高次元データ多クラス識別問題における GBDTライブラリの実装と改善

藤野 知之[1,a)]　千々和 大輝[1,b)]　税所 修[1,c)]　柏木 啓一郎[1,d)]

概要：機械学習技術の普及により，高度な学習処理を伴うデータの分類や回帰分析が様々な産業・サービスにおいて用いられている．機械学習の学習処理は計算量が大きい傾向にあり，クラウドコンピューティングを用い潤沢な計算資源の元実施されるのが一般的である．また，機械学習を用いたシステムを開発する際は入出力データやハイパーパラメータなどの試行錯誤が必要になり，繰り返し学習処理を行うことが多い．そのため，機械学習システムの開発コストの削減のため，学習処理自体の計算量を削減，効率化することが望ましい．本研究では勾配ブースティング法に着目し，高次元の入力データを分類する問題において学習処理の効率化手法を検討，実装し評価を行う．

1. 背景

GBDT(Gradient Boosting Decision Tree; 勾配ブースティング法) は深層学習と並んで広く普及している機械学習技術であり，様々な産業・用途で用いられている [5]．GBDT が広く利用されている理由として精度の高さがあり，様々な問題やコンペティションで state-of-the-art の精度を示している [2], [3]．もう一つの理由としてスケーラブルで大量のデータを扱えるライブラリの存在がある [3], [12]．これらのライブラリの登場によりクラウド上で膨大なデータに対しても学習・推論を行うことができるようになり，ビッグデータの解析手法として用いられている．

機械学習技術は多くのサービスで使われるようになっており，特に IoT システムにおいてユーザ端末上でセンサーデータを用いた機械学習の推論処理を行う Edge AI と呼ばれるコンピューティング形態が採用されるようになっている [15]．Edge AI の推論処理には TPU(Tensor processing unit) 等の数値計算に特化したハードウェアが用いられることもあるが，IoT ゲートウェイ製品では必ずしも専用のハードウェアを搭載していない場合も多く，限られたリソースで実行可能であることが望ましい．

GBDT は極めて推論処理が軽量であり広い用途で使うことが可能と考えられる．次章以降で詳細に述べるが GBDT は決定木 (Decision Tree) と呼ばれる 2 分探索木を複数構築し，推論時は複数の木に対して 2 分探索を実施し，探索結果の和を取ることで推論結果を得ることができる．2 分探索の時間計算量はノード数 T に対して $O(\log T)$ であるし，木数を M とすると推論処理の時間計算量は $O(M \log T)$ であり機械学習の処理としてはシンプルなものである．さらに，ツリーの探索は並列して実行することが可能であり，近年のプロセッサは IoT ゲートウェイのようなものでもマルチコアで動作しており，効率的に並列処理を実行できると考えられる．

一方で GBDT は学習処理において，入力データの次元数と分類するクラス数に対してそれぞれ比例して計算量が増加し，高次元データの分類問題において計算量が大きなものとなってしまうという課題がある．IoT システムにおいては多数のセンサデータや画像データなどを扱うことが考えられ，高次元の入力を扱うケースが多く存在し，実際のユースケースでは一般画像認識など分類するクラス数が多い分類問題も多く存在する．

一般にはこの課題はクラウドコンピューティングにより潤沢なリソースを用意して解決が図られる．入力次元数・識別クラス数に対してそれぞれ線形に計算量が増大するものの，その処理は並列実行可能であるため，学習処理するサーバーの数をスケールアウトすることで解決することができる．

クラウドにより大規模な学習を行う場合も，計算量の削減による効率化が有用な場合がある．例えば，システム設計，開発の際には，入出力やハイパーパラメータなどの調整を繰り返し行う必要があり，計算量を削減することで，

[1] NTT ソフトウェアイノベーションセンタ
NTT SIC, 3-9-11, Midori-cho, Musashino-shi, Tokyo 185–8585, Japan
[a)] tomoyuki.fujino.bn@hco.ntt.co.jp
[b)] daiki.chijiwa.mk@hco.ntt.co.jp
[c)] osamu.saisho.vm@hco.ntt.co.jp
[d)] keiichirou.kashiwagi.yk@hco.ntt.co.jp

開発工数の短縮やコンピューティングに必要なコストの低減に有用である．

我々は，高次元多クラスデータ分類を行うための学習・推論が軽量な GBDT ライブラリを目指し，実装を行った．実装のベースとなるアルゴリズムとして多クラスの分類問題を効率よく学習・推論可能な TFBT[16] を採用した．我々の実装では，TFBT をベースとして，高次元データを精度の劣化を最小限に抑えて次元削減を行う方法と，木のバランスを良くすることにより識別精度を高める方法を機能追加している．本稿では TFBT を導入したのち，それら二つの追加機能の提案と評価を行う．

次章で GBDT の関連研究を紹介し，3 章にて多クラス識別問題を効率よく分類する GBDT の手法 TFBT[16] を解説し，4 章で機能追加した二つの提案方法についてのべ，最後に我々の実装した GBDT ライブラリを用い提案方法の評価を行う．

2. 関連研究

Boosting は識別率の低い単純な弱識別器を組み合わせて強力な学習器を作るアンサンブル学習の一手法であり，Freund らによって提案された Adaboost[6] に端を発する．Friedman らは Adaboost が弱識別器を適切に決定することで目的関数の最小化を行なっていることを明らかにし [7]，Adaboost の目的関数よりも外れ値に強い Cross Entropy Loss を最小化する Logit Boost を提案した [8]．弱識別器として決定木を用い，目的関数にさらに正則化項を加えたものが GBDT である [4]．XGBoost[3] や LightGBM[12] といったライブラリは目的関数や正則化項を選択的に選べたり，外部から与えることができ，Logit Boost や Adaboost を包含する一般的な GBDT モデルと言える．

多クラス識別問題を Boosting によって解く方法についても様々な方法が提案されている．Adaboost ベースの Adaboost.MH[10] や前述の Logit Boost では One-vs-Rest 方式の多クラス識別方法を提案している．XGBoost や LightGBM など主要な GBDT ライブラリも One-vs-Rest 方式のみサポートしている．One-vs-Rest 方式は多クラス識別問題を，あるクラスに属するか否かの 2 値の識別問題に分解する方式で，1 回のイテレーションで各クラス毎に，属するか否か 2 値の決定木を構築する．本方式はシンプルで分かりやすい反面，イテレーション数×クラス数の木構築が必要となるため，クラス数が増えると計算量の面で効率が悪い上，木の数が多くなることによりモデルが複雑化し過学習が生じる危険性を持つ．

そこで，多クラスの問題を少ない木の数で解く方法として AOSO-LogitBoost[19]，TFBT[16] が提案されている．何も多クラス識別問題を直接的に解く方法であり，一つの木で全てのクラスの予測結果を出力するように設計されている．One-vs-Rest 方式に比較し少ない数の木で学習が可能である反面，過学習がおこりやすいと考えられる．本研究においてはよりより収束スピードが速く，少ない木の数で効率よく学習することのできる TFBT をベースとして用いる．

本研究では高次元の入力データを効率よく扱うために，入力を一部抽出したり変換したり，といった操作を加え，決定木毎に異なる入力を扱う手法を用いる．これは既存の Boosting 手法を一般化したものと言える．従来研究の中でこの手法を用いた最も成功した例として Viola-Jones Algorithm による顔認識技術がある [20]．Viola らは Haar-like 特徴量という画像の特徴量を抽出し，一つの特徴量に対して一つの弱識別器を構築する，という異なる入力に対するブースティングを用いている．また，時系列データに対して各時刻のデータを入力として弱識別器を構築する Earlyboost という方式も提案されている [9], [11].

3. TFBT による多クラス識別問題

本章では多クラスの問題を効率的に扱える TFBT のアルゴリズムを解説する [16].

N をサンプル数，D を入力の次元，C をラベルのクラス数とすると，学習データセットは $\{\boldsymbol{x}_i, y_i\}_{i=0}^{N}$ と表される，ただし, $\boldsymbol{x}_i \in \mathcal{R}^D$ は入力ベクトル，$y_i \in \mathcal{R}^C$ はラベルである．ブースティングでは以下の式のようにラベルの推定値 $\hat{y}_i$ を複数の弱識別器 $f_m(\boldsymbol{x}_i)$ の和で推定する．

$$\hat{y}_i = \sum_{m=1}^{M} f_m(\boldsymbol{x}_i) \tag{1}$$

ここで M は弱識別器の数である．TFBT のアルゴリズムの特徴はこの弱識別器を C 次元ベクトルを出力とする決定木で表すところである．すなわち $f_m(\boldsymbol{x}_i) : \mathcal{R}^D \to \mathcal{R}^C$ である．木構造の分岐は入力のうち一つの次元を指定し閾値を定める．入力の該当次元の大きさにより左側に分岐するか右側に分岐するかが決定する．葉ノードには重みベクトル $\boldsymbol{w}_t$ が設定され，ある入力に対する木の出力は分岐していった結果たどり着いた葉ノードの重みベクトルである．Gradient Boosting では $M-1$ までの弱識別器が与えられたとき目的関数を最小化するように M 個目の弱識別器をフィッティングさせる．目的関数は以下のような正則化項を含む関数が使用される．

$$L = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{m=1}^{M} \Omega(f_m) \tag{2}$$

$$\Omega(f_m) = \alpha T + \frac{1}{2}\lambda \sum_{t=1}^{T} \parallel \boldsymbol{w}_t \parallel^2 \tag{3}$$

ここで f_m は m 番目の木の構造を表しており，T は葉ノードの数を表す変数，α, λ は正則化パラメータである．l は多クラス識別問題では Cross entropy loss 関数が使われる [19]．(3) の第 1 項は葉の数に対する正則化項であり，木

構造が大きくなることに対して制約をかけている．第2項は重みに対する正則化項であり重みが大きくならないように制約をかけている．M 個目の木を構築する際，目的関数 (2) を以下のように和の形で記述した後，テーラー展開により 2 次の近似を行う．

$$L^{(M)} = \sum_{i=1}^{N} l(y_i, \hat{y}_i^{(M-1)} + f_M(\boldsymbol{x}_i)) + \sum_{m=1}^{M} \Omega(f_m) \quad (4)$$

$$\approx \sum_{i=1}^{N} \{l(y_i, \hat{y}_i^{(M-1)}) + f_M(\boldsymbol{x}_i)^{\mathsf{T}} \boldsymbol{g}_i + \frac{1}{2} f_M(\boldsymbol{x}_i)^{\mathsf{T}} H_i f_M(\boldsymbol{x}_i)\} + \sum_{m=1}^{M} \Omega(f_m) \quad (5)$$

ここで，$\boldsymbol{g}_i$ は $L^{(M-1)}$ の各クラスに対する予測値による偏微分値を持つ C 次元の勾配ベクトル，H_i は二階偏微分値を要素にもつ $C \times C$ 次元のヘッセ行列である．ここで木 f_M の分岐構造が与えられたとして，葉 t の持つ重みベクトルを $\boldsymbol{w}_t$ とすると，(5) の木 f_M による目的関数の変化分 $\tilde{L}^{(M)}$ を葉ごとの和の形で表すことができる．

$$\tilde{L}^{(M)} = \sum_{t}^{\mathsf{T}} \{\boldsymbol{w}_t^{\mathsf{T}} \bar{\boldsymbol{g}}_t + \frac{1}{2}(\boldsymbol{w}_t^{\mathsf{T}} \bar{H}_t \boldsymbol{w}_t + \lambda \boldsymbol{w}_t^{\mathsf{T}} \boldsymbol{w}_t)\} + \alpha T \quad (6)$$

ただし，葉 t に至るデータ点の集合を I_t とした時，$\bar{\boldsymbol{g}}_t = \sum_{i \in I_t} \boldsymbol{g}_i$，$\bar{H}_t = \sum_{i \in I_t} H_i$ である．

(6) の微分値が 0 となるような $\boldsymbol{w}_t$ を選択すると，

$$\boldsymbol{w}_t = -(\lambda E + \bar{H}_t)^{-1} \bar{\boldsymbol{g}}_t \quad (7)$$

を得る，ただし E は単位行列を表す．[19] によれば $\bar{H}_t$ は一般にはランクが $C-1$ であり，逆行列が一意に求まらないため，LogitBoost などの Boosting アルゴリズムではあるクラスを基底クラスとした One-vs-Rest 方式にすることで解の一意性を担保し，多クラス分類問題を解決している．TFBT では正則化項によりヘッセ行列のランクを確保することで，シンプルな多クラス分類アルゴリズムを実現している．

ここまでは木の構造が与えられたとき，重みベクトル $\boldsymbol{w}_t$ を決める問題であったが，ここからは木の分岐をどう決めるかを述べる．(7) を (6) に代入した値の符号を反転させた値を

$$\mathrm{Gain}_t = \frac{1}{2} \bar{\boldsymbol{g}}_t^{\mathsf{T}} (\lambda E + \bar{H}_t)^{-1} \bar{\boldsymbol{g}}_t \quad (8)$$

とするとこの値は，“適切な重みを当該ノードに割り当てたときの目的関数の減少量”を表し，本稿ではノードの利得値と呼ぶ．この値により，木の分岐の利得値を以下のように求めることができる．ノード P を分岐させ，ノード L とノード R を子ノードとした時の利得値は

$$\mathrm{Gain}_{P \to L,R} = \mathrm{Gain}_L + \mathrm{Gain}_R - \mathrm{Gain}_P - \alpha \quad (9)$$

で表される．TFBT を含む GBDT のアルゴリズムでは各ノードにおいて全ての入力次元において閾値を線形探索し，最も利得の高い“良い”分岐を決定する．

4. 提案方法：高次元多クラス分類 GBDT の改善

多クラス分類問題を効率よく解くために，我々は前章の TFBT のアルゴリズムを独自に実装した（以下 GBDT 実装と呼ぶ）．GBDT 実装では，オリジナルの TFBT アルゴリズムに，入力次元数の削減による計算量の増大を抑える仕組みと，木のバランスを取り過学習を抑制する仕組みを導入し，処理の効率化，高精度化を図った．本章では新たに導入した入力次元削減法，木のバランスを向上する正則化について順に述べる．

4.1 次元削減手法

従来では，画像データのような高次元の入力に対しては特徴量を抽出したり，影響の小さな次元を見つけ省いたり，事前に次元を削減しておくことで大規模なリソースを用いずに学習処理を行うのが一般的である．高次元な入力データには冗長な次元が存在したり，複数の次元間の相関により低次元な空間に射影可能なことが多く，非常に有効な手段と言える．しかしながら，低次元の特徴空間への射影は情報量を失う可能性のある処理であり，特徴量の選択次第では大きく精度を落としてしまう．

一方で，XGBoost や LightGBM には木ごと，あるいは木の深さごとに探索する次元をランダムに選択し，過学習を避ける機能が存在する．これは，各木への入力が一定である場合，勾配の近似計算結果が真値からずれてしまう，Prediction Shift と言う現象を予防するための機構である [17].

GBDT において木や分岐への入力空間は任意の写像により変換を行なうことが可能である．そこで GBDT 実装では，木への入力の次元をランダムに取捨選択するだけでなく，任意の変換関数 ϕ_m を設定可能とした．

$$\hat{y}_i = \sum_{m=1}^{M} f_m(\phi_m(\boldsymbol{x}_i)) \quad (10)$$

この変換関数を活用し，特徴量抽出を木毎・分岐毎に行い，次元数を減らすとともになるべく情報量を落とさないような二つの次元削減方法を提案する．

4.1.1 再帰的 PCA 法

一つ目は入力データについての事前情報がない場合の次元削減方法である．事前情報がない場合，データ間の相関関係の特定や不要な次元の選択はデータから推測する必要がある．ここでは，木への入力は全てのデータ点・入力次元を用い，分岐する際に削減する戦略をとる．分岐の探索は，ある分岐対象のノード t に含まれるデータ点を I_t とするときに I_t をある次元 d 上でソートし，可能な分割点を

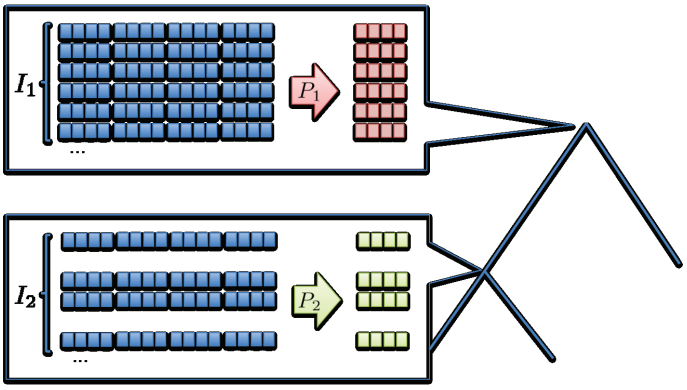

図 1 再帰的 PCA 法

線形に探索し，適切な閾値と次元 d を求める問題である．ここでは，PCA(Principal Component Analysis; 主成分分析) を用い，閾値を探索する方向を削減する．PCA は，次元間の相関を考慮し，入力空間を回転させてデータ点の分散が大きくなる方向を選択するアルゴリズムである．本方法ではこの PCA の方向選択に基づき最もデータの分散が大きくなる上位 D' 次元に削減する．これは入力空間を回転させ，分散が大きくなる斜めの方向を選択し分岐を探索することに相当する．PCA は以下で与えられる共分散行列の固有値分解により導かれる．

$$C_t = \frac{1}{|I_t| - 1} \sum_{i \in I_t} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}_t)(\boldsymbol{x}_i - \bar{\boldsymbol{x}}_t)^{\mathsf{T}} \tag{11}$$

$$C_t = V_t^{\mathsf{T}} D_t V_t \tag{12}$$

$$[P_t \bar{P}_t] = V_t \tag{13}$$

ここで対応する固有値の大きさ順に左から固有ベクトルを並べた際，左側 D' 列を射影行列 P，右側の $D - D'$ 列を $\bar{P}$ とした．射影行列により射影をした入力データ $P^{\mathsf{T}}\boldsymbol{x_i}$ の次元は D' に削減されており，それら対して分岐を探索を行う．本方式では PCA による次元削減はノードの分岐を探索する契機で行う．$|I_t|$ に含まれるデータ点は分岐毎に異なるため，射影行列 P_t により分岐毎に異なる座標変換が与えられる (図 1)．分岐探索の際，当該分岐に所属するデータ点集合に適した射影を得ることができるため，事前に全データに対する PCA を実施するのに対し情報落ちが少ないと考えられる．

全入力次元方向に探索した場合，分岐を探索するコストは $O(DNC + DNlogN)$ 程度である．一方 D' 次元に削減する場合，PCA のコストを鑑み $O(D^3 + D'NC + D'NlogN)$ となるが，PCA の処理は LAPACK[1]，ARPACK[14] といったライブラリにより高速に計算可能である．

4.1.2 スライディングウインドウ PCA 法

二つ目は画像データなど次元間の関係性が既知の場合の次元削減方法である．再帰的 PCA 法と同じく PCA を用いるが，PCA を行う対象となる次元をスライディングウインドウにより得る (図 2)．画像データの場合は 2 次元のウインドウを設定し，ウインドウサイズを変えながらスライドさせ，局所的な入力データの次元を抽出する．画像データ以外の場合は適切なウインドウを設定する必要がある，例えば時系列データの場合には時間軸方向のウインドウを用いることができる．さらに，抽出した次元に対して PCA による次元削減を行う．木への入力次元は木ごとに異なり，局所特徴を基に構築した木と大域的特徴を基に構築した木をアンサンブルすることができる．

4.2 分岐バランス正則化による過学習抑制

TFBT アルゴリズムにより繰り返し木構造の構築を行っていくと，一部のデータ点による利得への影響が強くなり，非常にバランスの悪い木構造が生まれてしまう場合がある (図 3)．これは，ある入力次元 d でのソート順で端に存在する外れ値の影響が強く出てしまうことによる．TFBT では各木が全てのクラスについて重みを割り当てることにより，One-vs-Rest 方式に比較して，特定のデータ点が大きな勾配を持ちやすくなっており利得への影響が強くなっていると考えられる．

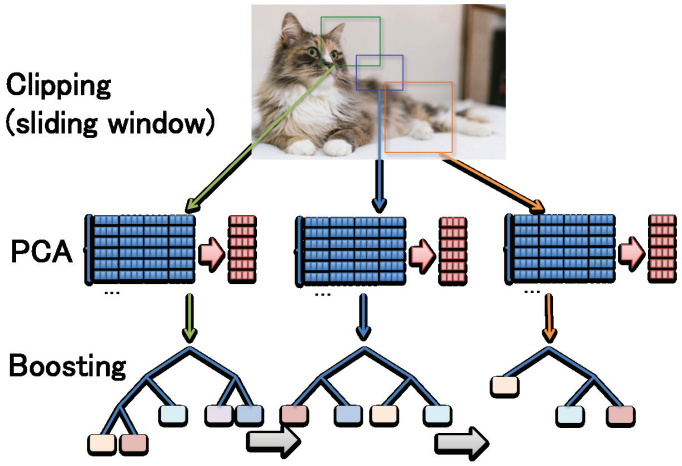

図 2 スライディングウインドウ PCA 法

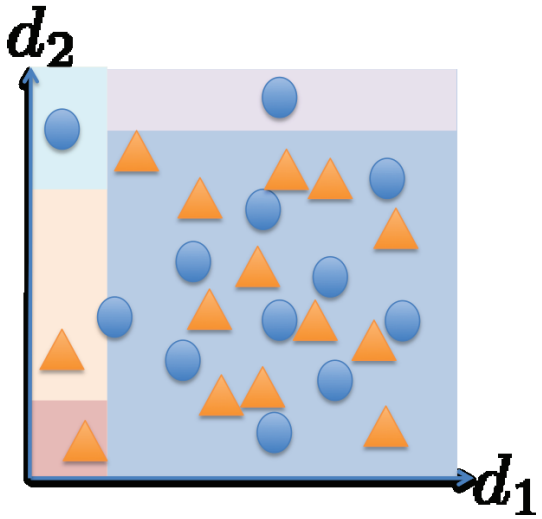

図 3 葉に所属するデータ点の数が偏ってしまう例

木のバランスは分岐するノードの選定と分岐位置の探索アルゴリズムに依存し，それぞれ改善方法を提案する．

4.2.1 Most-first アプローチによる分岐ノード選定

分岐ノードの選定においては，構築中の木 M の葉ノードのうち，次に分割するノードをどうやって決定するか，が課題となる．XGBoost では現在の木内で分岐していないノードのうち最も浅いノードで分岐を行う depth-wise 方式をとっている．depth-wise 方式は木構造が複雑になりすぎるのを抑制しているが，目的関数の収束性は後述する leaf-wise よりも低い．LightGBM では最も分岐後の利得の大きなノードを次の分岐の候補とする best-first の leaf-wise 方式を採用している [18]．leaf-wise 方式では利得を貪欲に増大化させていくため目的関数の収束は高速であるが，木構造が複雑化してしまう傾向があり過学習のリスクがある．

我々は学習データ点の木構造内での偏りに着目し，所属するデータ点数が最大のノードを分岐する most-first のアプローチを提案する．本提案方法は leaf-wise 方式で最も利得値 Gain_t の高いノードを分岐候補とするのに対し，t に含まれる学習データ点数 $|I_t|$ の大きいノードを分岐候補とする．本提案方法により，学習データ点数の多く含まれるノードの分割が優先され，前述のようなバランスの悪い木構造において，外れ値周辺に対して分岐が繰り返されることを抑止できる．

4.2.2 分岐バランス正則化

most-first アプローチにより巨大な葉を抑制することはできるが，外れ値周辺ではやはり所属するデータ点数が小さくなってしまい依然として課題が残る．LightGBM では最小データ点数のハイパーパラメータを設け，それ以下では分岐するのをやめるというアプローチをとっている．GBDT 実装にも同様のハイパーパラメータを設定可能としたが，その場合においてもハイパーパラメータを小さくすると効果が薄く，大きくすると分岐の余地がなくなってしまい逆に性能が低下してしまうという問題がある．

我々は本問題を解決する新たな正則化手法を提案する．分岐を探索する際の利得値 (9) に着目する．分岐の際には全ての方向で閾値を探索するが左右に所属するデータ点の数についての制約はない．そこで，偏った分岐に対してペナルティを与えるような正則化を行う．正則化パラメータ γ を新たに設け，以下のような正則化を提案する．

$$\mathrm{Gain}_{P\to L,R} = \beta(\mathrm{Gain}_L + \mathrm{Gain}_R) - \mathrm{Gain}_P - \alpha \quad (14)$$

$$\beta = 1 - \gamma\frac{(|I_R| - |I_L|)^2}{|I_P|^2} \quad (15)$$

(15) 式が形作る (図 4) のような形状の係数を分割利得にかけることで，偏りのある分岐に対してはペナルティが課されることになる．これにより，バランスの良い分岐が選択されやすくなる．

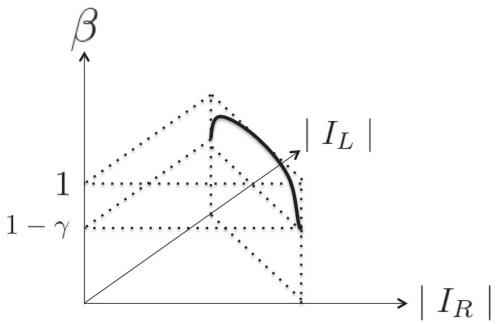

図 4 分岐時データ点数をバランスさせる正則化係数のイメージ

5. 有効性検証

検証には TFBT アルゴリズムをベースにした GBDT 実装を用いる．実装には C++の数値計算ライブラリ Eigen を使用し，PCA には ARPACK，並列処理に OpenMP を用いた．

GBDT 実装の特徴として，木への入力を木毎・分岐毎に変化させることができる点がある．これにより，再帰的 PCA 法・スライディングウィンドウ PCA 法を実装している．

デフォルトでは leaf-wise 方式の分岐選択を行うが，オプションとして most-first な分岐選択を可能にした．また，分岐バランス正則化パラメータ γ を設定可能とし，デフォルトでは 0(正則化なし) とした．

ベンチマークのデータセットとして CIFAR-10 データセットを用いる [13]．CIFAR-10 データセットは 32x32 の RGB 画像データであり，入力次元は 3072 次元，クラス数 10 クラス，データ数は学習用に 50000，テスト用に 10000 である．参考値として，2000 回 iteration を行った（木数 20000）XGBoost の学習器による error rate は 40.69%であった．XGBoost は L2 正則化パラメータなどのハイパーパラメータをグリッドサーチにより最適化したものを用いた．

5.1 実験 1: 次元削減による計算量低減効果の検証

次元削減による計算量の低減効果を検証するため，GBDT 実装による CIFAR-10 データセットの平均木構築時間を測定した結果を (表 1) に示す．

次元削減方法	木構築時間 [秒]
なし	1811.8
再帰的 PCA	107.2
スライディングウインドウ PCA	19.3

表 1 CIFAR-10 平均木構築時間

次元削減を行わない場合，3072 次元の分岐探索を都度行う必要があり，処理に非常に時間がかかっていることがわかる．一方，次元削減を行なった場合においては木構造の

探索次元が減っているため，再帰的 PCA 法・スライディングウインドウ PCA 法いずれも計算量の削減が行われており，計算時間が短縮されていることが確認できた．再帰的 PCA 法において，計算時間がスライディングウインドウ PCA と比較して大きくなっているが，これは前述のように，分岐毎に PCA を計算する処理に起因していると考えられる．

5.2 実験 2: 再帰的 PCA 法の精度検証

CIFAR-10 データセットを事前に PCA で 64, 128, 256 次元までそれぞれ削減したデータセットを GBDT 実装に入力した場合と，事前処理なしで再帰的 PCA 法を適用した場合とを比較する．ただし，再帰的 PCA 法の各分岐で削減した後の次元数は 64 とした．それぞれ 500 個の木を構築し，テストデータの error rate を評価した (図 5)．

5.3 実験 3: スライディングウインドウ PCA 法の精度検証

CIFAR-10 データセットに対して，(表 2) の設定でスライディングウインドウ PCA 法を行い，約 1500 の木を構築し，テストデータの error rate を評価した (図 6)．なお boosting を行う際の順序として，ウインドウサイズの大きいものから順とし，同じウインドウサイズでは左から右，上から下方向にスライドさせた順序とした．

ウインドウサイズ	32x32	16x16	8x8	4x4
ストライド	0	4	4	2
入力パターン数	1	25	49	225
削減後次元数	96	48	24	12
木数/パターン	50	8	8	4
木数計	50	200	392	900

表 2 スライディングウインドウ設定

5.4 実験 3: 分岐バランス正則化の精度検証

実験 2：再帰的 PCA 法および実験 3：スライディングウィンドウ PCA 法のそれぞれと同様の実験に分岐バランス正則化を施した．ハイパーパラメータの γ は $[0.01, 0.05, 0.1, 0.2, 0.5]$ で 5 分割のクロスバリデーションによる探索を行い，0.2 を採用した．(図 5)，(図 6) に分岐バランスの正則化を用いた際の error rate も合わせてプロットした．

5.5 考察

実験 1 より次元削減は学習時の計算量の低減に効果的であることが確認されたが，実験 2 で示されたように単純に事前処理として GBDT への入力を削減すると，識別精度が落ちてしまうというトレードオフ関係を見て取ることができた．

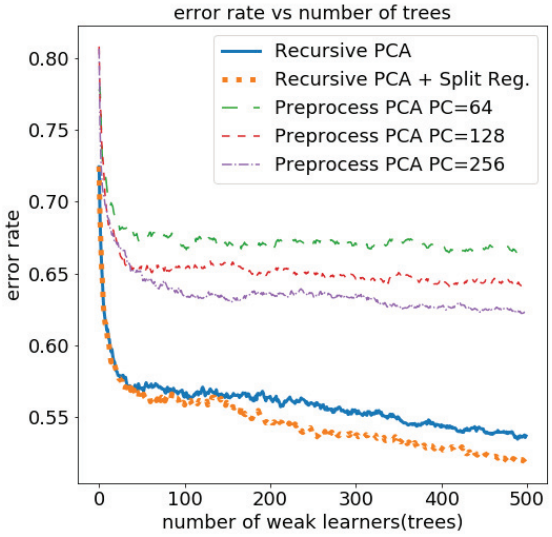

図 5 再帰的 PCA 法と事前に PCA により次元削減した場合の error rate 比較

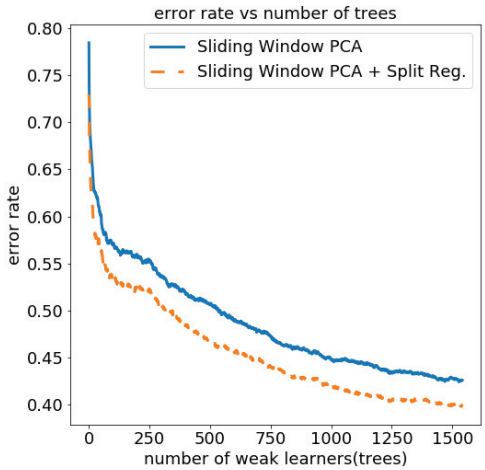

図 6 スライディングウインドウ PCA 法の結果と分岐バランス正則化を加えた場合の error rate

このトレードオフを緩和する方法として，再帰的 PCA 法とスライディングウインドウ PCA 法がいずれも効果があることが実験 2・3 より示された．特に，スライディングウインドウ PCA 法は画像などの一部のデータにしか用いることができないが，計算量の削減と精度を両立可能であり，分岐正則化と組み合わせることで XGBoost に勝る精度を 1500 木程度で実現することができた．

また，識別時の計算時間は本稿では実測していないが，木の数と比例関係にあるため，XGBoost と比較して一桁程度の計算量削減効果が見込める．

最後に参考値として，gcForest[21] との比較について述べる．gcForest は CIFAR-10 データの識別精度に関してツリーアンサンブル手法として高い精度を上げており，非

ニューラルネットの学習アルゴリズムとしては state-of-the-art のアルゴリズムと言える．gcForest は多段に連なった深層学習に似た複雑なアンサンブル手法をとっており，学習に必要な計算量は一般に Boosting 手法に比較しても高いが，その分高い精度を出すことができるという特徴を持つ．(表 3) に本研究の方式と gcForest との比較を示す．およそ gcForest に迫る精度が達成されていることがわかる．

アンサンブル法	CIFAR-10 error rate
gcForest	38.22% [21]
gcForest + GBDT	31.00% [21]
XGBoost	40.69 %
スライディングウインドウ PCA 法	42.62%
+分岐正則化	39.82%

表 3 CIFAR-10 分類精度の比較

また，分岐バランス正則化は再帰的 PCA 法・スライディングウィンドウ PCA 法いずれも精度向上に寄与することを示した．他の GBDT アルゴリズムに対しても同様の正則化を与えることは可能なので，効果のある適用範囲を今後探りたい．

6. 今後の展望と課題

GBDT は木毎，あるいは分岐毎に入力変数を変えることで計算量を削減したり精度を向上させることができることを示した．現在の多くの GBDT ライブラリは柔軟に木毎に入力を変換するといった機能が実装されておらず，我々の GBDT 実装により柔軟な GBDT の設計が可能になる．この特性を有効に使うことのできるユースケースを検討したい．

本稿執筆時点では我々の GBDT 実装は LightGBM で使われているヒストグラム方式などの高速化方式が未実装であり，今後それらの効果も実装し検証していく必要がある．

精度に関して，XGBoost に対抗し得る精度を出すことに成功したが，gcForest と GBDT を組み合わせた場合の精度には劣る．今後のさらなる検討が必要である．

また，木のバランスが識別精度に影響し，バランスを取ることにより精度が向上することが特定データセットについて示されたが，新たな正則化手法であるため，様々なデータセットにより慎重に有効性を検証する必要があると考えられる．

上述のような課題について，今回の CIFAR-10 データセットのみならず実問題解決を含む様々なデータセットに対して検討・評価していきたい．

参考文献

[1] E Anderson, Z Bai, C Bischof, J Demmel, J Dongarra, J DuCroz, A Greenbaum, S Hammarling, A McKenney, and D Sorensen. Lapack: A portable linear algebra library for high-performance computers. 1990.

[2] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. *Proceedings of the 23rd international conference on Machine learning - ICML '06*, Vol. 2006, pp. 161–168, 06 2006.

[3] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. ACM.

[4] Corinna Cortes, Mehryar Mohri, and Dmitry Storcheus. Regularized gradient boosting. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pp. 5449–5458. Curran Associates, Inc., 2019.

[5] A. Ferreira and M. Figueiredo. Boosting algorithms: A review of methods, theory, and applications. *Ensemble Machine Learning: Methods and Applications*, Vol. 3, pp. 35–85, 01 2012.

[6] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In Paul Vitányi, editor, *Computational Learning Theory*, pp. 23–37, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[7] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, Vol. 29, No. 5, pp. 1189–1232, 2001.

[8] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, Vol. 28, No. 2, pp. 337–407, 2000.

[9] Tomoyuki Fujino, Katsuhiko Ishiguro, and Hiroshi Sawada. Logitboost extension for early classification of sequences. In *International Conference on Computer Analysis of Images and Patterns*, pp. 579–588. Springer, 2011.

[10] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, Vol. 2, No. 3, pp. 349–360, 2009.

[11] Katsuhiko Ishiguro, Hiroshi Sawada, and Hitoshi Sakano. Multi-class boosting for early classification of sequences. In *BMVC*, pp. 1–10, 2010.

[12] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pp. 3146–3154. Curran Associates, Inc., 2017.

[13] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[14] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

[15] E. Li, L. Zeng, Z. Zhou, and X. Chen. Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, Vol. 19, No. 1, pp. 447–457, 2020.

[16] Natalia Ponomareva, Thomas Colthurst, Gilbert Hendry, Salem Haykal, and Soroush Radpour. Compact multi-class boosted trees. *2017 IEEE International Conference on Big Data (Big Data)*, pp. 47–56, 2017.

[17] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In *Advances in neural information processing systems*, pp. 6638–6648, 2018.

[18] Haijian Shi. *Best-first decision tree learning*. PhD thesis, The University of Waikato, 2007.

[19] Peng Sun, Mark D Reid, and Jie Zhou. Aoso-logitboost: Adaptive one-vs-one logitboost for multi-class problem. *arXiv preprint arXiv:1110.3907*, 2011.

[20] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, Vol. 1, pp. I–I. IEEE, 2001.

[21] Zhi-Hua Zhou and Ji Feng. Deep forest. *arXiv preprint arXiv:1702.08835*, 2017.