Sistemas de Múltiplos Classificadores
A novel ensemble method for classifying imbalanced data

Daniel Bion Barreiros
Centro de Informática - Universidade Federal de Pernambuco (UFPE)
dbb2@cin.ufpe.br

1. Introdução

Podemos chamar um conjunto de dados de desbalanceado quando o número de instâncias de uma classe é muito menor do que as demais. O artigo escolhido para o projeto [1] trata de desbalanceamento binário, onde as instâncias se distribuem em duas classes, a majoritária e a minoritária. Essa situação se faz presente em problemas do mundo real, quando o comportamento padrão ocorre com mais frequência que o comportamento anômalo, por exemplo, problemas de detecção de fraude em cartão de crédito, detecção de anomalias, detecção de defeito em máquinas ou softwares, etc. Em grande parte desses problemas, a classe minoritária é o objeto de estudo, onde reside o maior interesse dos pesquisadores.

Quando a classe minoritária é muito menor do que a classe majoritária, por exemplo 1% dos casos, algoritmos de classificação supervisionada têm dificuldade em aprender o comportamento da classe minoritária. Dessa forma, mesmo que o algoritmo acerte 99% dos casos, acaba ignorando a classe minoritária que é o nosso objeto de interesse. Pensando nisso, vários métodos da literatura buscam se adequar a esse tipo de problema ajustando a distribuição dos dados através de técnicas de reamostragem, aumentando a classe minoritária, diminuindo a classe majoritária, etc.

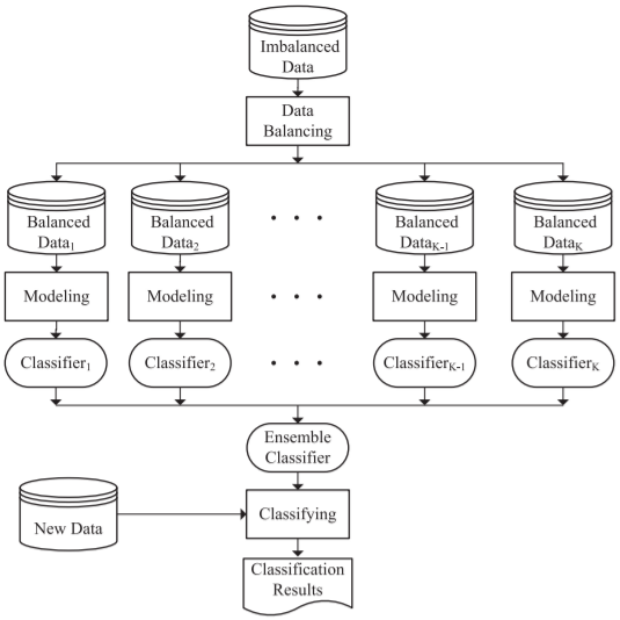

Figura 1. Método proposto para dados desbalanceados.

O método proposto pelo autor do artigo constrói vários conjuntos balanceados a partir do conjunto original como mostra a Figura 1. A classe majoritária é dividida em diversos *bins*, do tamanho da classe minoritária. Em seguida, a classe minoritária é adicionada em cada *bin*, formando vários conjuntos balanceados. Imagine um conjunto de treinamento com 1000 instâncias na classe majoritária e 100 na classe minoritária. A classe majoritária é dividida em 10 *bins* de 100 instâncias, e em cada *bin* se adiciona as 100 instâncias da classe minoritária, totalizando 10 *bins* de 200 instâncias, com 100 de cada classe. O autor propõe duas formas de divisão, o método SplitBal, que particiona a classe majoritária aleatoriamente, e o método ClusterBal que particiona a classe majoritária utilizando um algoritmo de agrupamento. Para cada *bin* é treinado um classificador base e é adicionado ao *pool* de classificadores, logo, o tamanho do *pool* é o número de *bins*.

Para classificar as instâncias de teste o autor utiliza 5 regras de combinação de classificadores da literatura listadas na Tabela 1, e propõe mais 5 regras listadas na Tabela 2.

Tabela 1. Regras utilizadas para combinação de classificadores

Rule	Strategy	Description
Max	$R_1 = \arg\max_{1 \le i \le K} P_{i1}$, $R_2 = \arg\max_{1 \le i \le K} P_{i2}$	Use the maximum classification probability of these K classifiers for each class label
Min	$R_1 = \arg\min_{1 \le i \le K} P_{i1}$, $R_2 = \arg\min_{1 \le i \le K} P_{i2}$	Use the minimum classification probability of these K classifiers for each class label
Product	$R_1 = \prod_{i=1}^{K} P_{i1}$, $R_2 = \prod_{i=1}^{K} P_{i2}$	Use the product of classification probability of these K classifiers for each class label
Majority vote	$R_1 = \sum_{i=1}^{K} f(P_{i1}, P_{i2})$, $R_2 = \sum_{i=1}^{K} f(P_{i2}, P_{i1})$	For the ith classifier, if $P_{i1} \ge P_{i2}$, class C_1 gets a vote, if $P_{i2} \ge P_{i1}$, class C_2 gets a vote
Sum	$R_1 = \sum_{i=1}^{K} P_{i1}$, $R_2 = \sum_{i=1}^{K} P_{i2}$	Use the summation of classification probability of these K classifiers for each class label

The function f(x,y) is defined as follows:

$$f(x,y) = \begin{cases} 1 & x \ge y \\ 0 & x < y \end{cases} \quad (1)$$

As regras listadas acima utilizam a probabilidade que cada classificador dá para uma instância de teste. Sendo $Pi1$a probabilidade do classificador i classificar a instância de teste como classe 1, e $Pi2$ a probabilidade do classificador i classificar a instância de teste como classe 2. A regra do máximo usa a maior probabilidade entre os K classificadores, enquanto que a regra do mínimo utiliza a menor probabilidade entre os classificadores. As regras de produto e soma combinam todas as probabilidades através de somatório e produtório respectivamente, enquanto que a regra do voto majoritário escolhe a classe da instância de teste baseado na soma dos votos de cada classificador.

Tabela 2. Regras utilizadas para combinação de classificadores

Rule	Strategy	Description
MaxDistance	$R_1 = \arg\max_{1 \le i \le K} \frac{P_{i1}}{D_{i1}+1}$, $R_2 = \arg\max_{1 \le i \le K} \frac{P_{i2}}{D_{i2}+1}$	Use the inverse of average distance to adjust the Max Rule
MinDistance	$R_1 = \arg\min_{1 \le i \le K} \frac{P_{i1}}{D_{i1}+1}$, $R_2 = \arg\min_{1 \le i \le K} \frac{P_{i2}}{D_{i2}+1}$	Use the inverse of average distance to adjust the Min Rule
ProDistance	$R_1 = \prod_{i=1}^{K} \frac{P_{i1}}{D_{i1}+1}$, $R_2 = \prod_{i=1}^{K} \frac{P_{i2}}{D_{i2}+1}$	Use the inverse of average distance to adjust the Product Rule
MajDistance	$R_1 = \sum_{i=1}^{K} \frac{f(P_{i1},P_{i2})}{D_{i1}+1}$, $R_2 = \sum_{i=1}^{K} \frac{f(P_{i2},P_{i1})}{D_{i2}+1}$	Use the inverse of average distance to adjust the Majority Vote Rule
SumDistance	$R_1 = \sum_{i=1}^{K} \frac{P_{i1}}{D_{i1}+1}$, $R_2 = \sum_{i=1}^{K} \frac{P_{i2}}{D_{i2}+1}$	Use the inverse of average distance to adjust the Sum Rule

As regras propostas pelo autor, utilizam a distância entre a instância de teste e cada instância de treinamento, sendo $Di1$ a média das distâncias entre a instância de teste e todas as instâncias de treinamento da classe 1 do *bin i*, e $Di2$ a média das distâncias entre a instância de teste e todas as instâncias de treinamento da classe 2 do *bin i*. O autor utiliza a distância entre a instância de teste e cada classe como peso para as probabilidades de cada classificador. Dessa forma é possível dar um peso maior para os classificadores treinados com *bins* mais parecidos com a instância de teste.

Neste estudo foram utilizados 46 conjuntos de dados de alvo binário altamente desbalanceados retirados do repositório Keel [2]. Para cada conjunto foi aplicada a estratégia de validação cruzada 10-*fold*. Como algoritmos base foram utilizados o Naive Bayes, C4.5, RIPPER, Random Forest, SMO e IBK, todos implementados no Weka [3] com parâmetros padrão. Para a replicação do artigo foi utilizado o pacote RWeka [4], que conta com uma interface para o Weka diretamente do R. Como métrica de validação de desempenho foi utilizado apenas o AUC (*area under* ROC *curve*). As principais investigações do artigo foram descobrir que regra de combinação funciona melhor com o SplitBal e com o ClusterBal, comparar a melhor regra de um com a melhor regra do outro e comparar o melhor método proposto com algoritmos de tratamento de dados desbalanceados da literatura.

2. Experimentos e Proposta

Os experimentos foram conduzidos utilizando os mesmos 46 conjuntos de dados desbalanceados. Para cada conjunto foi aplicada a técnica de validação cruzada 10-*fold*, em seguida as técnicas de balanceamento SplitBal (particionamento aleatório) e ClusterBal (particionamento utilizando o *k-means*). Todos os classificadores base foram treinados para os *bins* obtidos no processo, e finalmente escolhido o resultado final para cada instância de teste utilizando cada uma das dez regras.

A fim de investigar o desempenho das regras propostas pelo autor, foram gerados os resultados presentes na Tabela 3 para o SplitBal e na Tabela 4 para o ClusterBal. No artigo original, 5 dos 6 algoritmos se deram melhor com a regra *MaxDistance* tanto para o SplitBal quanto para o ClusterBal, enquanto que na replicação cada algoritmo se deu melhor com uma regra diferente. Contudo, fica claro que as regras de combinação propostas pelo autor geram resultados superiores às regras de combinação presentes na literatura. Segundo os rankings mostrados na Tabela 5 e Tabela 6, a melhor regra no geral foi a *MinDistance* para o SplitBal, enquanto que para o ClusterBal a melhor regra foi o *MaxDistance*.

Tabela 3. Média de AUC de 46 conjuntos de dados para diferentes regras com SplitBal

Classifier	Max	Min	Product	Majority	Sum	MaxDistance	MinDistance	ProDistance	MajDistance	SumDistance
Naive Bayes	0.8351	0.8457	0.8158	0.8350	0.8305	0.8041	0.8360	0.8297	0.8340	**0.8467**
C4.5	0.7704	0.7715	0.7826	0.7772	0.7667	0.7747	0.7757	0.7783	0.7620	**0.7835**
RIPPER	0.7838	0.7633	0.7689	0.7757	0.7791	0.7645	0.7776	0.7672	**0.7875**	0.7619
Random Forest	0.8844	0.8538	0.8646	0.8895	0.8886	0.8888	**0.8896**	0.8562	0.8878	0.8479
SMO	0.7899	0.7836	0.7895	0.8090	0.7835	0.8001	0.7791	0.7875	**0.8189**	0.7961
IBK	0.8269	0.8370	0.8310	0.8327	0.8360	0.8340	0.8531	**0.8588**	0.8422	0.8385

Tabela 4. Média de AUC de 46 conjuntos de dados para diferentes regras com ClusterBal

Classifier	Max	Min	Product	Majority	Sum	MaxDistance	MinDistance	ProDistance	MajDistance	SumDistance
Naive Bayes	0.6155	0.5330	0.5354	0.5453	0.6112	0.6393	0.5869	0.5892	0.5690	**0.6473**
C4.5	0.6648	0.5262	0.5263	0.5262	0.6411	**0.6847**	0.5932	0.5291	0.5963	0.6595
RIPPER	0.6770	0.5223	0.5223	0.5223	0.6563	**0.7216**	0.6249	0.6024	0.6043	0.6759
Random Forest	0.6798	0.7643	0.7643	0.7148	0.6746	0.7016	0.8072	**0.8274**	0.7827	0.7393
SMO	0.7459	0.5159	0.5159	0.5159	0.7177	0.7632	0.5208	0.5783	0.5562	**0.7673**
IBK	0.7299	0.7816	0.7816	0.7187	0.7176	0.8171	0.8025	**0.8378**	0.7678	0.7987

Tabela 5. Ranking das regras com SplitBal

Classifier	Max	Min	Product	Majority	Sum	MaxDistance	MinDistance	ProDistance	MajDistance	SumDistance
Naive Bayes	4	2	9	5	7	10	3	8	6	1
C4.5	8	7	2	4	9	6	5	3	10	1
RIPPER	2	9	6	5	3	8	4	7	1	10
Random Forest	6	9	7	2	4	3	1	8	5	10
SMO	5	8	6	2	9	3	10	7	1	4
IBK	10	5	9	8	6	7	2	1	3	4
Sum	35	40	39	26	38	37	25	34	26	30
Rank	6	10	9	2	8	7	1	5	2	4

Tabela 6. Ranking das regras com ClusterBal

Classifier	Max	Min	Product	Majority	Sum	MaxDistance	MinDistance	ProDistance	MajDistance	SumDistance
Naive Bayes	3	10	8	9	4	2	6	5	7	1
C4.5	2	9	8	9	4	1	6	7	5	3
RIPPER	2	8	8	8	4	1	5	7	6	3
Random Forest	9	4	4	7	10	8	2	1	3	6
SMO	3	8	8	8	4	2	7	5	6	1
IBK	8	5	5	9	10	2	3	1	7	4
Sum	27	44	41	50	36	16	29	26	34	18
Rank	4	9	8	10	7	1	5	3	6	2

Segundo o autor um dos problemas do ClusterBal é que algoritmos de agrupamento não garantem um número similar de instâncias em todos os grupos, dessa forma o ClusterBal pode gerar novos conjuntos desbalanceados. Experimentos realizados com SplitBal para um dos conjuntos confirmam que a classe majoritária tem aproximadamente o mesmo tamanho em todos os *bins*. Já no ClusterBal o tamanho da classe majoritária em cada *bin* varia muito, como podemos ver na Tabela 8.

Tabela 7. Tamanho das classes utilizando os métodos propostos

Bin	SplitBal		ClusterBal	
	Majority	Minority	Majority	Minority
1	321	293	760	293
2	321	293	525	293
3	321	293	154	293
4	321	293	82	293
5	321	293	83	293

Analisando a Tabela 7 notamos que em alguns casos o ClusterBal pode tornar a tarefa de classificação mais difícil ainda, gerando *bins* com a classe majoritária bem maior que a minoritária, e *bins* onde a classe majoritária acaba se tornando muito menor do que a classe minoritária. Ao combinar o resultado de classificadores gerados com situações tão discrepantes acabamos piorando o resultado final da classificação.

Após a replicação dos experimentos surgiu o interesse de investigar o desempenho do método desenvolvido pelo autor utilizando os algoritmos mais simples da literatura de seleção dinâmica de classificadores, supondo que as regras de combinação propostas pelo autor podem ser consideradas como parte dessa área, pois a resposta de cada classificador do *pool* varia de acordo com a distância calculada em cima de cada instância de teste.

Partindo do ponto que o método proposto particiona a classe majoritária em vários subconjuntos e gera um classificador para cada subconjunto, podemos aplicar as técnicas OLA (*Overall Local Accuracy*) e LCA (*Local Class Accuracy*) [5] de seleção dinâmica para escolher o melhor classificador da região de competência da instância de teste. Para isso, o conjunto de treinamento, sem ser particionado, foi utilizado como conjunto de validação, e as regras de combinação foram trocadas pelos algoritmos de seleção dinâmica.

Observamos na Tabela 8 os resultados dos classificadores com diferentes técnicas de tratamento de dados desbalanceado e os métodos propostos. A combinação do SplitBal + MinDistance tem desempenho superior aos métodos da literatura apenas nos classificadores Naive Bayes, Random Forest e IBK. O ClusterBal + MaxDistance não supera nenhum método da literatura.

Tabela 8. Média de AUC dos 46 conjuntos de dados para diferentes métodos de desbalanceamentos de dados

		Conventional Imbalance Methods			Proposed by the Author		My methods			
Classifier	Original	RUS	ROS	SMOTE	SplitBal+MinDist	ClusterBal+MaxDist	SplitBal+OLA	SplitBal+LCA	ClusterBal+OLA	ClusterBal+LCA
Naive Bayes	0.8130	0.8157	0.8139	0.8095	0.8360	0.6393	0.8395	**0.8453**	0.6708	0.7191
C4.5	0.8162	0.8159	0.8153	0.8160	0.7757	0.6847	**0.8445**	0.8106	0.6814	0.8206
RIPPER	0.8118	0.8168	0.8150	0.8144	0.7776	0.7216	**0.8302**	0.7870	0.7130	0.8113
Random Forest	0.8165	0.8150	0.8157	0.8156	**0.8896**	0.7016	0.8762	0.8707	0.7038	0.8386
SMO	0.8138	0.8161	0.8171	0.8173	0.7791	0.7632	**0.8473**	0.8451	0.7341	0.8232
IBK	0.8172	0.8189	0.8159	0.8159	0.8531	0.8151	**0.8545**	0.8459	0.7128	0.8255

Dentre as combinações propostas neste estudo, a aplicação de seleção dinâmica de classificador no SplitBal teve um resultado melhor com o OLA para todos os algoritmos menos o Naive Bayes, que foi melhor com o LCA. Já para o ClusterBal o LCA obteve um resultado superior ao OLA em todos os algoritmos base.

Quando comparado com os demais métodos o SplitBal + OLA obtém os melhores resultados no C4.5, RIPPER, SMO e IBK, perdendo apenas para o SplitBal + MaxDistance com Random Forest, e para o SplitBal + LCA com Naive Bayes.

É importante frisar que o uso da seleção dinâmica de classificador LCA com o ClusterBal. Na replicação do artigo nenhuma regra foi capaz de superar o desempenho dos métodos da literatura. Contudo, com a aplicação do LCA os resultados acabam sendo melhores ao selecionar apenas um classificador ao invés de combinar todos os classificadores do *pool*.

Outro ponto importante é o tempo de execução dos métodos. Os métodos OLA e LCA restringem a interação das instâncias de teste apenas com sua região de competência. Enquanto que as regras de combinação propostas pelo autor utilizam da distância de cada instância de teste para cada *bin* gerado pelos métodos de particionamento, aumentando exponencialmente o custo computacional.

3. Considerações Finais

Foi apresentado neste estudo a replicação de um artigo sobre tratamento de dados desbalanceados utilizando combinação de classificadores. Os métodos propostos pelo autor divide a classe majoritária em subconjuntos, treinando um classificador para cada subconjunto. Em seguida, combina os resultados dos classificadores através de regras de combinação também propostas. O autor compara as regras de combinação propostas entre si, e seleciona a melhor regra para cada uma dos métodos de particionamento. Ao final, compara a melhor regra de cada método com os métodos de tratamento de dados desbalanceados da literatura.

Foi observado que nem todos os classificadores treinados utilizando os métodos propostos obtiveram um resultado superior aos classificadores puros, ou seja, sem nenhuma técnica de tratamento de dados desbalanceados. Sendo assim, surgiu a necessidade de aprimorar os métodos desenvolvidos pelo autor, através de aplicação de métodos de seleção dinâmica de classificador da literatura. Dessa

forma, a combinação dos classificadores foi substituída pela seleção dinâmica de classificador.

Observamos resultados superiores ao do artigo ao aplicar as técnicas OLA e LCA tanto para o SplitBal, quanto para o ClusterBal, concluindo que os métodos propostos neste estudo são capazes de lidar com a classificação de dados desbalanceados, e possuem desempenho superior aos métodos da literatura.

Referências

[1] Zhongbin Sun, Qinbao Song, Xiaoyan Zhu, Heli Sun, Baowen Xu, Yuming Zhou, A novel ensemble method for classifying imbalanced data, Pattern Recognition, Volume 48, Issue 5, 2014, Pages 1623-1637.

[2] J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, et al., Keel: a software tool to assess evolutionary algorithms for data mining problems, Soft Comput. – Fusion Found. Methodol. Appl. 13 (2009) 307–318.

[3] Witten I. H. and Frank E. (2005). Data Mining: Practical Machine Learning Tools and Techniques, 2nd edition. Morgan Kaufmann, San Francisco.

[4] Hornik K., Buchta C. and Zeileis A. (2009). Open-Source Machine Learning: R Meets Weka. Computational Statistics, 225-232.

[5] Woods K., Kegelmeyer Jr W. P. and Bowyer K., Combination of multiple classifiers using local accuracy estimates, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 19, no. 4, pp. 405-410, 1997.