Sistemas de Múltiplos Classificadores

Lista 1 - Andréa Duque

1 Pools homogêneos utilizando Bagging e Random Subspace

1.1 Metodologia dos Experimentos

Essa seção detalha a geração do pool de classificadores utilizando as técnicas de Bagging e Random Subspace, e como se deu a avaliação dessas técnicas. Inicialmente, escolhi as bases binárias JM1 e CM1 do repositório Promise para realizar os experimentos. O objetivo de treinar os classificadores nesses dados é detectar defeitos em software, realizando uma classificação binária. Ambas as duas bases possuem o mesmo número de atributos, 21 no total, entretanto são bem distintas em relação ao número de instâncias. A base JM1 possui 10885 instâncias, enquanto a base CM1 apenas 498. Ambas as bases são bastante desbalanceadas em relação ao número de classes, possuindo muito mais exemplos negativos que positivos. Enquanto a base JM1 possui uma proporção de aproximadamente 80% de exemplos negativos, a base CM1 possui cerca de 90%. Na seção de resultados, analiso o impacto disso no desempenho dos modelos. Não foi realizado nenhum preprocessamento de dados na base CM1. Já na base JM1, quatro variáveis com valores desconhecidos foram preenchidas com a média da característica a qual a variável pertence. A composição detalhada das bases encontra-se na Tabela 1.

Para avaliar o desempenho dos pools, utilizou-se um esquema de validação cruzada estratificada 10-fold, a fim de manter a proporção entre as classes para cada base de dados. Para fins comparativos, manteve-se a mesma divisão de folds nos experimentos para cada base.

Em todos os experimentos, gerei pools de 100 classificadores. A classe prevista para um exemplo de teste é dada como a classe com o maior número de votos dentre as classes previstas pelos 100 classificadores bases (Hard Voting). Utilizei como classificadores base Árvore de Decisão e Perceptron. Em relação aos parâmetros dos modelos base utilizados, as árvores utilizadas foram totalmente crescidas e sem poda, e como critério de divisão utilizou-se a impureza de Gini. Já para o Perceptron, utilizou-se arbitrariamente o número de iterações igual a 1000.

Para a técnica de Bagging, variou-se o número máximo de instâncias a serem retiradas do conjunto de treinamento, com o objetivo de avaliar o impacto desse parâmetro na performance do sistema de classificadores. Já para a técnica de Random Subspace, utilizou-se como número máximo de atributos, para criar os subconjuntos, 50% do número de atributos total.

Os experimentos foram implementados na linguagem Python, utilizando o pacote Scikit-Learn. Utilizei a classe BaggingClassifier do Scikit para geração de pools tanto utilizando a técnica de Bagging quanto de Random Subspace.

Na próxima seção, analiso o impacto da escolha do classificador base na construção do sistema de classificadores. Em seguida, observo o efeito da variação do número de instâncias retiradas no desempenho da técnica de Bagging. Por fim, comparo as técnicas de Bagging e Random Subspace aplicadas a cada dataset em relação as métricas utilizadas.

Dataset	#Instâncias	#Atributos	#Classes	Divisão Classes
JM1	10885	21	2	8779;2106
CM1	498	21	2	449;49

Tabela 1: Composição detalhada das diferentes bases utilizadas nos experimentos.

1.2 Resultados e Discussão

As métricas utilizadas para avaliar o desempenho do pool foram acurácia, AUC, G-mean e F-score. Para cada métrica, calculou-se a média e desvio padrão dos resultados de cada pool de classificadores, treinados e avaliados nos dez conjuntos de treino e teste gerados na validação

cruzada. Os resultados para a base JM1 estão detalhados na Tabela 2 e 3 e para a base CM1 nas tabelas 4 e 5, para as técnicas de Bagging e Random Subspace respectivamente.

Houve uma grande disparidade entre os resultados de acordo com cada métrica utilizada. Esse resultado já era esperado, pois as métricas de acurácia e AUC podem resultar em valores muito otimistas para bases desbalanceadas. Um exemplo extremo da diferença entre as métricas ocorreu principalmente nos resultados da base CM1 (Tabelas 4 e 5). Utilizando a técnica Random Subspace e o classificador base Árvore de Decisão, obtivemos bons valores de acurácia (0.88) e AUC (0.76) e péssimos valores de G-mean (0.14) e F-score (0.10). Isso ocorreu porque o ensemble aprendeu muito a classe negativa e pouca da positiva, ou seja, teve uma baixa taxa de falsos positivos, mas errando boa parte da classe positiva. Já utilizando Random Subspace e Perceptron para a base CM1, para a maioria dos folds de teste, o ensemble detectou nenhum exemplo da classe positiva. As métricas G-mean e F-score são menos sensíveis a dados desbalanceados, por isso são mais adequadas para avaliar classificadores nesse caso. Dentre essas duas métricas utilizarei a métrica F-score para comparar os resultados, por ser a mais popular.

Considerando ambas as bases, ambas as técnicas de Bagging e Random Subspace apresentaram melhores resultados quando utilizei árvores de decisão como classificador-base, em relação aos ensembles de Perceptron. Uma explicação para isso é o fato de Perceptrons serem mais sensíveis a desbalanceamento que árvores de decisão, e como não estamos utilizando nenhuma forma de penalização para as classes, os resultados com Perceptron não ficaram bons.

Em relação a técnica de Bagging, o número máximo de instâncias utilizadas para treino melhorou os resultados quando foram utilizadas árvores de decisão para a base CM1, obtendo o melhor resultado quando foi utilizado todo o conjunto de treino. Foi obtido 0.12 de F-score em comparação a 0.02 utilizando 50% do conjunto. Uma possível causa para esse resultado foi o fato do conjunto ser muito pequeno e bastante desbalanceado, e desse modo o número de instâncias da classe positiva aumentou. Para a base JM1, utilizamos o teste não-paramétrico de Wilcoxon pareado para saber se houve melhora significativa entre pool gerado com Bagging utilizando 100% do conjunto e utilizando 70%. Obtive p-value = 0.053, concluindo que não houve diferenças significativas. A variação desse parâmetro utilizando Percetron também não apresentou melhoras no desempenho.

Para a base JM1, obtive o melhor F-score utilizando Bagging e árvores de decisão (0.34+-(0.02)), treinando com 100% do conjunto de treino. Já utilizando Random Subspace e árvores de decisão com máximo de 50% de features, obtivemos um F-score de 0.32+-(0.03). Utilizei o teste de Wilcoxon pareado, aplicado as amostras de tamanho dez obtidas na validação cruzada, para saber se houve melhoras significativas do Bagging em relação ao Random Subspace para essa base. Obtive p-value = 0.005, concluindo que houve melhoras significativas para $\alpha = 0.01$. Na Figura 1, observa-se o boxplot das amostras obtidas nesse experimento para o Bagging e para o Random Subspace e utilizadas na comparação.

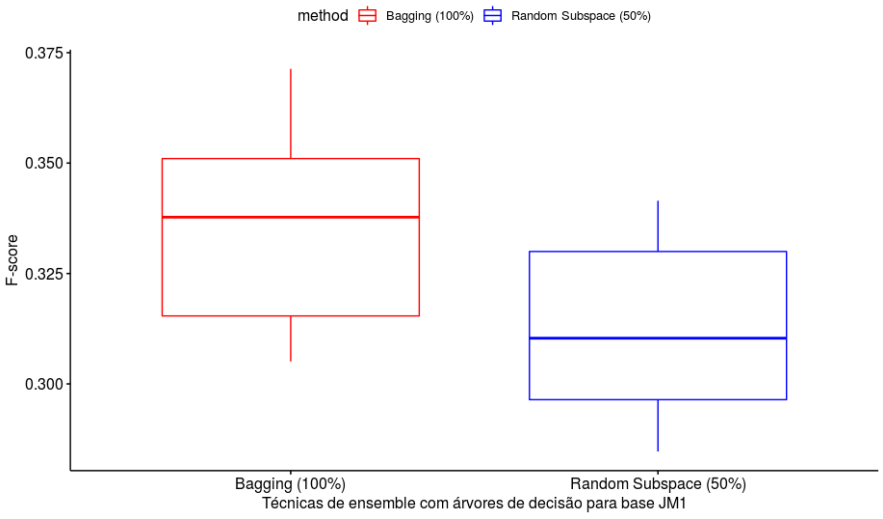

Figura 1: a) Resultados obtidos na base JM1 utilizando Bagging com árvores de decisão para 100% do conjunto de treino e utilizando Random Subspace com árvores de decisão e máximo de 50% de features selecionadas.

1.3 Conclusões

Podemos concluir que o desbalanceamento afetou o desempenho dos classificadores gerados. A variação do percentual do conjunto de treino na geração de pool com Bagging não promoveu melhoras significativas para a base JM1. Para a base CM1, os ensembles com Perceptron, gerados tanto com Bagging e Random Subspace não aprenderam a classe positiva. Para melhorar esses resultados, acredito que além de um esquema de balanceamento dos dados é necessário um esquema de pesos para penalizar a classe negativa.

	Arvore de Decisao				Perceptron			
Perc	Acuracia	AUC	G-mean	F-score	Acuracia	AUC	G-mean	F-score
0.5	0.81+-(0.0)	0.75+-(0.02)	0.45+-(0.02)	0.31+-(0.02)	0.78+-(0.01)	0.61+-(0.06)	0.37+-(0.04)	0.21+-(0.04)
0.6	0.82+-(0.01)	0.75+-(0.02)	0.46+-(0.02)	0.32+-(0.02)	0.78+-(0.01)	0.63+-(0.03)	0.40+-(0.02)	0.23+-(0.02)
0.7	0.82+-(0.0)	0.75+-(0.02)	0.47+-(0.02)	0.33+-(0.02)	0.78+-(0.01)	0.60+-(0.06)	0.38+-(0.04)	0.22+-(0.04)
0.8	0.82+-(0.01)	0.75+-(0.02)	0.47+-(0.02)	0.33+-(0.03)	0.78+-(0.01)	0.63+-(0.02)	0.40+-(0.05)	0.23+-(0.05)
0.9	0.82+-(0.01)	0.75+-(0.02)	0.48+-(0.02)	0.34+-(0.03)	0.78+-(0.01)	0.61+-(0.06)	0.38+-(0.02)	0.21+-(0.02)
1.0	0.82+-(0.0)	0.75+-(0.02)	0.48+-(0.02)	**0.34+-(0.02)**	0.78+-(0.01)	0.63+-(0.02)	0.40+-(0.03)	0.23+-(0.02)

Tabela 2: Resultados para Bagging utilizando a base JM1 e variando o numero de instancias retiradas por bootstrap.

	Acuracia	AUC	G-mean	F-score
Arvore de Decisao	0.82+-(0.01)	0.75+-(0.01)	0.46+-(0.02)	**0.32+-(0.03)**
Perceptron	0.81+-(0.00)	0.59+-(0.08)	0.31+-(0.04)	0.16+-(0.04)

Tabela 3: Resultados para Random Subspace utilizando a base JM1 utilizando 50% de features maximos

	Arvore de Decisao				Perceptron			
Max Amostras	Acuracia	AUC	G-mean	F-score	Acuracia	AUC	G-mean	F-score
0.5	0.89+-(0.03)	0.76+-(0.10)	0.04+-(0.13)	0.02+-(0.08)	0.90+-(0.00)	0.27+-(0.14)	0.00+-(0.00)	0.00+-(0.00)
0.60	0.88+-(0.03)	0.76+-(0.09)	0.09+-(0.19)	0.07+-(0.14)	0.90+-(0.00)	0.26+-(0.14)	0.00+-(0.00)	0.00+-(0.00)
0.7	0.89+-(0.03)	0.76+-(0.10)	0.14+-(0.21)	0.10+-(0.16)	0.90+-(0.00)	0.27+-(0.14)	0.00+-(0.00)	0.00+-(0.00)
0.8	0.89+-(0.03)	0.76+-(0.10)	0.09+-(0.19)	0.06+-(0.13)	0.90+-(0.00)	0.26+-(0.14)	0.00+-(0.00)	0.00+-(0.00)
0.9	0.88+-(0.03)	0.75+-(0.11)	0.14+-(0.21)	0.10+-(0.15)	0.90+-(0.00)	0.27+-(0.14)	0.00+-(0.00)	0.00+-(0.00)
1	0.89+-(0.03)	0.75+-(0.09)	0.28+-(0.39)	**0.12+-(0.16)**	0.90+-(0.00)	0.27+-(0.14)	0.00+-(0.00)	0.00+-(0.00)

Tabela 4: Resultados para Bagging utilizando a base CM1 e variando o número de instâncias retiradas por bootstrap.

	Acuracia	AUC	G-mean	F-score
Arvore de Decisao	0.88+-(0.04)	0.76+-(0.08)	0.14+-(0.21)	**0.10+-(0.16)**
Perceptron	0.90+-(0.01)	0.27+-(0.14)	0.00+-(0.00)	0.00+-(0.00)

Tabela 5: Resultados para Random Subspace utilizando a base CM1 utilizando 50% de features máximas