Sistemas de Múltiplos Classificadores
Lista de Exercícios 2

Daniel Bion Barreiros
dbb2@cin.ufpe.br

Os experimentos foram conduzidos para comparar *ensembles* gerados com algoritmos de poda, em diferentes conjuntos de validação. Como classificador base, foi utilizado o *Perceptron.* As bases de dados escolhidas, "CM1/software defect prediction" [1] e "Class-level data for KC1" [2] do *Promise Software Engineering Repository*, contém informações sobre detecção de falhas em *softwares*. Ambas as bases de dados são binárias, contendo duas classes desbalanceadas. A base CM1 possui 449 instâncias e 22 features, tendo em torno de 9% das instâncias pertencendo à classe minoritária, enquanto que a base KC1 possui 147 instâncias e 95 features, tendo em torno de 6% das instâncias pertencendo à classe minoritária.

Buscamos verificar o comportamento dos algoritmos de poda com conjuntos de validação diferentes, criando três situações: um conjunto de validação utilizando as instâncias difíceis do conjunto de treinamento, outro conjunto utilizando instâncias fáceis do conjunto de treinamento, e por último o conjunto completo de treinamento. Como forma de verificar a dificuldade das instâncias, foi utilizado o algoritmo *k-Disagreeing Neighbors* (kDN) [3], que calcula a fração dos vizinhos mais próximos de uma instância que não são da mesma classe que ela. O valor do parâmetro k foi escolhido empiricamente como 10, logo, se entre os 10 vizinhos mais próximos de uma instância apenas 3 são da mesma classe que ela, essa instância tem uma dificuldade de 0.7 em uma escala 0 a 1. O *threshold* de separação entre as instâncias fáceis e difíceis também foi escolhido empiricamente como 0.2. Esse valor foi suficiente para gerar conjuntos com aproximadamente 20% das instâncias mais difíceis da base CM1, e conjuntos com aproximadamente 5% das instâncias difíceis da base KC1.

Os algoritmos de poda implementados foram o *Best First Pruning* e *Reduce-Error Pruning.* O *Best First* ordena os classificadores pela taxa de erro, cria N *ensembles* de tamanho 1 a N e escolhe o *ensemble* com a menor taxa de erro, sendo N o tamanho do *pool*. O *Reduce-Error* é inicializado com o classificador de menor taxa de erro, e vai adicionando iterativamente ao *ensemble* o classificador que produz o menor erro de validação, até que nenhum classificador consiga reduzir o erro.

Foi verificado experimentalmente que a geração de *ensembles* do algoritmo *Best First* para as bases escolhidas gera empates entre a taxa de acerto dos primeiros *ensembles* como mostra a Figura 1 e Figura 2. Com o intuito de testar o aumento da diversidade do ensemble, e evitar que o mesmo tenha apenas um classificador, foi utilizado como critério de desempate o *ensemble* com mais classificadores. Já no *Reduce-Error* o critério de parada foi modificado

para que o *ensemble* aceite classificadores que não melhorem a taxa de erro atual, contudo que não piorem, caso o tamanho seja menor que 3.

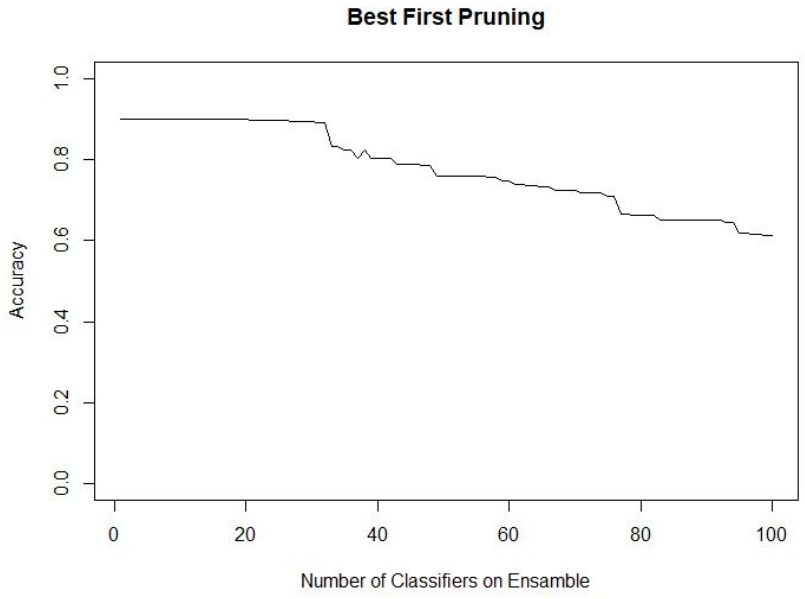

Figura 1. Taxa de acerto dos ensembles gerados pelo *Best First* na base CM1

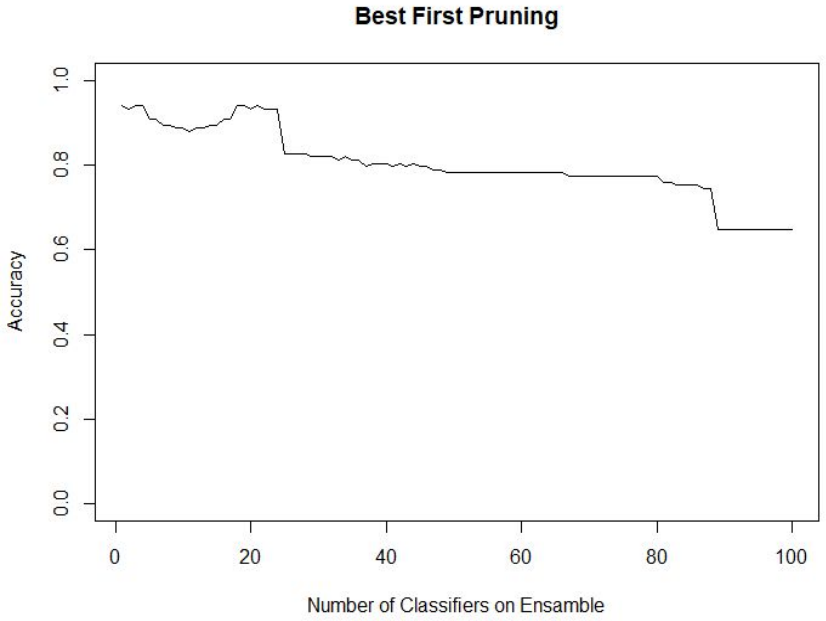

Figura 2. Taxa de acerto dos ensembles gerados pelo *Best First* na base KC1

Para verificar o desempenho dos classificadores foram utilizadas as métricas de *Accuracy*, AUC, *F-Measure* e *G-Mean*. Para avaliar a diversidade dos *ensembles* gerados foram utilizados os algoritmos de diversidade pareados *Q-Statistic* e *Kappa-Statistic*. O *Q-Statistic* varia entre -1 e 1, tendo valor positivo quando os classificadores reconhecem o

mesmo objeto corretamente e valor negativo quando cometem erros em objetos diferentes. O *Kappa-Statistic* também varia entre -1 e 1, tendo valor positivo se os classificadores concordam entre si, valor próximo a zero se os classificadores concordam aleatoriamente, e negativo caso concordem menos do que o esperado normalmente.

A partir das métricas de desempenho das bases na Tabela 1 e Tabela 2, começamos a analisar o impacto dos algoritmos de poda nos três cenários desenvolvidos. Para o cenário A (Conjunto completo) da base CM1 ambos algoritmos de poda geram *ensembles* que melhoraram a classificação da classe majoritária, piorando a classificação da classe minoritária, como mostra a queda da área da curva ROC. No cenário B (Instâncias difíceis) podemos chegar à conclusão que os algoritmos de poda selecionam os classificadores mais especializados nas instâncias difíceis, acertando mais a classe minoritária e aumentando a área de curva ROC enquanto diminui as demais métricas. O cenário C (Instâncias fáceis) mantém o comportamento do cenário A.

Cenário A					
Accuracy			AUC		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.6505	**0.8976**	**0.8976**	**0.5769**	0.5022	0.5022
G-mean			F-measure		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.8824	**0.9012**	**0.9012**	0.8277	**0.9459**	**0.9459**
Cenário B					
Accuracy			AUC		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.6507	0.2164	0.2388	0.5769	**0.5988**	**0.5936**
G-mean			F-measure		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.8824	0.7757	0.8226	**0.8277**	0.2890	0.3303
Cenário C					
Accuracy			AUC		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.6507	**0.8996**	**0.9016**	**0.5769**	0.5011	0.5000
G-mean			F-measure		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.8824	**0.9014**	**0.9016**	0.8277	**0.9471**	**0.9482**

Tabela 1. Resultados dos cenários de validação para a base CM1

Cenário A					
Accuracy			AUC		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.1347	**0.9176**	**0.9176**	0.5107	**0.5846**	**0.5846**
G-mean			F-measure		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.9166	**0.9461**	**0.9461**	0.8461	**0.9539**	**0.9539**
Cenário B					
Accuracy			AUC		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.1347	0.0680	**0.1385**	**0.5107**	0.5000	**0.5225**
G-mean			F-measure		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.0916	0.0000	**0.2900**	0.0846	**0.9539**	**0.9539**
Cenário C					
Accuracy			AUC		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.1347	**0.9176**	**0.9176**	0.5107	**0.5846**	**0.5846**
G-mean			F-measure		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.9166	**0.9461**	**0.9461**	0.8461	**0.9539**	**0.9539**

Tabela 2. Resultados dos cenários de validação para a base KC1

Para a base KC1, verificamos que o desempenho dos *ensembles* gerados pelos algoritmos de poda são iguais ou superam os resultados do *pool* original para os três cenários.

Analisando as medidas de diversidade no *pool* original e nos *ensembles* gerados pelos métodos de poda, mostradas na Tabela 3 e Tabela 4, verificamos que apesar de terem desempenhos parecidos, o algoritmo *Best First* apresenta uma diversidade superior ao *Reduce-Error*.

Concluímos que os algoritmos de poda estática de classificadores podem trazer vantagens para a classificação final, como o aumento de desempenho e de diversidade, em contrapartida aumentando o custo computacional do método.

Cenário A					
Q-Statistic			Kappa-Statistic		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.3102	**1.0000**	0.2504	0.3819	**0.9874**	0.3811
Cenário B					
Q-Statistic			Kappa-Statistic		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.3102	**0.4815**	-0.1760	**0.3819**	**0.3508**	0.0595
Cenário C					
Q-Statistic			Kappa-Statistic		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.3102	**1.0000**	0.2504	0.3819	**0.9938**	0.4130

Tabela 3. Teste de diversidade para a base CM1

Cenário A					
Q-Statistic			Kappa-Statistic		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.5317	**0.6200**	0.3000	**0.7046**	0.6545	0.4430
Cenário B					
Q-Statistic			Kappa-Statistic		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.5317	**0.8800**	0.6000	0.7046	**0.8700**	0.3567
Cenário C					
Q-Statistic			Kappa-Statistic		
Sem poda	Best First	Reduce Error	Sem poda	Best First	Reduce Error
0.5317	**0.7000**	0.3000	**0.7046**	**0.7222**	0.4430

Tabela 4. Teste de diversidade para a base KC1

Referências

[1] Koru, A. G. “Class-level data for KC1.”, http://promise.site.uottawa.ca/SERepository/datasets/kc1-class-level-defectiveornot.arff

[2] Menzies, T. “CM1/software defect prediction.” http://promise.site.uottawa.ca/SERepository/datasets/cm1.arff

[3] Walmsley, F. N. “Um Método de Geração de Pools de Classificadores Usando Instance Hardness.” Trabalho de Graduação - CIn - UFPE, Recife, 2017.