Proposta de Projeto - Computação Natural

Matheus Cândido Teixeira

2 de outubro de 2020

1 Introdução

Muitos classificadores em *Machine Learning* (ML) podem ser classificados em duas categorias: classificadores *multi-class* (MCC) ou *one-class* (OCC). Os MCC, que pertencem a primeira categoria, são utilizados quando há amostras suficientes de todas as classes na qual o método deve ser capaz de classificar. Porém, há situações em que não é possível acumular uma quantidade estatísticamente significativa de exemplos de todas as classes para fazer treinamento dos algoritmos [2, 3]. Para situações tais como estas, o OCC pode ser mais adequado, pois ele constroi um modelo preditivo a partir de exemplos de uma única classe.

O OCC diverge do MCC pelo fato de ele ser treinado para apenas uma classe. A classe de treinamento é denominada como positivas ou classe alvo e as demais amostras que não possuem uma classificação (ou *labels*) são denominadas negativas ou *aliens* [3, 6]. Atualmente, o OCC vem sendo aplicado a muitas áreas, como visão computacional, detecção de anomalias e processamento de imagens médicas, onde é comum haver mais amostras do comportamento normal do que de anomalias. Essa característica torna o OCC muito atrativo para diversas aplicações, porém um fator que afeta a performance dele é o excesso de *features* que trazem pouca informação ou redundância.

Ao construir um classificador é necessário treiná-lo com amostras das diversas classes. A capacidade de classificação está relacionada com capacidade de distinguir as diferenças entre os dados de cada classes. Os dados de entrada, denominados *features*, determinam qual classe a amostra pertence. Portanto, é comum encontrar *datasets* com inumeras *features* para que o classificador possa extrair algum padrão dos dados e, assim, inferir a classe da amostra de entrada.

Um problema associado ao excesso de *features* é a presença de *features* que não contribuem para a classificação. Isso pode ocorrer de duas maneiras: (1) algumas *features* do conjunto possuem muita correlação entre si ou (2) as *features* não possem nenhuma correlação com as classes de saída. O primeiro cenário ocorre pois uma váriavel é suficiente para descrever uma característica e as demais se tornam redundantes e passam a afetar negativamente a performance do classificador, pois acrescenta dados que não trazem ganho para a predição. O segundo cenário ocorre pois os excesso de features se comporta como ruído nos dados causando *bias* e reduzindo a performance do classificador [1, 5, 4, 7].

Para os algoritmos *multi-class* a precisão não é significativamente afetada pelo excesso de *features*, porém, para os algoritmos OCC, o efeito pode ser significativo [6]. Para resolver esse problema, um método utilizado é o Feature Selection (FS). O FS busca reduzir o espaço dimensional do conjunto de *features* para um subconjunto menor e ainda capaz de descrever os dados de entrada de modo eficiente, isso é alcançado pela remoção de *features* que trazem pouca informação relevante para a classificação ou de *features* reduntantes [1]. Alguns métodos de FS e as suas características são apresentadas na Seção 3.

O FS tem sido usado como uma forma de aumentar a performance do OCC. Ele é usado para encontrar um subconjunto *optimal* de *features* para que o OCC obtenha uma performance comparável com o dos MCCs. Diversos algoritmos têm sido propostos, como *Random Feature Selection* (RFS), *Selecting Features from Clusters* (SFC), *Selecting the Highest Information Gain from Feature Clusters* (HIC) [2], porém apenas um, SFC, foi capaz de gerar um selecionar as *features* de modo que o OCC obtivesse um desempenho próximo ao dos MCC. Portanto, o objetivo deste trabalho é utilizar o algoritmo genético (GA) como uma heurística para encontrar subconjunto *suboptimal* de *features* que maximizem a performance do OCC e comparar a performance obtida dele com classificadores multi-classes. O uso do GA para FS já foi feito em outros trabalhos, como em [1]. A representação de cada indivíduo é da seguinte maneira. Um cromossomos representa um conjunto de *features*, cada locus representa uma *feature* e o gene {0, 1} representa a presença ou a ausência da *feature*. Desse modo, neste trabalho faz a análise do desempenho do classificar com o FS.

Este trabalho é dividido da seguinte maneira: A Seção 2 apresenta trabalhos relacionados e aplicações do FS em OCC. Na Seção 3 é apresentado o estado-da-arte tanto dos algoritmos FS como dos algoritmos OCC. Na Seção 4 é apresentado a forma como a pesquisa foi desenvolvida. Na Seção 5, os resultados da pesquisa são apresentados. Por fim, na Seção 6, os resultados da seção anterior são discutidos, e, também, é apresentado as conclusões obtidas nessa pesquisa.

2 Trabalhos relacionados

Khalifa et al. [2] utilizam OCC para detecção de pre-miRNA. Os autores identificam que o método mais comum na literatura era utilizar TCC para a classificação. Um problema que havia nessa aplicação era a falta de exemplos negativos para o treinamento. A solução utilizada na literatura era selecionar partes aleatórias consideradas não-miRNA ou a partir de sequências geradas aleatoriamente. Portanto, os autores sugeriram o uso de OCC para contornar a falta de exemplos negativos. Além disso, eles também consideraram o impacto da quantidade de features (mais de 1000) na performance do OCC. Assim, eles testram diferentes algoritmos de FS e compararam o efeito na performance da classificação feita por algoritmos TCC (o algoritmo utilizado foi o SVM)e OCC. Os resultados indicaram que o impacto do FS é muito mais significativo para o OCC do que para os TCCs e que, em geral, os TCC possuem melhor desempenho do que os OCC, porém com a escolha de um FS adequado, a performance do OCC se aproxima dos TCC. O algoritmo FS que apresentou melhor resultado foi o *Selecting Features from Cluster* (SFC), onde tanto o TCC quanto o OCC atingiram mais de 95% de precisão.

3 Revisão da literatura

Nesta seção é apresentado o estado-da-arte tanto para os algoritmos FS como para os OCC. Os trabalhos são classificados em duas seções: uma para abordar os algorimos FS e outra para os OCC.

3.1 *Feature Selection* (FS)

Chandrashekar and Sahin [1] apresentam três tipos métodos para FS: *Filter Methods* (FM), *Wrapper Methods* (WM) e *Embedded Methods* (EM). Os FM utilizam técnicas de ranking para ordenar as variáveis para seleção. Uma critério de ranking é utilizado para pontuar as features e remover as que estão abaixo de um threshold. Os métodos de ranking (MR) são aplicados antes da classificação para remover variáveis pouco relevantes (baseado na pontuação obtida).

Uma vantagem dos MR é que eles são computacionalmente leves e evitam *overfitting*. Uma desvantagem dos MR é que o subconjunto gerado pode não ser *optimal* no sentido que pode haver features redundades entre si. O WM utiliza um preditor (algoritmo de classificação) como um sub-elemento do método. Este método seleciona um subconjunto de features e aplica um algoritmo de classificação tendo esse subconjunto como entreada. O desempenho do classificador é avaliado com base na performance que ele obteu para aquele subconjunto de entrada. O objetivo é determinar qual subconjunto de features maximiza o desempenho do classificador. O total de subconjuntos possíveis é 2^k, onde k é a quantidade de features presentes no dataset. Devido a essa natureza exponencial, este é um problema NP-hard, por isso, algoritmos de busca (sequencial ou heurística) são empregados para encontrar um subconjunto de features de modo mais eficiente. Por fim, o EM busca contornar o problema NP-hard do método WM. Nesse método ao invés do preditor ser um subelemento do FS, o FS é inserido no classificador durante a fase de treinamento. Esse método utiliza uma estatística denominada mutual information (MI), que é calculada como a diferença: $I(Y, X) = H(Y) - H(Y|X)$, onde $H(\cdot)$ é a medida de entropia de Shannon e $H(\cdot|X)$ é a entropia condicional[1]. O MI é empregado de forma que ele é maximizado entre a saída, Y, e uma feature de entrada X_i e mínimizada entre features de entrada. A consequência disso é que são apenas mantidas as features que introduzem o máximo de informação para a predição, enquanto não possuem redundância entre si.

3.2 *One Class Classifiers* (OCC)

Khan and Madden [3] apresentam uma pesquisa sobre os OCCs. Nesse trabalho, os autores criaram uma taxonomia para o estudo de problemas OCCs. A taxonomia proposta é dividida em três categorias: dados disponíveis para treinamento, metodologia usada e o domínio de aplicação. A primeira categoria analisa qual a forma dos dados utilizados para treinamento, isto é, se apenas com dados positivos, com dados positivos e algumas poucas amostras de dados negativos (ou até mesmo artifialmente gerados) ou com dados positivos e amostras não-rotuladas. No que tange a metodologia de classificação, os autores identificam duas amplas categorias: One-Class Support Vector Machines (OSVM) e não-OSVMs. A última categoria diz respeito a aplicação que o algoritmo OCC é aplicado: classificação de textual ou outra aplicação.

4 Materiais e Métodos

4.1 Dataset

A seguir é apresentado como o FS foi implementado utilizando uma variante do GA, denominado *Constraint Handling Coevolutionary Genetic Algorithm* (CHCGA). A primeira seção apresenta os detalhes da representação dos indivíduos e em seguida é apresentado os demais operadores genéticos, como o operador de seleção, de mutação e de *crossover*.

4.2 Representação dos invíduos em CHCGA

O dataset utilizado neste trabalho contém 31 *features* extraídas de um conjunto de usuários que pertencem a 3 classes distintas: *humans*, *bots* e *news*. Portanto, para representar um indivíduo, um vetor com 31 elementos é utilizado. Cada elemento do vetor pode assumir um valor binário, onde 1 indica que a *feature* pertence ao subconjunto de *features* utilizadas no treinamento do classificador e 0 indica a ausência.

[1] A formulação detalhada pode ser encontrada em [1]

4.3 Operadores evolucionários

O primeiro operador a ser definido é o operador de *fitness*, que é responsável por mensurar a qualidade do indivíduo. Neste trabalho, a *fitness* de um indíviduo é igual a *Area Under the Curve* (AUC) da *Receiver Operating Characteristic* (ROC). No CHCGA, um *Highly Disruptive Half Uniform Crossover* (HUX) é usado como operador de *crossover*, o mecânismo desse operador consiste em selecionar dois indivíduos na população (sem reposição) e avaliar se a distância de Hamming entre eles é maior que um *threshold* pré-definido, caso a distância exceda esse *threshold*, metade dos alelos diferentes entre eles são alterados. Geralmente, esse *threshold* é definido como $L/4$, onde L é o tamanho do indivíduo e é decrementado caso não haja decendentes suficientes. No caso de o *threshold* chegar a 0 e não houver uma nova geração geradas, o melhor indivíduo da população é tomado como template e $N-1$ indivíduos são gerados aplicando-se o operador de mutação com probabilidade entre 35-40%, onde N é o tamanho da população. O Algoritmo **??** sumariza o algoritmo.

Com esse operadores definidos, o *wrapper* FS é adotado para obter o subconjunto de *features* que maximizem o valor da AUC do classificador *One-Class*. O funcionamento da estrutura é da seguinte maneira: uma população inicial é gerada contendo P_0 indivíduos. Cada indivíduo possui 31 genes (cada um representando uma *feature*). Quando o valor do gene for 1, a *feature* representada por esse gene é incluida no treinamento do classificador. Para o cálculo da *fitness*, 50% do dataset é utilizado como treinamento e o retante para teste. O valor da AUC resultante da predição do classificador após o treinamento é o valor da *fitness*. Portanto, esse é um problema de maximização.

5 Resultados

6 Discussão e Conclusão

Referências

[1] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers and Electrical Engineering*, 40(1):16–28, 2014. ISSN 00457906. doi: 10.1016/j.compeleceng.2013.11.024. URL `http://dx.doi.org/10.1016/j.compeleceng.2013.11.024`.

[2] Waleed Khalifa, Malik Yousef, Müşerref Duygu Saçar Demirci, and Jens Allmer. The impact of feature selection on one and two-class classification performance for plant microRNAs. *PeerJ*, 2016(6):1–13, 2016. ISSN 21678359. doi: 10.7717/peerj.2135.

[3] Shehroz S. Khan and Michael G. Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3): 345–374, jun 2014. ISSN 0269-8889. doi: 10.1017/S026988891300043X. URL `http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Technical+Forum+Group+on+Agents+in+Bioinformatics+1{#}0https://www.cambridge.org/core/product/identifier/S026988891300043X/type/journal{_}article`.

[4] Huan Liu and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Springer US, Boston, MA, 1998. ISBN 978-1-4613-7604-0. doi: 10.1007/978-1-4615-5689-3. URL `http://link.springer.com/10.1007/978-1-4615-5689-3`.

[5] Jianyu Miao and Lingfeng Niu. A Survey on Feature Selection. *Procedia Computer Science*,

91(Itqm):919–926, 2016. ISSN 18770509. doi: 10.1016/j.procs.2016.07.111. URL http://dx.doi.org/10.1016/j.procs.2016.07.111.

[6] Pramuditha Perera and Vishal M. Patel. Learning Deep Features for One-Class Classification. *IEEE Transactions on Image Processing*, 28(11):5450–5463, 2019. ISSN 19410042. doi: 10.1109/TIP.2019.2917862.

[7] Wei Zheng, Xiaofeng Zhu, Guoqiu Wen, Yonghua Zhu, Hao Yu, and Jiangzhang Gan. Unsupervised feature selection by self-paced learning regularization. *Pattern Recognition Letters*, 132:4–11, 2020. ISSN 01678655. doi: 10.1016/j.patrec.2018.06.029. URL https://doi.org/10.1016/j.patrec.2018.06.029.