Universidade Estadual do Ceará - PPGCC
Aprendizado de Máquina em Grafos 2021.2
Prof. Dr. Leornardo Sampaio Rocha
Aluno: Matheus Santos Araújo

Relatório - Trabalho Avaliação 1 – Classificação de Grafos

Circular Dataset Skip Links (CSL) Graph Classification using Weisfeiler-Lehman kernel and Decision tree

Introdução

O trabalho proposto consistiu em utilizar o *Dataset Circular Skip Links (CSL)* proposto no artigo "Benchmarking Graph Neural Networks", [Dwi+20], para realizar a tarefa de classificação de grafos utilizando como *features* de cada grafo o kernel *Weisfeiler-Lehman* bem como o classificador *Decision Tree Classifier (DCT)* para o devido treino da base de dados

Base de Dados e Pré-Processamento

A base de datos CSL é formada por cento e cinquenta grafos sendo quinze de cada classse. Os grafos são 4-regulares e seu conjunto de arestas pode ser visto como um ciclo hamiltoniano junto a um conjunto de arestas entre vértices do ciclo. Tais grafos podem ser denotados por G(N, C), onde N é o número de vértices e C é a classe de isomorfismo a qual o grafo corresponde e são todos da forma $G(41, C)$, onde $C \in 2, 3, 4, 5, 6, 9, 11, 12, 13, 16$.

A base de dados foi carregada do módulo *PyTorch Geomtric* e passou por um pré-processamento onde a base foi convertida para o formato da biblioteca *Networkx* e posteriormente para formato da biblioteca *Grakel*, ambas melhor detalhadas no próximo tópico. Além disso, foi inserido em cada nó um identificador *label* responsável por discriminar cada nó individual como coloração inicial na execução do cálculo do Kernel *Weisfeiler-Lehman* (ao invés do grau dos nós, que são todos iguais pelo grafo ser regular). E por fim, a biblioteca de aprendizado de máquina *Scikit-learn*.

Ferramentas

As ferramentas utilizadas neste trabalho foram o *PyTorch Geomtric* uma biblioteca construída sobre o *PyTorch* para escrever e treinar Redes Neurais de Grafos (GNNs) para uma ampla gama de aplicações relacionadas a dados estruturados e a qual possui uma série de bases de dados disponíveis para carregamento como o utilizado, CSL. Em seguida, o *Networkx*, um pacote Python para a criação, manipulação e estudo da estrutura, dinâmica e funções de redes complexas foi utilizado para a ilustração gráfica do grafo, pré-processamento e entrada para o tipo de grafo Grakel. Grakel, por sua vez, é um pacote Python que fornece implementações de vários kernels de grafo, uma família de métodos poderosos que permitem abordagens de aprendizagem baseadas em kernel e totalmente compatível com a biblioteca utilizada para aprendizado de máquina, *Scikit-learn*. *Scikit-learn* é uma biblioteca de aprendizado de máquina de código aberto, que possui diversos métodos de pré-processamento, classificadores e regressores para previsão de dados. Mais precisamente, foi utilizado o classificador de árvore de decisão neste trabalho.

Implementação

A implementação utilizou as bibliotecas supracitadas, sobretudo a biblioteca de *kernels* de grafos grakel, a qual implementa uma das três instâncias do kernel Weisfeiler-Lehman definida no artigo original em [She+11] e chamada Weisfeiler-Lehman Subtree. Portanto, é deinida uma cor inicial (rótulo ou *label*) para cada nó do grafo utilizando a base de dados no formato *Networkx* para essa inserção, por conveniência. Logo, o cálculo do kernel é

realizado e obtemos a matriz de similaridade (normalizada) entre o grafo e os demais que será entrada para o algoritmo de aprendizado de máquina. Essa matriz é cálculada com base no cálculo do kernel entre cada par de grafos, como são 150 grafos então teremos 150 pares (deste grafo com todos os demais) para cada grafo resultando em uma entrada final de dimensão 150x150 (150 grafos com 150 valores calculados cada).

Também no estágio onde o conjunto de dado se encontra no formato *Networkx* (antes da conversão para formato Grakel) dividimos as amostras em treino e teste, de forma aleatória e a considerar todas as classes utilizando uma constante de estado aleatório para replicar o experimento. As configurações de divisão da base de dados foi realizada de forma a possuir uma boa parte para treino em busca de generalização, mas ao mesmo tempo com uma razoável quantidade de amostras de teste para não obter *Overfiting*. Por fim, a partir de análises empíricas e testes a divisão a seguir foi definida por gerar bons e equilibrados resultados:

Dados de treino: 80%

Dados de teste: 20%

Experimentos e Resultados

Os experimentos foram executados utilizando as bibliotecas e parâmetros descritos nos tópicos anteriores. A Árvore de Decisão utilizada, da biblioteca *Scikit-learn*, foi configurada com os parâmetros padrão da função não sendo necessária a personalização do modelo treinado para se atingir valores altos de acurácia. A partir disso, realizamos a predição com os 20% de grafos destinados a teste cálculamos a acurácia e matriz de confusão comparando as classes reais e preditas. Os resultados são apresentados a seguir:

Acurácia: 100%

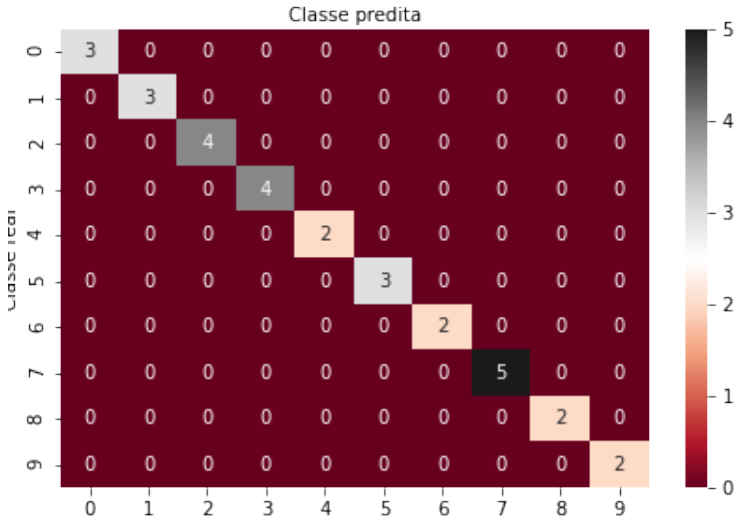

Figura 1: Matriz de confusão.

Referências

[She+11] Nino Shervashidze et al. "Weisfeiler-Lehman graph kernels." Em: *Journal of Machine Learning Research* 12.9 (2011).

[Dwi+20] Vijay Prakash Dwivedi et al. "Benchmarking graph neural networks". Em: *arXiv preprint arXiv:2003.00982* (2020).