プロフェッショナルシリーズ, 講談社(2015).
8) SAM 猫: About connecting the dots., 勾配ブースティングについてざっくりと説明する(オンライン), 入手先〈http://smrmkt.hatenablog.jp/entry/2015/04/28/210039〉（参照 2015-10-19）
9) The Apache Software Foundation:Apache Spark™ - Lightning-Fast Cluster Computing(オンライン), 入手先〈http://spark.apache.org〉(参照 2015-10-10)
10) 0xdata:H2O.ai - H2O(オンライン), 入手先〈http://h2o.ai/product/sparkling-water/〉(参照 2015-10-10)
11) Ruboss:Read Deep Learning Booklet | Leanpub(オンライン), 入手先〈https://leanpub.com/deeplearning/read〉(参照 2015-10-10)
12) UCI Machine Learning Repository: Credit Approval Data Set(オンライン), 入手先〈https://archive.ics.uci.edu/ml/datasets/Credit+Approval〉(参照 2015-10-10)
13) Takashi J OZAKI: tjo.hatenablog.samples/r_samples/public_lib/jp/exp_uci_datasets/card_approval at master · ozt-ca/tjo.hatenablog.samples · GitHub(オンライン), 入手先〈https://github.com/ozt-ca/tjo.hatenablog.samples/tree/master/r_samples/public_lib/jp/exp_uci_datasets/card_approval〉(参照 2015-10-10)
14) R: The R Project for Statistical Computing(オンライン), 入手先〈https://www.r-project.org〉(参照 2015-10-10)
15) Amazon.com, Inc.:Amazon EC2 (仮想クラウドサーバー) | アマゾン ウェブ サービス（AWS 日本語）(オンライン), 入手先〈https://aws.amazon.com/jp/ec2/〉(参照 2015-10-19)
株式会社インテリジェント ウェイブ(IWI.)(オンライン), 入手先〈http://www.iwi.co.jp〉(参照 2015-10-20)