Sistemas de Múltiplos Classificadores
Lista de Exercícios 1

Daniel Bion Barreiros
dbb2@cin.ufpe.br

Os experimentos foram conduzidos para comparar a geração de *pools* homogêneos de classificadores utilizando as técnicas de *ensemble Bagging* e *Random Subspace*. Foram utilizados *Perceptrons* e Árvores de Decisão como classificadores base. As bases de dados escolhidas, "CM1/software defect prediction" e "Class-level data for KC1" do *Promise Software Engineering Repository*, contém informações sobre detecção de falhas em *softwares*. Ambas as bases são binárias, contendo duas classes desbalanceadas.

No processo de geração do *pool*, de tamanho 100, foi utilizado o *10-fold cross-validation* estratificado por classe para dividir o conjunto em 10 partições de treinamento e teste, garantindo que os conjuntos possuem no mínimo uma observação de cada classe. A fim de verificar diferenças no processo de geração, foram treinados 100 classificadores para cada *pool* utilizando as técnicas *Bagging* e *Random Subspace* aplicados aos subconjuntos de 50%, 60%, 70%, 80%, 90% e 100% do conjunto de treinamento.

Cada observação dos conjuntos de teste foi classificada a partir da combinação estática dos classificadores, utilizando o voto majoritário. Em seguida, foi possível calcular as métricas de desempenho para cada classificador (árvore de decisão e *perceptron*) treinado com cada técnica de *ensemble* (*bagging* e *random subspace*) para cada porcentagem do conjunto de treinamento (50% a 100%) de cada *fold,* para ambos os conjuntos de dados, como ilustrado na Figura 1.

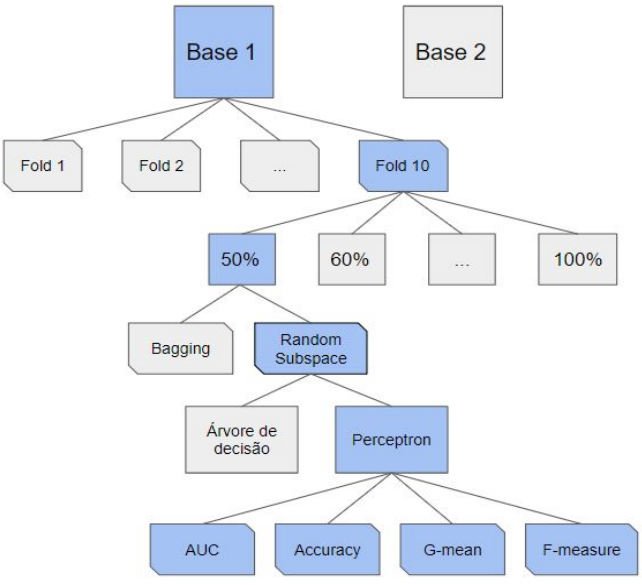

Figura 1. Diagrama da metodologia

Como forma de verificar o desempenho dos classificadores utilizamos algumas métricas como a taxa de acerto, umas das formas mais simples de avaliar o desempenho de um classificador, considerando o valor de *True Positives* (TP) e *True Negatives* (TN). Contudo, para base desbalanceadas, a taxa de acerto pode não ser uma boa medida de desempenho, pois o classificador treinado pode acertar a classe majoritária enquanto erra completamente a classe minoritária, e ainda assim ter um valor alto. Neste tipo de problema podemos utilizar outras medidas como o f-measure e g-mean, que consideram o valor de *True Positives*, *False Positives* (FP) e *False Negatives* (FN), assim como a AUC, que considera também o valor de *True Negatives* (TN).

A partir das métricas de desempenho podemos começar a testar algumas hipóteses dos experimentos. Primeiramente foi testado se os classificadores treinados com subconjuntos de 50% a 100% do conjunto de treinamento apresentam desempenhos diferentes no conjunto de teste. Foi aplicado o teste de Friedman a fim de verificar se algum classificador obteve um resultado muito acima ou abaixo dos demais. O teste foi executado para cada métrica (*accuracy, f-measure, g-mean, auc*) obtida dos *pools* de árvores de decisão e *perceptrons* treinadas com *bagging* e *random subspace*.

$\alpha : 0.05$

H_0 : O desempenho dos classificadores no conjunto de teste se mantém igual ao serem treinados apenas com uma porcentagem do conjunto de treinamento

H_1 : O desempenho dos classificadores são diferentes

Em 90% dos testes executados o p-value foi acima de 0.05, indicando que não temos informações suficientes para rejeitar a hipótese nula nesses casos, ou seja, na maioria dos casos das bases 1 e 2, não existe diferença significativa de desempenho dos classificadores ao serem treinados apenas com uma porcentagem do conjunto de treinamento.

Em seguida verificamos se os classificadores treinados com *bagging* e *random subspace* apresentam desempenhos similares.

$\alpha : 0.05$

H_0 : O desempenho dos classificadores no conjunto de teste se mantém igual ao serem treinados com *bagging* ou *random subspace*

H_1 : O desempenho dos classificadores são diferentes

Apenas em um caso da base 2, a métrica de AUC foi superior no *pool* de *perceptrons* treinado com *bagging*. Nos demais casos, 93% dos testes executados, o p-value foi acima de 0.05, indicando que não temos informações suficientes para rejeitar a hipótese nula, ou seja, na maioria dos casos das bases 1 e 2, não existe diferença significativa de desempenho dos classificadores ao serem treinados com *bagging* ou *random subspace.*

Por fim, testamos o desempenho da árvore de decisão em relação ao *perceptron*.

α : 0.05

H_0 : O desempenho do *perceptron* é igual ao da árvore de decisão
H_1 : O desempenho dos classificadores são diferentes

Os testes de Friedman apontam que as métricas de AUC foram diferentes tanto para a base 1 como para a base 2. Por isso, foi aplicado o teste de Wilcoxon pareado, a fim de verificar qual classificador teve desempenho superior.

α : 0.05

H_0 : P*erceptron + Bagging = Decision Tree + Bagging*
H_1 : P*erceptron + Bagging > Decision Tree + Bagging*

α : 0.05

H_0 : P*erceptron + Random Subspace = Decision Tree + Random Subspace*
H_1 : P*erceptron + Random Subspace > Decision Tree + Random Subspace*

α : 0.05

H_0 : P*erceptron + Bagging = Perceptron + Random Subspace*
H_1 : P*erceptron + Bagging > Perceptron + Random Subspace*

α : 0.05

H_0 : *Decision Tree + Bagging = Decision Tree + Random Subspace*
H_1 : *Decision Tree + Bagging > Decision Tree + Random Subspace*

α : 0.05

H_0 : *Perceptron + Bagging = Decision Tree + Random Subspace*
H_1 : *Perceptron + Bagging > Decision Tree + Random Subspace*

	Base 1		Base 2	
	Bagging	Random Subspace	Bagging	Random Subspace
Perceptron			x	
Árvore de Decisão	x	x		x

Tabela 1. Melhores resultados

Na base 1 o *pool* de árvores de decisão obteve um resultado superior ao *pool* de *perceptrons*, tanto no *pool* treinado com *bagging* quanto no *pool* treinado com *random subspace*. Na base 2 há indícios que o *bagging* funcionou melhor na geração dos *pools* de *perceptrons* enquanto que o *random subspace* funcionou melhor na geração dos *pools* de árvores de decisão como mostra a Tabela 1. Contudo, os *pools* de *perceptrons* treinados com *bagging* tiveram performance similares aos *pools* de árvores de decisão treinadas com *random subspace*.

Referências

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

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