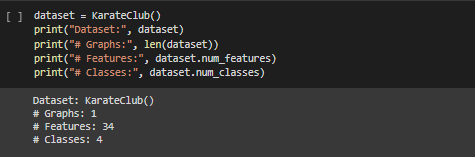
Bộ dữ liệu Zachary’s Karate Club Network thể hiện mối quan hệ tình bạn giữa những thành viên trong câu lạc bộ karate được nghiên cứu bởi Zachary trong những năm 70. Mỗi cạnh trong graph kết nối với hai cá thể nếu hai cá thể đó tiếp xúc với nhau ở bên trong câu lạc bộ.

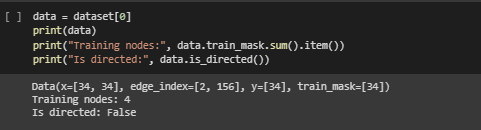
Bộ dữ liệu này có sẵn trong PyTorch Geometric. Trong đó chứa đủ các loại phương thức học sâu trong graph và dữ liệu khác. Chúng ta bắt đầu bằng việc xem xét thuộc tính của bộ dữ liệu này. Bộ dữ liệu này chỉ chứa duy nhất 1 graph vì nó chỉ miêu tả 1 câu lạc bộ. Hơn nữa mỗi node trong bộ dữ liệu này sẽ có một vector 34 chiều độc lập. Mỗi một thành viên trong clb là một phần trong bốn nhóm hoặc là lớp theo ngôn ngữ ML.



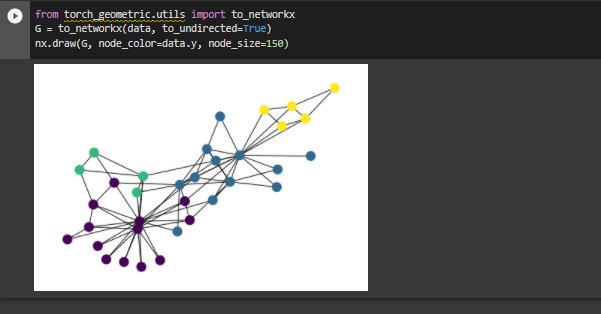
Chúng ta sẽ tìm hiểu sâu hơn về graph trong bộ dữ liệu này. Có thể thấy rằng đây là một graph vô hướng với 34 node, mỗi node có 34 thuộc tính. Thì cạnh được biểu diễn dưới dạng tuples và chúng ta có tổng là 156.

Tuy nhiên, trong PyG mỗi cạnh vô hướng sẽ được biểu diễn bởi 2 tuples cho mỗi hướng, điều đó có nghĩa là có 78 cạnh khác nhau trong Karate Graph.

Mỗi node sẽ có một nhãn chứa thông tin về lớp mà lớp đó thuộc về. Bộ dữ liệu còn chứa *train\_mask* sẽ giúp cho biết node thực sự nó đang đoán. Có 4 node thực, mỗi node sẽ thuộc 1 nhóm khác nhau và những node này sau này sẽ suy ra tới những nhóm của các node còn lại.



Chúng ta sẽ biến đổi Karate Club Network thành một Networkx Graph, nó cho phép chúng ta dùng lệnh *nx.draw* để minh họa nó. Những cái node sẽ được tô màu dựa theo nhóm nó thuộc về.



Khi huấn luyện một mô hình để biểu diễn giải quyết node classfication nó có thể được biết đến như là máy học có giám sát.

GCNs sẽ được dùng để phân loại node trong bộ dữ liệu test. Giới thiệu ngắn gọn về lý thuyết, mỗi cái layer sẽ được viết dưới dạng như một hàm không tuyến f.

Đầu vào sẽ là một ma trận A ở trong graph và một node chứa những thuộc tính H cho một vài layer *l.* Một cái công thức đơn giản cho GNN trông như sau

Với W là trọng số ma trận cho neural network thứ l, và là một hàm chức năng không tuyến tính. Nhân trọng số đó với ma trận kề nó có nghĩa là các vector thuộc tính của các vector liền kề được cộng lại và tổng hợp lại cho mỗi node. Tuy nhiên, trong tổng đó sẽ không chứa vector thuộc tính của node đó.

Lớp GCN có sẵn trong PyG và có thể được import. Giống với cách xếp chồng các lớp trong Neural Network thông thường, chúng ta cũng có thể xếp chồng nhiều lớp GCN. 3 lớp GCN sẽ tạo ra 3 bước lan truyền nối tiếp nhau. Số lượng Input của lớp đầu tiên phải bằng số lượng thuộc tính của mỗi node.

Text

Description automatically generated

Ta sử dụng cross-entropy làm hàm mất mát vì nó rất phù hợp cho các bài toán phân loại nhiều lớp và khởi tạo Adam như một trình tối ưu hóa gradient ngẫu nhiên. Ta tạo một vòng lặp đào tạo PyTorch tiêu chuẩn và để nó chạy trong 300 epoch. Sự mất mát được giảm bớt trong quá trình huấn luyện, có nghĩa là việc phân loại hoạt động tốt. Các bản nhúng 2 chiều từ lớp GCN cuối cùng được lưu trữ dưới dạng danh sách để chúng tôi có thể tạo hiệu ứng cho quá trình nhúng trong quá trình đào tạo, cung cấp một số thông tin chi tiết về không gian tiềm ẩn của mô hình.

Text

Description automatically generated