Árvore de Decisão para Classificação de Grafos de Moléculas Químicas Inibidoras do HIV Utilizando *node2vec*

Matheus Santos Araújo

[1]Centro de Ciências e Tecnologia – Universidade Estadual do Ceara (UECE)
Avenida Silas Munguba – Fortaleza – CE – Brazil

Resumo. *O conjunto de dados ogbg-molhiv é utilizado para a predição de propriedades moleculares. Todas as moléculas são pré-processadas para apresentarem um formalismo de grafos, onde cada grafo representa uma molécula, onde os nós são átomos e as arestas são ligações químicas. Logo, a partir disso é possível classificar a molécula em inibidora ou não do vírus HIV. Dado isso, este trabalho apresenta uma solução para a realização da classificação utilizando árvores de decisão a partir da geração de uma projeção de cada grafo dadas as projeções de seus nós utilizando node2vec.*

1. Introdução

O conjunto de dados *ogbg-molhiv* apresentado em [Wu et al. 2018] e também avaliado em [Hu et al. 2020a] é utilizado para a predição de propriedades moleculares. Todas as moléculas são pré-processadas para apresentarem um formalismo de grafos, onde cada grafo representa uma molécula, onde os nós são átomos e as arestas são ligações químicas. As *features* de nós são 9-dimensionais, contendo características como número atômico e quiralidade, bem como outras característica adicionais do átomo, como carga formal e se o átomo está em um anel ou não. As *features* de aresta são tridimensionais, contendo o tipo de ligação, estereoquímica da ligação, bem como uma *feature* adicional que indica se a ligação é conjugada. Logo, a partir disso é possível classificar a molécula em inibidora ou não do vírus HIV. Entretanto, as *features* adicionais de nós e arestas não são orbigatórias para identificar corretamente moléculas e não foram adotados em trabalhos anteriores como [Hu et al. 2020b] e [Ishiguro et al. 2019]. Porém, em [Hu et al. 2020a] as *features* foram levadas em consideração em alguns cenários e resultaram em melhoria da acurácia. Nesse sentido, este trabalho apresenta uma solução alternativa para a realização da classificação utilizando árvores de decisão a partir da geração de uma projeção de cada grafo dadas as projeções de seus nós utilizando o *node2vec*, os resultados serão avaliados a partir da métrica ROCAUC e comparados a outras soluções na literatura.

2. Fundamentação Teórica e Revisão da Literatura

2.1. *ogbg-molhiv*

A origem do conjunto de dados de HIV utilizado neste trabalho foi a *AIDS Antiviral Screen do Drug Therapeutics Program (DTP)*, que testou a capacidade de inibir a replicação do HIV para mais de 40.000 compostos. Os resultados da triagem foram avaliados e colocados inicialmente em três categorias: confirmado inativo (IC), confirmado ativo (CA) e confirmado moderadamente ativo (MC). Além disso, em [Wu et al. 2018] os dois últimos rótulos foram combinados para a definição atual e utilizada neste trabalho onde temos a classificação entre inativo (CI) e ativo (CA e CM).

2.2. *node2vec*

Aprender representações úteis de objetos estruturados como grafos é útil, para uma variedade de aplicativos de aprendizado de máquina. A representação e aprendizado de grafos visa aprender projeções para os nós ou arestas do grafo, que podem ser usados para uma variedade de tarefas de aprendizado de máquina, como predição da classe de um de nó por exemplo, categorizar uma molécula com base em sua estrutura e previsão de conexões em uma rede social. As projeções de nós em um grafo são passíveis, como visto neste trabalho, de representar a projeção do próprio grafo a partir de sua soma ou média. Outra estratégia é o uso de um nó virtual.

Dada essa tarefa de projeção, o node2vec descrito em [Grover and Leskovec 2016] é uma técnica simples, mas escalonável e eficaz para aprender projeções de baixa dimensão para nós em um grafos, otimizando um objetivo de preservação de vizinhança. O objetivo é aprender projeções semelhantes para nós vizinhos, no que diz respeito à estrutura do grafo.

Simplificadamente, o *node2vec* funciona da seguinte maneira:

1. Gerar sequências de nós usando passeio aleatório (tendencioso).
2. Criar exemplos de treinamento positivos e negativos a partir dessas sequências.
3. Treinar um modelo *word2vec* (*skip-gram*) para aprender as projeções dos nós.

2.3. Árvore de Decisão

Uma árvore de decisão é um mapa dos possíveis resultados de uma série de escolhas relacionadas. Podem ser usadas tanto para conduzir diálogos informais quanto para mapear um algoritmo que prevê a melhor escolha, matematicamente. Uma árvore de decisão geralmente começa com um único nó, que se divide em possíveis resultados. Cada um desses resultados leva a nós adicionais, que se ramificam em outras possibilidades. Assim, cria-se uma forma de árvore. Existem três tipos de nós: nós de probabilidade, nós de decisão e nós de término. Um nó de probabilidade, representado por um círculo, mostra as probabilidades de certos resultados. Um nó de decisão, por sua vez, representado por um quadrado, mostra uma decisão a ser tomada, e um nó de término mostra o resultado final de um caminho de decisão.

3. Metodologia e Resultados

3.1. Conjunto de Dados

A linguagem utlizada para a implementação dos experimentos será a linguagem de programação Python [Van Rossum and Drake 2009], pois a linguagem oferece as ferramentas necessárias de apoio ao carregamento do conjunto de dados e a implementação tanto dos algoritmos em grafos quanto de aprendizagem de máquina. Primeiro, carregamos a base de dados contendo 41.127 grafos moleculares, com rótulos de caso negativo (0) e positivo (1). Especificamente, no conjunto de dados *ogbg-molhiv* os autores recomendam o uso da métrica ROC-AUC para a avaliação. Pois no *ogbg-molhiv*, como o equilíbrio das classes é extremamente distorcido (apenas 1,4% dos dados são positivos), os dados precisam utilizar a Precisão Média calculada como métrica de avaliação. Para a validação cruzada a seguinte divisão foi estabelecida por sugestão dos autores da base de dados em [Hu et al. 2020a]:

Table 1. Detalhes dos Experimentos.

Conjunto	Porcentagem
Dados para Treino	80%
Dados para validação	10%
Dados para Teste	10%

3.2. Grafos

A primeira etapa da implementação consistiu na conversão do conjunto de dados para um formato mais conveniente, no caso o formato da biblioteca *NetworkX*, o grafo da primeira molécula do conjunto de treino pode ser visto na figura 1. onde pode ser vista a presença de 24 nós representando os átomos da molécula e 25 arestas representando, por sua vez, as ligações químicas entre os átomos.

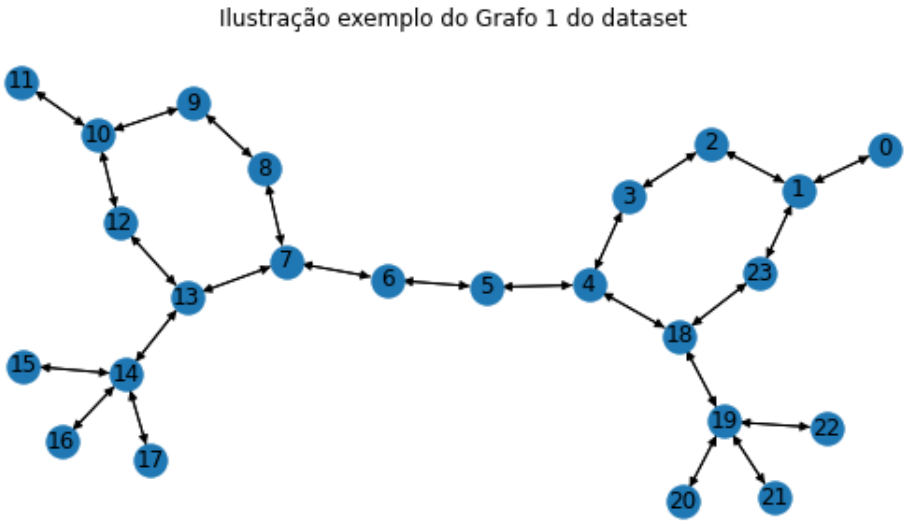

Figure 1. Exemplo de grafo do conjunto de dados.

3.3. Passeios Aleatórios e *node2vec*

O passeio aleatório enviesado, necessário para a execução do *node2vec*, foi implementado com a estratégia de realizarmos um passeio por nó do grafo em questão com um tamanho fixado em 15. Os valores que p e q, foram definidos como 0.15 e 7, respectivamente. Um valor alto de p significa que temos menos probabilidade de retornar ao nó anterior e um valor alto de q aproxima a amplitude da primeira pesquisa, o que significa que a vizinhança ao redor do nó é explorada. Valores baixos dão maior chance de sair da vizinhança e, portanto, aproximam a profundidade da primeira pesquisa.

Portanto, em seguida, para obter as projeções precisas usando os passeios aleatórios gerados, utilizamos uma ferramenta advinda do processemento de linguagem natural, o algoritmo *Word2Vec*. Em particular, vamos usar o modelo *skip-gram* com uma

camada *softmax* hierárquica. A ideia principal do modelo skip-gram é prever o contexto de uma sequência de um nó particular. Por exemplo, se quisermos treinar uma projeção de um determinado nó, treinaremos nosso modelo (rede neural) com o objetivo de prever os nós que aparecem em seus passeios aleatórios. Portanto, a entrada do modelo será o nó, a camada intermediária será a projeção real e a saída será a previsão do contexto do nó, de forma simplificada.

Para execução do *Word2Vec* definimos alguns parâmetros como a quantidade de *threads* na execução para acelerar o treinamento fixada em 8, dimensionalidade dos vetores de nós fixada em 64 (tamanho da projeção), $sg = 1$ indicando que a rede neural skip-gram será utilizada e $hs = 1$ indicando que uma camada *softmax* hierárquica será usada. o *mincount* define um limite inferior ao tamanho da sequência. Por fim, o parâmetro *Window* define a distância máxima entre o nó atual e o previsto em uma sequência e foi fixado em 10. Os parâmetros restantes foram definidos como padrão da biblioteca utilizada. **Todos os parâmetros de passeio aleatório e *node2vec* em nós foram escolhidos a partir de sucessivos testes manuais em busca da configuração de melhor resultado.**

A projeção do grafo considerou a média das projeções utilizando *node2vec* de todos os nós do grafo. Para assim, servir de entrada para o classificador utilizado na etapa de treinamento.

3.4. Treinamento

O modelo de classificação é treinado para o conjunto de dados *ogbg-molhive*, onde possuimos o vetor de 41.127 rótulos indicando se a molécula representada pelo grafo inibe ou não o vírus HIV. No caso da entrada X do conjunto de dados, foram utilizadas as saídas da projeção dos grafos, as *features* de nós e arestas não foram consideradas. A divisão conforme detalhado anteriormente na tebal 1 consistiu em 80% dos grafos para o conjunto de treino, 10% para o conjunto de validação e 10% para o conjunto de testes. O classificador utilizado foi a árvore de decisão, a sua configuração de hiperparâmetros utilizado foi a padrão da biblioteca *Scikit-learn*.

3.5. Resultados

Para execução dos testes do problema utilizamos a linguagem python em um ambiente Linux controlado com a distribuição Ubuntu versão 16.04 LST em um computador com 4GB de RAM e processador i5 onde foram implementados utilizando os algoritmos das bibliotecas *Torch*, *Networkx*, *Gensim* e *Scikit-learn*. Para os experimentos descritos no trabalho, utilizamos o servidor e ambiente *LASIDHUB* disponível pelo Laboratório de Sistemas Digitais (LASID) da Univesidade Estadual do Ceará em Fortaleza.

Table 2. Resultados dos Experimentos

Conjunto	**Métrica ROCAUC**
Dados para Treino	100%
Dados para validação	54.61%
Dados para Teste	51.02%

Os resultados obtidos pelo conjunto de dados com as entradas consistindo nas projeções foram promissores. O ROCAUC de treino foi de 99% ao classificar o grafo

da molécula em inibibidor ou não do HIV, seguindo de 50% de ROCAUC no conjunto de validação e 50% no conjunto de testes. O resumo dos resultados é apresentado na tabela 2. A matriz confusão calculada a partir das classificações no conjunto de testes é apresentada na tabela 3.

Table 3. Matriz de Confusão do Conjunto de Testes

		Classes Verdadeiras	
		Positivo	Negativo
Classes Preditas	Positivo	9	121
	Negativo	221	3789

4. Conclusão

Este trabalho apresentou uma solução para a realização da classificação a partir da geração de uma projeção de cada grafo dadas as projeções de seus nós utilizando *node2vec*. Os resultados mostraram que, para o conjunto de dados de treino a acurácia foi próxima de 100%, enquanto para os conjuntos de validação obteve resultados razoáveis.

Em resumo, os resultados foram promissores, apresentando um estudo empírico acerca da utilização da projeção de grafos a partir da projeção de nós usando *node2vec* para classificação de grafos. Demonstrou-se que, apesar de utilizar um classificador simples, o algoritmo obteve desempenho relevante ao classificar os grafos de moléculas em inibidoras ou não de HIV.

References

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. (2020a). Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*.

Hu, Z., Dong, Y., Wang, K., and Sun, Y. (2020b). Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pages 2704–2710.

Ishiguro, K., Maeda, S.-i., and Koyama, M. (2019). Graph warp module: an auxiliary module for boosting the power of graph neural networks.

Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. (2018). Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530.