Interpretabilidade em modelos DL na classificação de compostos químicos

Bernard Ambrosio Georges
Mestrado em Engenharia Informática
Braga, Portugal
pg53698@alunos.uminho.pt

João Paulo Peixoto Castro
Mestrado em Engenharia Informática
Braga, Portugal
pg53929@alunos.uminho.pt

Universide do Minho - Campus de Gualtar, Braga, Portugal

Repositório do Projeto: https://github.com/BernardGeorges/ProjetoAP

Abstract. Este artigo expõe a exploração de modelos de deep learning para prever a capacidade de interação entre compostos químicos e a proteína p53. Na primeira parte foi feita uma análise completa de modelos de previsão e interpretabilidade de compostos químicos. Assim, este relatório tem como objetivo visualizar as diferentes combinações possíveis e as suas vantagens e desvantagens quanto a sua previsão e interpretabilidade.

1 Introdução

Na área de *Deep Learning* existe uma grande dificuldade na compreensão de como as redes neuronais efetuam as suas decisões (problema da "black box"). Isto deve-se a variados fatores, como a interação complexa e não-linear entre os parâmetros, a pouca compreensão relativamente às funções de ativação aplicadas ao longo da rede, à existência de ferramentas de interpretabilidade ainda pouco completas e eficazes.

No contexto molecular, inúmeras são as companhias que pretendem aplicar modelos de deep learning para a previsão rápida e eficaz das propriedades de moléculas em áreas como culinária, medicina, farmacêutica, etc. Nestes e em outros casos, os métodos de interpretabilidade são importantes para oferecer uma maior transparência e confiança quanto às decisões do modelo.

2 Objetivos

O problema irá consistir na classificação de compostos químicos baseando-se na sua capacidade de interação com a proteína p53 (elemento importante na prevenção do desenvolvimento de tumores). Deste modo, estabelecemos como objetivos deste projeto adaptar um modelo Graph Neural Network e implementar estratégias que nos forneçam explicações locais e globais das previsões feitas para cada uma das moléculas. Por outro lado, pretendemos utilizar a framework *deepmol* como *baseline* e implementar técnicas de *feature importance* complementares às anteriormente implementadas.

3 Processamento dos dados

Como ponto de partida foi escolhido o dataset TOX21, que inclui um conjunto de compostos químicos em notação SMILES, e atributos referentes a propriedades químicas dos mesmos (NR-AR, NR-ER, NR-AhR, NR-PPAR-gamma, NR-Aromatase, SR-ARE, SR-p53, que se trata da label). A notação SMILES consiste numa representação química simplificada que descreve as estruturas moleculares usando uma sequência de caracteres alfanuméricos.

No que toca ao tratamento de dados foi feita uma substituição dos valores NA referentes aos *missing values*, foi removida a coluna mol_id que para o contexto deste problema era irrelevante. Foi feita uma transformação da coluna 'smiles', convertendo cada um dos valores em formato String para um objeto *rdkit.Chem.rdchem.Mol*. O balanceamento deste dataset consiste num racio 0.23/0.77, em que 1480 compostos são reativos à SR-p53, enquanto que 6351 dos compostos químicos não são reativos à SR-p53. Esta diferença cria um claro bias para o lado dos compostos reativos visto que o modelo terá uma accuracy de 77% adivinhando apenas positivo. Por isso foi necessário alterar o dataset de modo que não haja uma discrepância tão grande. Com esse objetivo, foi removido alguns registos de modo que a diferença seja aceitável.

O próximo passo do processamento de dados consistiu na conversão de cada uma das moléculas que estavam no formato de um objeto rdkit.Chem.rdchem.Mol e transformá-las para uma instância do torch_geometric.data.Data do Pytorch Geometric. Como o rdkit possui uma api que permite obter uma listagem dos átomos, ligações das moléculas, e características associadas como número atómico, se tem uma estrutura em anel, entre outras. Deste modo, foram feitas travessias nessas estruturas para criar vetores de índices com essas propriedades e depois esses vetores são convertidos para tensores do Pytorch. Assim, para cada molécula foram criados quatro tensores (um correspondente às propriedades dos átomos, um correspondente aos índices das ligações, outro das características das ligações e por fim um relativo às características extraídas do dataset. Estes tensores são essenciais para criarmos uma instância Data do torch_geometric.data, que consiste na representação de um grafo homogéneo. Desta forma, a representação em grafo segue uma estrutura muito semelhante à estrutura molecular em que os edges correspondem a ligações e os nodes a átomos.

Por fim, como último passo foi aplicada esta função a todos os elementos da coluna 'smiles'.

No que toca à visualização de dados, o rdkit permite uma visualização em duas dimensões das moléculas. Por outro lado, também foi possível visualizar aos grafos de cada uma das batches recorrendo às bibliotecas Explainer e CaptumExplainer.

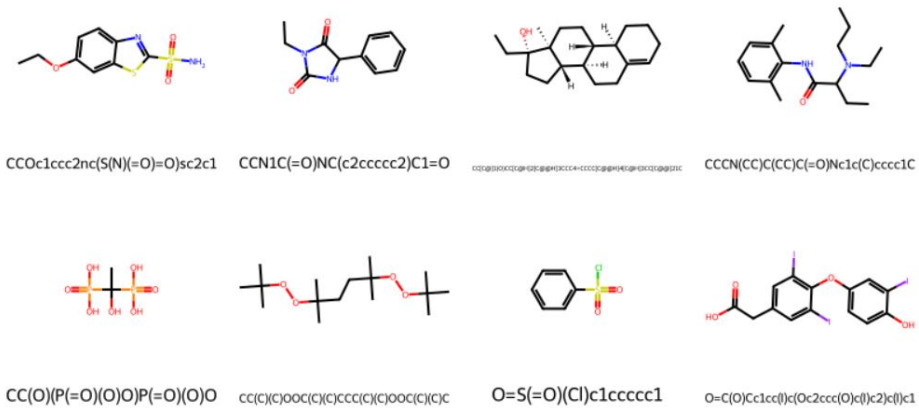

Fig. 1. Visualização estrutural dos compostos químicos.

4 Modelo de Deep Learning- GCN

No que toca à arquitetura do modelo Graph Neural Network construído foram utilizadas 3 camadas convolucionais GCNConv, uma operação de global_mean_pool (pooling global da média das características), e uma camada Linear (aplicação de uma transformação linear aos dados de entrada). Para a aplicação deste modelo foram fornecidos o número de features de input, o número de classes de output, e o número de in e out channels para as camadas intermédias.

```
class GCNModel(torch.nn.Module):
    def __init__(self):
        super(GCNModel, self).__init__()
        torch.manual_seed(12)
        self.input = None
        self.my_grads = None
        self.final_conv = None
        self.conv1 = GCNConv(n_features, 8)
        self.conv2 = GCNConv(8, 16)
        self.conv3 = GCNConv(16, 32)
        self.pooling = global_mean_pool
        self.linear = Linear(32, 2)
```

Fig. 2. Camadas da arquitetura.

No método de propagação foi aplicada a função de ativação ReLU entre as camadas convolucionais. Nesse método foi também registado um hook para proceder à recolha dos gradientes. O hook basicamente irá permitir-nos executar a função na qual durante a aprendizagem serão capturados os gradientes dos

parâmetros do modelo. Os gradientes irão ser importantes para averiguarmos quais são as características que são mais relevantes para a previsão final do modelo.

```python
# registar os gradientes em my_grads
def activations_hook(self, grads):
    self.my_grads = grads

def forward(self, x, edge_index, edge_attr, batch):

    self.input = x

    x = self.conv1(x, edge_index,edge_attr)
    x = F.relu(x)
    x = self.conv2(x, edge_index,edge_attr)
    x = F.relu(x)

    with torch.enable_grad():
        self.final_conv  = self.conv3(x,edge_index,edge_attr)

    # registar hook para a layer final conv3
    self.final_conv.register_hook(self.activations_hook)

    x = self.pooling(self.final_conv, batch=batch)

    x = self.linear(x)
    return x
```

Fig. 3. Recolha de gradientes

De seguida, procedeu-se ao treino do modelo, sendo que o conjunto de dados foi dividido num conjunto de treino e teste e agrupado por batches de tamanho 32. Após a fase de treino, podemos verificar que o modelo atingiu uma accuracy a rondar os 82% ao fim de 150 epochs.

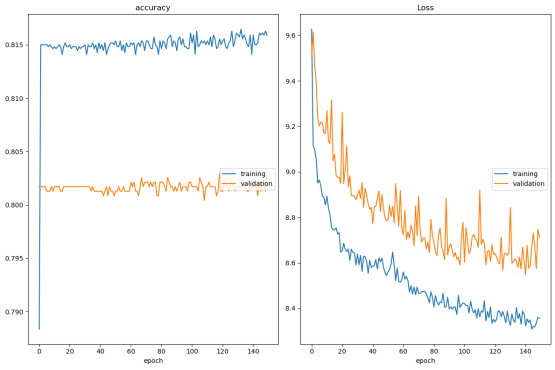

Fig. 4. Métricas de avaliação

5 Métodos de interpretabilidade

A primeira abordagem passou por calcular a importância de cada uma das características de entrada para a previsão do modelo. Deste modo, procedeu-se

à acumulação dos gradientes absolutos das características (feature_importance += torch.abs(inputs.grad).sum(dim=0)), para que desta forma fosse possível averiguar quanto cada característica contribuiu para a variação na perda. Esses valores por fim foram normalizados e ordenados por ordem descendente.

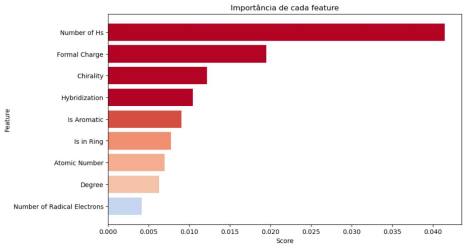

Fig. 5. Feature importance

Foi também implementada uma técnica de interpretabilidade para entender quais eram os nodos mais relevantes para a decisão do modelo. Para isso obtiveram-se os gradientes das características de entrada x armazenadas em gcn_model.input durante a backpropagation. De seguida, iterou-se por cada conjunto de gradientes correspondentes do atom_grad = atom_gradients[i, :] e calculou-se a saliência do nodo usando a norma dos gradientes. Por fim, esses valores são armazenados numa lista e é calculada a média dos gradientes de cada átomo. Após isso procedeu-se à invocação de Draw.MolToImage(m, kekulize=True, wedgeBonds=True, highlightAtoms=weights).

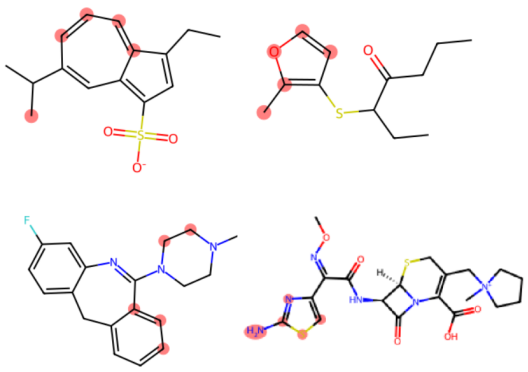

Fig. 6. Saliency map of atoms importance

Outra abordagem seguida foi proceder ao cálculo dos scores normalizados para os atributos das arestas com base nos gradientes das mesmas. Deste modo, é feito o cálculo da norma dos gradientes e a normalização desses scores. Após isso criámos um dicionário onde as keys correspondem aos índices das ligações e os valores às cores mapeadas dos scores das ligações usando o matplotlib.pyplot.get_cmap. Assim, as ligações mais importantes vão aparecer com uma tonalidade cada vez mais vermelha, dentro de um intervalo que vai de azul-escuro a vermelho-escuro.

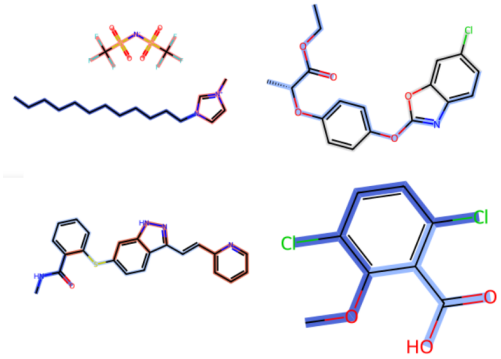

Fig. 7. Saliency map of molecule bonds importance

Por outro lado, utilizaram-se os submódulos Explainer e CaptumExplainer da biblioteca torch_geometric.explain que nos permite analisar estruturas de subgrafos e node features importantes para cada um dos batches na previsão do modelo. Para além disso, é possível obter uma representação em grafo dos mesmos.

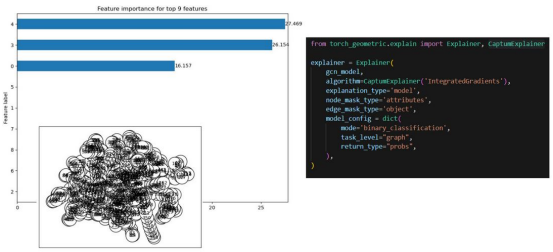

Fig. 8. Explainability for node classification on graph using Explainer and Captum-Explainer

6 Framework deepmol

De modo a comparar o modelo criado com outros modelos previamente testados foi utilizado as bibliotecas deepmol e deepchem, sendo que ambas são frameworks utilizadas para a criação de modelos de deep learning com objetivos na area de moléculas, como é o nosso caso. Para manter este estudo o mais justo o possível, foi feito um pré-processamento semelhante a primeira parte do método usado acima.

6.1 Random Forest

Primeiramente, foi criado um modelo de Random Forest. Este foi criado com expectativas baixas e como um modelo base, porém foi o modelo com melhor resultados quanto a sua accuracy e loss. Quanto a sua interpretabilidade este permitiu a visualização mais extensa sendo possível não só as ligações importantes como os restantes modelos mas também foi possível visualizar a influencia dos outros parâmetros através dos grafos abaixo apresentados.

É importante notar que para este modelo não foi feita nenhuma transformação de dados, como é o caso para os outros modelos, onde é necessário transformar os smiles em grafos. Está foi a única diferença em pré-processamento para os outros modelos.

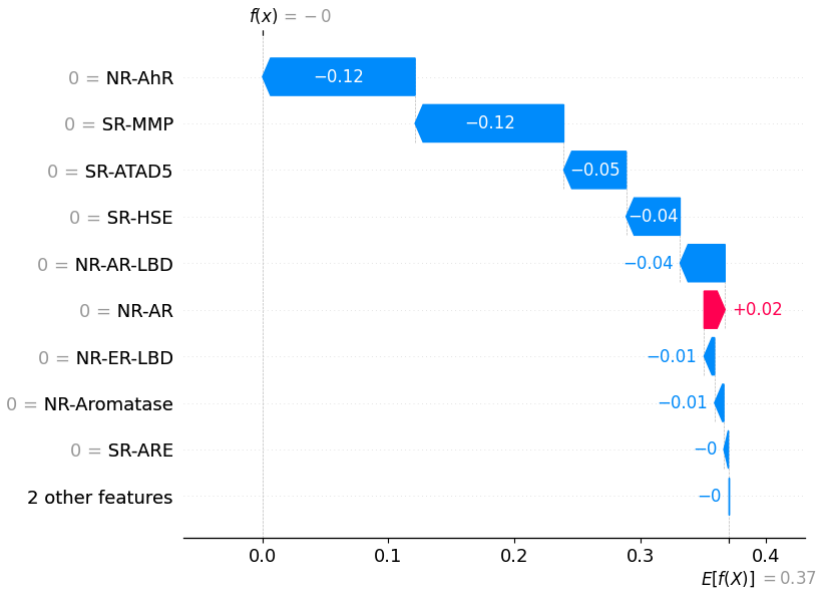

Fig. 9. Waterfall Plot

Este grafo apresenta a influencia de cada valor em uma instância especifica do dataset. O modelo começa com um valor base e cada valor apresentado no grafo teve o influencia a decisão final deste caso consoante seu peso. Os valores negativos, apresentados a azul, influencia a decisão a ser positiva e os valor a vermelho representa o incentivo a ser falso. É notável que o valor base inicial é 0 e que o resultado final foi 0.37, ou seja, este exemplo muito provavelmente teve a sua previsão como falso. Com está informação é possível concluir que o NR-AhR e SR-MMP foram os elementos mais importantes para previsão deste valor, sendo o NR-AR o unico valor a apresentar um contributo para a presença do pr-53 neste caso.

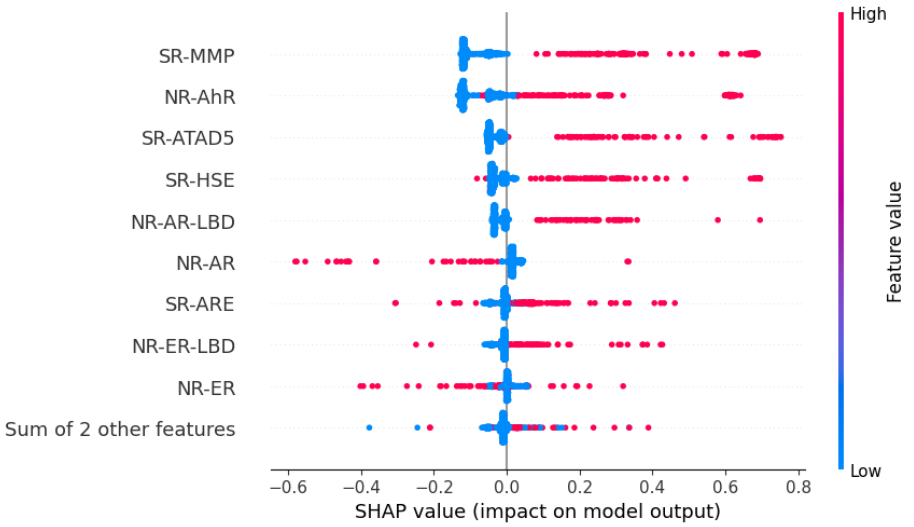

Fig. 10. Summary Plot- Shap Values

Ao contrário do grafo na figura 12 este representa o efeito nas várias instâncias, sendo que a coloração mantem-se. Assim, um valor com uma grande variedade de valores apresenta-se como importante para a decisão final do modelo. Este gráfico vai de encontro com os resultados apresentados no gráfico de cascata, sendo que os elementos-chave, os que possuem valores mais dispersos, são o SP-MMP, NR-AhR e o SR-ATAD5, sendo o NR-ER-LBD o que aparenta ser o mais insignificante.

6.2 GraphConvModel

De seguida foi usado o GraphConvModel fornecido pelo deepmol de modo a se aproximar do modelo criado. É importante notar que este modelo obteve resultados similares ao criado pelo grupo quanto a suas métricas. Este modelo não

tem em consideração todos os valores expostos, sendo usado apenas o formato da molécula. Sendo assim, não é possível fazer os grafos apresentados no modelo de Random Forest. O modelo que o grupo conseguiu fazer após várias tentativas de integração de outras ferramentas e grafos foi o de atenção as ligações das moléculas apresentado abaixo.

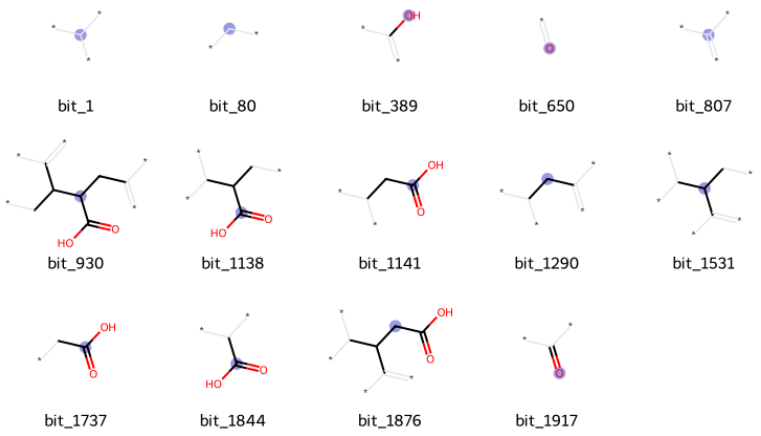

Fig. 11. Atom importance table

Esta tabela é igual para ambos os modelos, isto pode ser um sinal quanto a real importância destas ligações e átomos no esquema. Complementarmente, foi desenvolvida uma pequena API que retorna as feature importance estruturais das moléculas, sendo possível ter uma visão web para analisar e interpretar a previsão de um dos modelos do deepmol.

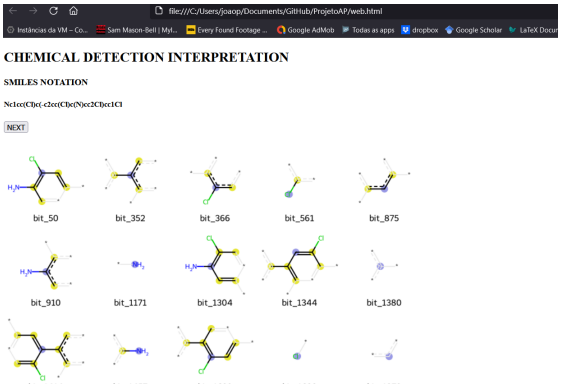

Fig. 12. Vista web desenvolvida da feature importance de um modelo do deepmol

7 Comparação

Para além dos modelos mencionados nas secções 6 o grupo tentou implementar outros modelos e tipos de interpretabilidade criados pelas bibliotecas deepchem e deepmol de modo a explorar de uma forma mais profunda as decisões, contudo estes foram pouco flexíveis quanto as ferramentas e datasets a serem usados. Esta flexibilidade é uma grande vantagem do modelo criado pelo grupo visto que é possível adicionar camadas de interpretabilidade internamente ao modelo de um modo mais fácil.

Ainda mais, foi comparado os seus scores de modo a verificar qual foi o melhor modelo com as suas decisões. Como previamente indicado, o melhor modelo no seu dataset de teste foi o Random forest com uma accuracy de 0.826 seguido pelo modelo criado pelo grupo com 0.816 foi finalmente o GraphConvModel com 0.765.

Como mencionado anterior mente o pré-processamento é muito semelhante a todos os modelos de modo a manter a comparação entre estes, porém há diferenças notáveis quanto a transformação dos dados em objetos. Enquanto o random forest não necessita de qualquer transformação, usando o smile na sua forma de string para fazer as suas comparações, os outros dois modelos necessitaram a transformação de alguma informação em grafos. Para o modelo criado pela biblioteca deepmol foi usado também o seu featurizer visto que este não é compatível com o grafo criado para o modelo pessoal, isto diferencia-se na falta dos valores de ligação com os químicos, sendo o único levado em consideração sendo o valor objetivo.

8 Conclusão

Em suma, o grupo se encontra contente com o seu modelo criado, sendo que este apresenta uma combinação forte entre ter uma boa accuracy e interpretabilidade, perdendo por muito pouco para o modelo de Random Forest, sendo que este foi desde o início apresentado como o modelo a derrotar. Por isso, chegar tão próximo deste modelo, mantendo a boa interpretabilidade e com um tempo limitado e bastante curto foi excelente.