不均衡データ分類フレームワークにおける サンプリング比率の最適化

植原　リサ† 　駒水　孝裕† 　小川　泰弘† 　外山　勝彦†

† 名古屋大学　〒 464–8603 愛知県名古屋市千種区不老町

E-mail: uehara@kl.i.is.nagoya-u.ac.jp, taka-coma@acm.org, {yasuhiro,toyama}@is.nagoya-u.ac.jp

あらまし　分類器を構築する上で，ラベルの不均衡性は分類性能を低下させる要因の一つである．不均衡性を持つデータを用いて学習した分類器は，多数派ラベルを出力しやすい．例えば，すべてのデータを多数派ラベルであると予測すると，全体の正解率は高い値となるが，少数派の正解率は 0%となる．しかし，実際には少数派ラベルの方が重要となるアプリケーションが多く存在する．そのため，多数派ラベルの分類性能を落とさずに，少数派ラベルを性能良く分類する必要がある．これに対し，先行研究では，複数比率のアンダーサンプリングを用いた分類フレームワークが提案され，従来手法に比べて良い分類性能を示した．この先行研究では，フレームワーク中の弱分類器数制御関数について 3 種類のナイーブな関数を設計している．これらのナイーブな関数は，データごとの分布や不均衡度合の違いに対処できず，分類フレームワークの性能を十分に引き出すことができない．そこで本稿では，データに対しより柔軟に対応できるように，ガウス関数を弱分類器数制御関数として導入する．加えて，グリッドサーチを用いたパラメータ最適化を行うことで，事前に設計する必要のあるパラメータ数を削減する．公開されている 19 個のデータセットを用いた実験の結果，ガウス関数を用いた弱分類器数の制御が分類フレームワークの性能を向上させることを確認した．さらに，パラメータ最適化により，理想的なパラメータを選択した場合と近い性能を発揮できることを確認した．

キーワード　分類，機械学習，アンサンブル，アンダーサンプリング，不均衡データ

1　はじめに

分類器を構築する上で，ラベルの不均衡性は分類性能を低下させる要因の一つである．不均衡性を持つデータを用いて学習した分類器は，多数派ラベルを出力しやすい．例えば，多数派データの数が 990 個，少数派データの数が 10 個の場合，すべてのデータを多数派ラベルであると予測すると，全体の正解率は 99% となるが，少数派の正解率は 0%となる．しかし，実際には少数派ラベルの方が重要となるアプリケーションが多く存在する．その例として，クレジットカードの債務不履行予測が挙げられる．クレジットカードの債務者の中で，債務不履行に陥る人は極めて少ない．しかし，クレジットカード会社にとって，債務不履行による損害は避けたいことの一つである．債務者の中から債務不履行に陥る可能性が高い人を予測することで，そのリスクを回避することができる．そのため，多数派ラベルの分類性能を落とさずに，少数派ラベルを性能良く分類する必要がある．

不均衡データの分類に対する有効な方法の一つに，リサンプリングがある．リサンプリングは，アンダーサンプリングとオーバーサンプリングに大別される．アンダーサンプリングでは多数派のデータを少数派の数に合わせてサンプリングすることによって，また，オーバーサンプリングでは少数派のデータを増やすことによって不均衡性を解消する．アンダーサンプリングは，学習に用いるデータ数を減らしているため学習コストが小さく，多くの場面で利用されている．しかし，アンダーサンプリングには多くの多数派データが学習に使われないという問題がある．これに対し，多数派データを活用するため，複数回のアンダーサンプリングを組合せたアンサンブル手法が提案された [1]．この手法では，それぞれでサンプリングしたデータに対してアンサンブル手法を用いて弱分類器を学習し，それらの弱分類器をアンサンブルして最終的な分類器を学習する．このような従来手法では，一つのサンプリング比率を用いている．しかし，データセットごとに適切なサンプリング比率は異なるため，その比率を事前に決めることは容易ではない．

この問題に対し，先行研究で複数比率のアンダーサンプリングを用いた分類フレームワークを提案した．このフレームワークでは，異なるサンプリング比率に対して弱分類器を学習し，それらをアンサンブルすることで分類を行う．この先行研究では，フレームワーク中の弱分類器数制御関数について 3 種類のナイーブな関数を設計している．しかし，これらのナイーブな関数は，データごとの分布や不均衡度合の違いに対処できず，分類フレームワークの性能を十分に引き出すことができない．

そこで本稿では，データに対しより柔軟に対応できるように，ガウス関数を弱分類器数制御関数として導入する．しかし，実利用上の観点から，パラメータ数が多いことは望ましくない．また，ガウス関数のパラメータ設定は，設計するためにデータに対する深い理解を必要とする．そのため，手動でのパラメータ設定は現実的ではない．そこで，グリッドサーチを用いたパラメータ最適化を行うことで，事前に設計する必要のあるパラメータ数を削減する．

提案手法の有効性を検証するために，公開されている 19 個

のデータセットを用いて実験を行った．その結果，ガウス関数を用いた弱分類器数の制御が分類フレームワークの性能を向上させることを確認した．さらに，パラメータ最適化により，理想的なパラメータを選択した場合と近い性能を発揮できることを確認した．

本稿の構成は次の通りである．2 節では関連研究について述べる．3 節では複数比率のアンダーサンプリングを用いるための分類フレームワークについて説明する．4 節では提案手法について述べる．5 節では評価実験の結果を報告する．6 節では本稿のまとめと今後の課題について述べる．

2 関 連 研 究

不均衡データの分類に対する従来手法には，リサンプリングに関する手法がいくつか存在する．単純なオーバーサンプリングでは，少数派データを複製することによって不均衡性を解消するが，過学習を起こしやすい．この問題に対して，k-NN を用いて少数派データを擬似的に生成する SMOTE [2] や，分類困難な少数派データに焦点を当ててデータを生成する ADASYN [3] が提案された．また，多数派データと少数派データの分布は似ているという仮説から，多数派データの分布を基に少数派データを生成する SWIM [4] が提案されている．

加えて，SMOTETomek [5] や SMOTEENN [6] など，オーバーサンプリングとアンダーサンプリングを組合せた手法もいくつか存在する．このような手法では，オーバーサンプリングによって少数派データを増やした後に，分類をする上で不要なデータを除去することで，クラス間の境界をより明確にしている．SMOTETomek では Tomek link [7] を用いて，SMOTEENN では Edited Nearest Neighbours [8] を用いてデータの除去を行う．

一方で，単純なアンダーサンプリングでは多くの多数派データが学習に使われず，有用なデータを失っている．多数派データを活用するために，アンダーサンプリングとアンサンブル手法を組合せた EasyEnsemble [1] や RUSBoost [9] が提案された．EasyEnsemble は，複数回のアンダーサンプリングをした後に，それぞれでサンプリングしたデータ AdaBoost を適用し，学習した弱分類器のすべてをアンサンブルする手法である．AdaBoost [10] とはアンサンブル学習の一つであり，RUSBoost は AdaBoost のフレームワークにアンダーサンプリングを取り入れた手法である．

いずれの手法もアンダーサンプリングに用いる比率は一定であり，一般的に 1 とする場合が多い．しかし，データセットごとに適切なサンプリング比率は異なるため，その比率を事前に決めることは容易ではない．加えて，適切なサンプリング比率が一つの値であるとは限らない．これに対し，先行研究 [11] および提案手法では，複数のサンプリング比率を同時に利用する分類フレームワークを提案した．

3 フレームワーク

本節では，先行研究で提案した分類フレームワークについて説明する．

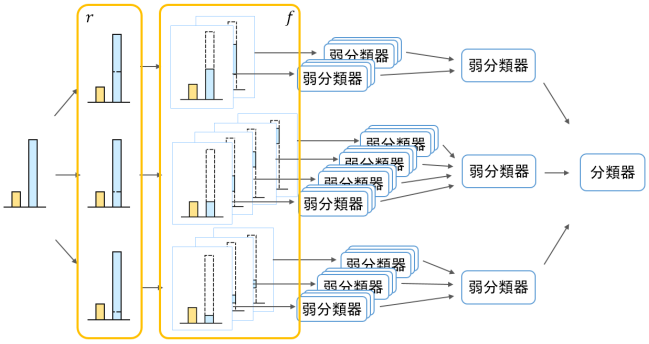

図 1: 分類フレームワークの全体図

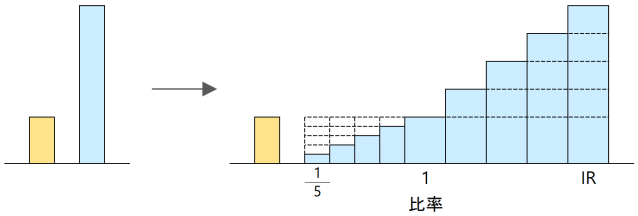

図 2: 少数派の数とサンプリングする多数派の数

このフレームワークでは，複数の比率を用いてアンダーサンプリングを行い，その比率ごとに弱分類器を学習し，それらをアンサンブルすることで最終的な分類器を構築する．分類フレームワークの全体図を図 1 に示す．このフレームワークで，関数 r と f によって， サンプリング比率と，比率ごとに学習する弱分類器の数を制御する．これらの関数は，データの性質に応じた関数を自由に設計できる．このフレームワークでは，r と f のそれぞれについて，ナイーブな関数を提供している．

3.1 比率制御関数 r

最初に，学習データから比率制御関数 r によって用いるサンプリング比率を決定する．決定したサンプリング比率それぞれに対して複数回のサンプリングを行い，弱分類器を学習する．関数 r は，サンプリングする回数を表す c と用いるサンプリング比率の数 n によって値が決まり，0 以上の実数値を返す．

一般に，アンダーサンプリングに用いる比率は 1 以上とすることが多いが，1 より小さい比率を取ることによって少数派に寄せた分類器を学習できる．そのため，1 より小さいサンプリング比率も返すように関数 r を設計する．また，扱うデータセットによって不均衡率（少数派データの総数に対する多数派データの総数の割合）が異なるため，データセットごとに適切なサンプリング比率を用いる．そこで，$n = 2m - 1$ とし，m の値と不均衡率 IR から 1 以上のサンプリング比率を決定する．図 2 は，$m = 4\,(n = 7)$ のときの少数派の数とサンプリングする多数派の数を図にしたものである．オレンジの四角は少数派データ，水色の四角は多数派データを表す．$c = m = 4$ のときにサンプリング比率が 1 になるように設計し，大小それぞれ等間隔で比率を変化させる．このときの関数 r は，c, m, IR を用いて，式 (1) のように定義する．

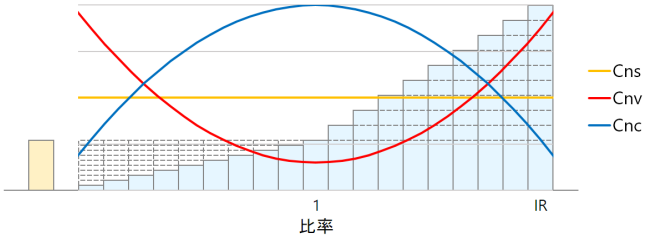

図 3: 先行研究における弱分類器数制御関数 f のグラフ

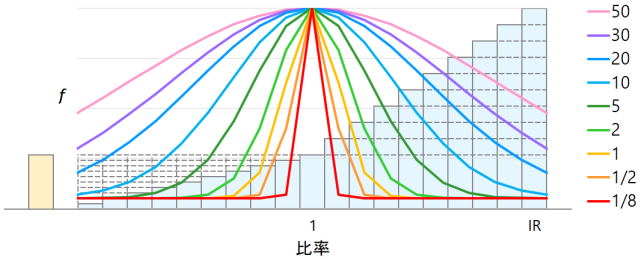

図 4: 提案手法における弱分類器数制御関数 f のグラフ ($\mu = 10$)

$$r(c, m, IR) = \begin{cases} \frac{1}{m} \cdot c & (c \leqq m)\,, \\ 1 + \frac{IR-1}{m-1} \cdot (c - m) & (otherwise) \end{cases} \quad (1)$$

この式において，比率を最小 $\frac{1}{m}$ から最大 IR まで，計 $2m - 1$ 回変化させる．

3.2 弱分類器制御関数 f

学習時にどのサンプリング比率を重視するのかについては，弱分類器の数によって調整する．弱分類器数制御関数 f から比率ごとで行うサンプリング回数を調整することで，学習する弱分類器の数を調整することができる．関数 f は，サンプリングする回数を表す c と用いるサンプリング比率の数 n によって値が決まり，0 以上の整数値を返す．

一般に，サンプリング比率が 1 から遠くなるほど，学習される弱分類器は多数派または少数派に寄る．そのため，多数派や少数派に寄せた弱分類器を重視するかどうかという観点から，弱分類器の数を制御する関数 f を設計する．このときの関数 f を以下の式 (2)〜式 (4) に示す．

$$\textbf{Cns} : f(c, m) = b_s\,, \quad (2)$$

$$\textbf{Cnv} : f(c, m) = \left\lfloor a_v (c - m)^2 + b_v \right\rfloor, \quad (3)$$

$$\textbf{Cnc} : f(c, m) = \left\lfloor -a_c (c - m)^2 + b_c \right\rfloor, \quad (4)$$

ここで，a_v, a_c, b_s, b_v, b_c は定数であり，a_v, a_c は正の値をとる．a_v, a_c では弱分類器数のとりうる範囲を，b_s, b_v では最小，b_c では最大となる弱分類器の数を調整する．また，式 (3)，式 (4) の $\lfloor\,\rfloor$ はフロア関数であり，整数値を返すように調整している．この三つの関数のグラフを図 3 に示す．オレンジの四角は少数派データ，水色の四角はサンプリングする多数派データの数を表す．ここでは，$m = 10$, $a_v = a_c = \frac{1}{5}$, $b_s = 10$, $b_v = 3$, $b_c = 20$ としており，$c = m = 10$ のときにサンプリング比率が 1 となる．Cns（式 (2)）は，各サンプリング比率に対して弱分類器の数が同じになることを表し，Cnv（式 (3)）は，比率が 1 から遠くなるほど弱分類器の数を多く，また，Cnc（式 (4)）は，比率が 1 に近づくほど弱分類器の数が多くなることを表している．

3.3 先行研究の問題点

先行研究では，二次関数を用いて関数 f を設計した．二次関数は，係数でコントロールできる傾斜のなだらかさに限界がある．そのため，サンプリング比率ごとの弱分類器数を柔軟に調整することができない．

また，先行研究では，関数 f をサンプリング比率 1 の近くを重視するか，遠くを重視するかという観点で関数を設計している．実験により，サンプリング比率 1 の近くを重視する関数 (Cnc) で高い分類性能となることが示されたが，最重視するサンプリング比率はどのデータセットに対しても 1 であり，固定の値となっている．しかし，データセットごとに適切なサンプリング比率は異なるため，最重視する比率を固定することは最良であるとはいえない．

4 提 案 手 法

本節では，弱分類器数制御関数 f について，従来の関数に替わる新たな関数の設計を述べる．より柔軟な関数を導入し，最重視するサンプリング比率を調整するパラメータを導入する．

4.1 関数 f の設計

本稿では，前述の先行研究の問題に対し，データセットに応じて適切なサンプリング比率を重視できるように，最重視する比率を調整するパラメータを導入する．さらに，先行研究では二次関数を用いて設計していたが，弱分類器の数をより柔軟に加減できるようにガウス関数を用いる．同時に，弱分類器の数の加減を調整するパラメータを導入する．

今回設計した関数を式 (5) に示す．

$$\textbf{Gauss} : f(c, m) = \left\lfloor a_g \cdot \exp\left(-\frac{(c - \mu)^2}{2\sigma^2}\right) + b_g \right\rfloor \quad (5)$$

ここで，a_g, b_g は定数であり，a_g では弱分類器数のとりうる範囲を，b_g では最小となる弱分類器数を調整する．また，式中の $\lfloor\,\rfloor$ はフロア関数であり，整数値を返すように調整している．μ, σ^2 はそれぞれ，最重視するサンプリング比率，比率ごとの弱分類器数の加減を調整するパラメータである．$m = 10$, $\mu = 10$, $\sigma^2 = \{\frac{1}{8}, \frac{1}{2}, 1, 2, 5, 10, 20, 30, 50\}$ としたときの関数 f のグラフを図 4 に示す．$\mu = m = 10$ とすることでサンプリング比率 1 を最重視し，σ^2 の値を大きくすることで比率ごとの弱分類器数の加減を小さくすることができる．σ^2 の値により，先行研究の Cnc や Cns に近い値をとることができる．μ の値を小さくすることで，小さいサンプリング比率を最重視することができる．

4.2 パラメータの組合せ

設計した関数には，二つのパラメータ μ, σ^2 が存在し，事前に値を決める必要がある．しかし，データセットによって適切なパラメータの値は異なる．図 5 は，パラメータ μ, σ^2 を変

パラメータ		σ^2								
		1/8	1/2	1	2	5	10	20	30	50
μ	2	.402	.314	.284	.276	.297	.321	.391	.422	.490
	4	.550	.532	.525	.472	.436	.428	.459	.487	.532
	6	.613	.620	.614	.603	.581	.548	.541	.564	.585
	8	.653	.651	.660	.654	.658	.646	.635	.636	.635
	10	.676	.679	.695	.694	.695	.694	.694	.689	.673
	12	.684	.670	.668	.668	.660	.688	.699	.701	.695
	14	.680	.638	.607	.572	.562	.585	.637	.669	.698
	16	.678	.644	.576	.544	.519	.517	.564	.616	.683
	18	.671	.586	.544	.530	.503	.496	.514	.559	.645

(a) データセット D9 の場合

パラメータ		σ^2								
		1/8	1/2	1	2	5	10	20	30	50
μ	2	.598	.532	.485	.516	.530	.569	.636	.657	.699
	4	.698	.690	.668	.659	.631	.653	.672	.703	.714
	6	.706	.722	.713	.696	.709	.695	.699	.710	.710
	8	.735	.717	.708	.707	.707	.700	.706	.708	.689
	10	.708	.676	.674	.674	.658	.684	.669	.671	.682
	12	.593	.560	.547	.553	.562	.577	.586	.624	.627
	14	.538	.486	.421	.450	.408	.421	.514	.572	.614
	16	.566	.467	.440	.428	.366	.409	.442	.464	.534
	18	.488	.424	.416	.404	.386	.389	.399	.426	.489

(b) データセット D10 の場合

図 5: パラメータを変化させたときの G-mean

化させたときの多数派の正解率と少数派の正解率の幾何平均 G-mean をヒートマップで表した図である．背景がピンクになるほど高い値，青になるほど低い値を示している．図 5 (a) は実験で用いたデータセット D9 の結果を，図 5 (b) は D10 の結果を表している．これより，D9 では $\mu = 10$ で，D10 では $\mu = 6$ で高い値をとることが分かる．同様に，最も高い値をとる σ^2 も，データセットごとに異なる．また，これらの値を適切に決めるには，データについての深い理解が必要とされるが，理解があったとしても適切な設定は困難である．そのため，高い値をとるようなパラメータ μ, σ^2 の値を事前に決めることは容易ではない．

4.3 パラメータ最適化

データセットごとに適切なパラメータの値を決定できるように，グリッドサーチを用いてパラメータの最適化を行う．グリッドサーチとは，すべてのパラメータの組合せに対して実験を行い，最も性能の良い組合せを探索する手法である．学習データのみを用いてグリッドサーチを行い，k 分割交差検証によって最適なパラメータの組合せを探索する．

5 実　　験

提案手法の有効性を確認するために公開されているデータを用いた実験を行った．本実験では，次の二点について明らかにする．まず，グリッドサーチによる最適化により最適なパラメータが選択できているかどうか，次に，提案手法は不均衡データ分類に対して有効な手法かどうかである．以下では，まず，実験に用いたデータセットと評価指標について説明する．その後に，上記二点に対応する実験の内容および結果について順に説明する．

5.1 データセット

データセットとして表 1 に示す 19 種類を用いた．表中の D1～D10 は UCI 機械学習リポジトリ [12] から取得したものであり，D11～D19 は Kaggle[1]から取得したものである．なお，今回は二値分類を行うため，データセットが多クラスの場合にはクラス数を 2 となるようにした．データセット名のカッコ内は，それぞれ多数派，少数派クラスとしたクラス名を表し，IR は不均衡率を表す．実験に用いたデータセットは，次元数，データ数，不均衡率の点で多様である．そのため，提案手法のデータに対する得手・不得手を議論できることが期待される．学習データとテストデータの割合は 1:1 とし，多数派と少数派の割合が一定となるように分割した．

表 1: データセットの詳細

ID	データセット名	次元	データ数		IR
			多数派	少数派	
D1	Abalone (9, 18)	8	689	42	16.4
D2	Anuran Calls (Le., Bu.)	22	4,420	68	65.0
D3	Covertype (2, 5)	54	283,301	9,493	29.8
D4	default of credit card	23	23,364	6,636	3.5
D5	HTRU2	8	16,259	1,639	9.9
D6	Online Shopper	18	10,422	1,908	5.5
D7	Polish bankruptcy	64	5,500	410	13.4
D8	Spambase	56	2,788	1,813	1.5
D9	Wine Red ((3,4), others)	11	1,536	63	24.4
D10	Wine White (7, 3)	11	880	20	44.0
D11	Churn Modelling	9	7,963	2,037	3.9
D12	Credit Card Fraud	30	284,315	492	577.9
D13	ECG HB - Arr. (N, F)	187	90,589	803	112.8
D14	Financial Distress	85	3,536	136	26.0
D15	LoanDefault LTFS AV	39	182,543	50,611	3.6
D16	Mafalda Opel - DS	14	9,530	2,190	4.4
D17	Mafalda Peugeot - DS	14	12,559	678	18.5
D18	Rain in Australia	20	110,316	31,877	3.5
D19	Surgical	24	10,945	3,690	3.0

5.2 評 価 指 標

不均衡データに対して，多数派，少数派共に性能良く分類する必要がある．そのため，True Negative Rate（多数派クラスの正解率）と True Positive Rate（少数派クラスの正解率）の幾何平均 G-mean [13] を評価指標とした．

1 : https://www.kaggle.com/datasets

表 2: 実験 1 の結果

ID	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16	D17	D18	D19
最適化結果	.732	.958	.807	.701	.934	.848	.908	.917	.694	.735	.762	.938	.900	.860	.593	.713	.791	.765	.798
最良結果	.772	.971	.809	.701	.935	.849	.910	.919	.701	.735	.762	.939	.900	.865	.593	.713	.791	.765	.803
差	.040	.013	**.002**	**.000**	**.001**	**.001**	**.002**	**.002**	.007	**.000**	**.000**	**.001**	**.000**	**.005**	**.000**	**.000**	**.000**	**.000**	**.005**

5.3 実験 1： 最適化の適切性

グリッドサーチによるパラメータ最適化が適切であるかどうかを検証するため，比較実験を行った．

探索するパラメータの組合せすべてに対して学習データを用いて 3 分割交差検証を行い，10 回の操作のマクロ平均が最も高いパラメータの組合せを最適なものとした．探索する各パラメータはそれぞれ次の 9 種類の値をとることとした．

$$\mu \in \{2, 4, 6, 8, 10, 12, 14, 16, 18\},$$
$$\sigma^2 \in \{\frac{1}{8}, \frac{1}{2}, 1, 2, 5, 10, 20, 30, 50\}$$

したがって，とりうるパラメータの組合せは 81 通りである．最適化により決定されたパラメータを用いて学習，予測を行い，10 回の実験のマクロ平均を算出した．また，理想的なパラメータを選択できた場合の性能と比較するために，81 通りのパラメータの組合せすべてに対して学習，予測を行い，10 回の実験のマクロ平均の最も高い結果と最適化の結果を比較した．

今回の実験では，比率制御関数 r（式 (1)）において $m = 10$, 弱分類器制御関数 f（式 (5)）において $a_g = 19$, $b_g = 1$ とした．フレームワーク中の AdaBoost は Scikit-learn (v.0.20.3)[2] を用いて実装し，`n_estimator = 10` とした．

5.4 実験 2： 従来手法との比較

設計した関数 f による有効性を検証するため，従来手法，先行研究との比較実験を行った．

用いた手法は以下の 11 種類である．

- ORG： サンプリングなし
- RUS： 単純なアンダーサンプリング
- RSB： RUSBoost
- EE： EasyEnsemble
- SMT： SMOTE
- ADS： ADASYN
- SWM： SWIM
- Cns： 先行研究（式 (2)）
- Cnv： 先行研究（式 (3)）
- Cnc： 先行研究（式 (4)）
- Gauss： 提案手法（式 (5)）

それぞれの手法に対して 10 回の実験を行い，それらのマクロ平均を算出した．

今回の実験では，比率制御関数 r（式 (1)）において $m = 10$, 先行研究の弱分類器制御関数 f（式 (2)〜(4)）において $a_v = a_c = \frac{1}{5}$, $b_s = 10$, $b_v = 20$, $b_c = 3$，提案手法の弱分類器制御関数 f（式 (5)）において $a_g = 19$, $b_g = 1$ とした．

2 : https://scikit-learn.org/

表 3: グリッドサーチでの G-mean の値

ID	最適化結果	最良結果
D1	.691±.104	.686±.159
D2	.967±.033	.956±.023
D3	.808±.005	.808±.005
D4	.699±.007	.699±.007
D5	.939±.009	.937±.009
D6	.844±.009	.843±.009
D7	.913±.016	.912±.016
D8	.918±.009	.916±.010
D9	.692±.069	.680±.091
D10	.701±.094	.701±.094
D11	.759±.011	.759±.012
D12	.935±.016	.935±.016
D13	.901±.014	.900±.014
D14	.849±.032	.844±.032
D15	.593±.003	.593±.003
D16	.709±.010	.709±.010
D17	.784±.015	.784±.015
D18	.766±.003	.766±.003
D19	.792±.014	.791±.017

ベースとなる分類器には決定木（CART 法）を用い，Scikit-learn (v.0.20.3) を用いて実装した．SWIM 以外の従来手法は imbalanced-learn (v.0.4.3)[3] を用いて実装し，EasyEnsemble のパラメータを `n_estimator = 20` とした．SWIM の実装は，著者が GitHub 上で公開されているソース [4]を基に実装した．また，EasyEnsemble，先行研究，提案手法で用いる AdaBoost のパラメータを `n_estimator = 10` とした．

5.5 結　　果

実験 1 の結果を表 2 に示す．最適化の結果と最良の結果の差の中で，値が 0.005 以下であるものを太字で表している．これより，13 個のデータセットで最良結果との差が 0 に近いことが分かる．よって，グリッドサーチによるパラメータ最適化は有効であるといえる．

しかし，データセット D1, D2 のように差が 0.01 以上となるデータセットも存在した．ここで，グリッドサーチでの G-mean の平均と標準偏差を表 3 に示す．これより，D1, D2 における G-mean の標準偏差が大きいことが分かる．そのため，値の揺れが大きく，最適なパラメータが選択できなかったといえる．

表 4 は，実験 2 の結果において不均衡率 IR について昇順に並べたものである．各データセットにおいて最も高い値となっ

3 : https://imbalanced-learn.readthedocs.io/

4 : https://github.com/cbellinger27/SWIM_Mahalanobis

表 4: 実験 2 の結果

ID	IR	ORG	RUS	RSB	EE	SMT	ADS	SWM	Cns	Cnv	Cnc	Gauss
D8	1.5	.900	.896	**.931**	.916	.900	.898	.896	.917	.914	.919	.917
D19	3.0	**.803**	.785	.761	.760	.787	.760	**.803**	.767	.760	.770	.798
D18	3.5	.677	.714	.641	.762	.690	.689	.678	.765	**.766**	.764	.765
D4	3.5	.581	.616	.528	.689	.585	.584	.580	.694	**.700**	.689	**.701**
D15	3.6	.466	.538	.463	.592	.474	.476	.442	.592	.577	.591	**.593**
D11	3.9	.642	.678	.619	.761	.652	.647	.642	.761	.758	**.762**	**.762**
D16	4.4	.760	.770	.747	.710	**.780**	.771	.757	.712	.713	.708	.713
D6	5.5	.713	.790	.731	.845	.733	.739	.709	.844	.846	.843	**.848**
D5	9.9	.896	.907	.897	.930	.910	.908	.906	.931	**.934**	.930	**.934**
D7	13.4	.810	.854	.786	**.908**	.829	.834	.760	.907	.904	.906	**.908**
D1	16.4	.580	.670	.577	.741	.675	.671	.642	.732	.674	**.751**	.732
D17	18.5	.708	**.794**	.702	.779	.755	.737	.724	.771	.786	.746	.791
D9	24.4	.420	.624	.436	.680	.467	.473	.519	.643	.549	.685	**.694**
D14	26.0	.546	.775	.606	**.863**	.548	.576	.562	.859	.847	.858	.860
D3	29.8	.891	**.928**	.852	.798	.924	.916	.747	.778	.779	.777	.807
D10	44.0	.475	.616	.412	.662	.444	.574	.666	.664	.689	.665	**.735**
D2	65.0	.915	.925	.954	**.963**	.931	.897	.909	.956	.955	.956	.958
D13	112.8	.822	.883	.831	.895	.859	.853	.829	.896	.897	.897	**.900**
D12	577.9	.876	.905	.895	.937	.877	.865	.917	**.939**	.938	.938	.938
平均	50.9	.710	.772	.704	.800	.727	.730	.720	.796	.789	.798	**.808**

たものを太字で表している．これより，提案手法である Gauss で最も高い値をとる傾向にあり，各データセットの結果を平均すると提案手法で最も高い値をとることが分かる．また，提案手法で最も良い性能を示すデータセットは不均衡率の大きさに依存せず，不均衡率が大きいほど従来手法との差は大きい．したがって，不均衡データの分類に対して提案手法が有効であることがいえる．

しかし，データセット D3 や D8, D16 のように，提案手法の結果が従来手法を下回るものも存在した．これらのデータセットでは，サンプリングをしない ORG との差が小さい，または，ORG を下回る結果となっている．基本的に，提案手法は不均衡率が高いときほど良い性能を示す．この点において，D8 や D16 は低い不均衡率であり，従来手法でも十分対処可能と考えられる．しかし，D3 については，不均衡率が 29.8 と高いながら，ORG や RUS など工夫の少ない手法が効果を上げている．D3 のデータが持つ特性をより詳細に分析する必要がある．この点については，現在調査中であり，今後の課題である．

また，先行研究である Cns, Cnv, Cnc と提案手法を比較すると，ほぼすべてのデータセットにおいて先行研究の値を上回っている．また，データセット D12 や D18 のように先行研究の方が高い値をとる場合でも，提案手法との差は 0.001 と小さい値となる．したがって，分類フレームワーク中の関数 f にガウス関数を導入することは有効であることがいえる．

6 おわりに

本稿では，不均衡データ分類フレームワークにおけるサンプリング比率の最適化を提案した．フレームワーク中の弱分類器数制御関数について新たな関数を設計し，そのパラメータをグリッドサーチを用いて最適化した．公開されている 19 個のデータセットを用いた実験により，その有効性が示された．提案手法ではパラメータを離散値として扱っているが，本来は連続値であるため，まだ探索していないパラメータが多く存在する．そのため，パラメータ最適化手法の検討は今後の課題である．

謝　辞

本研究は JSPS 科研費 JP18K18056 および栢森情報科学振興財団の助成を受けて遂行された．

文　献

[1] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory Undersampling for Class-Imbalance Learning. *IEEE Trans. Systems, Man, and Cybernetics, Part B*, Vol. 39, No. 2, pp. 539–550, 2009.

[2] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.*, Vol. 16, pp. 321–357, 2002.

[3] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *IJCNN 2008*, pp. 1322–1328, 2008.

[4] Shiven Sharma, Colin Bellinger, Bartosz Krawczyk, Osmar R. Zaïane, and Nathalie Japkowicz. Synthetic Oversampling with the Majority Class: A New Perspective on Handling Extreme Imbalance. In *ICDM 2018*, pp. 447–456, 2018.

[5] Gustavo E. A. P. A. Batista, Ana L. C. Bazzan, and Maria Carolina Monard. Balancing Training Data for Automated Annotation of Keywords: a Case Study. In *II Brazilian Workshop on Bioinformatics*, pp. 10–18, 2003.

[6] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data.

SIGKDD Explorations, Vol. 6, No. 1, pp. 20–29, 2004.

[7] I. Tomek. Two Modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 7(2), pp. 679–772, 1976.

[8] Dennis L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 2, No. 3, pp. 408–421, 1972.

[9] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. RUSBoost: A Hybrid Approach to Alleviating Class Imbalance. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, Vol. 40, No. 1, pp. 185–197, 2010.

[10] Robert E. Schapire. A Brief Introduction to Boosting. In *IJCAI 1999*, pp. 1401–1406, 1999.

[11] 植原リサ, 駒水孝裕, 小川泰弘, 外山勝彦. 弱分類器の調整に基づく不均衡データ向けアンサンブル・フレームワーク. WebDB Forum 2019 論文集, pp. 81–84, 2019.

[12] Dheeru Dua and Casey Graff. UCI Machine Learning Repository, 2019.

[13] Miroslav Kubat and Stan Matwin. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In *ICML 1997*, pp. 179–186, 1997.