TP3: GNNs

Fecha de entrega límite: 15 de diciembre de 2021

Objetivo: Entender el funcionamiento de una red neuronal en un grafo. Poder modificar sus parámetros comprendiendo las implicancias de esas modificaciones.

Requerimientos:

- Conocimientos de Python básico (y tener Python instalado o utilizar alguna plataforma online de Python en notebooks: Google Collaboratory, Kaggle, etc)
- Teoría de ML: redes neuronales feed-forwards, redes neuronales convolucionales, redes neuronales en grafos
- Uso básico de Git y GitHub.

Ejercicio 1: Estudio de los datos

Descargar los datos contenidos en authorshipData.mat y el archivo dataTools.py que contiene las primitivas necesarias para procesar los datos.

A. Estudiar la clase Authorship en el archivo dataTools.py. Investigar los atributos y métodos de la clase.
B. Elegir Jane Austen como autora para analizar, instanciar la clase para procesar los datos. Seleccionar el 95% de las muestras para entrenamiento y 8% de ellas para validación.
C. Crear un grafo que esté normalizado por filas, que no contenga nodos aislados, que sea no-dirigido y que esté conectado (observar las opciones de inicialización de la clase).
D. Imprimir el listado de “palabras funcionales” que se consideran.
E. Mostrar el grafo resultante para Jane Austen utilizando las herramientas en graphTools.py. Obtener el grafo para Edgar Allan Poe. ¿Se ven diferencias a simple vista? ¿Cuál es el grado máximo para cada grafo?
F. Imprimir la cantidad de señales en el grafo que hay en el conjunto de entrenamiento, de validación y de evaluación. ¿Cuántas de estas muestras corresponden a Jane Austen? Mostrar también un promedio de los histogramas de palabras funcionales.

Ejercicio 2: Clasificación con Redes Neuronales en Grafos

Utilizar la clase GraphConv provista en graphML.py. Esta clase funciona de manera similar a nn.Conv1d.

A. Crear una GNN de una capa, con una función de activación ReLU, e incluir una capa de salida (un perceptrón multicapa) luego de la capa de convolución en el grafo. Obs.: La salida de la capa convolucional es una señal en el grafo con múltiples features. Esta señal debe ser convertida en un vector para poder procesarla con un perceptrón multicapa en la capa de salida.
B. Entrenar la GNN con una pérdida de CrossEntropyLoss entre las 2 clases de salida (el texto pertenece al autor o no) y los labels provistos en las muestras de entrenamiento. Incluir early stopping en el entrenamiento.
C. Probar distintos valores de features a la salida de la capa convolucional F = [16, 32, 64], y distintos valores de inclusión de vecindarios K = [2, 3, 4].
D. Probar distintos valores de tasa de aprendizaje [0.001, 0.005, 0.01].
De ahora en adelante, utilice los valores de F, K y tasa de aprendizaje que arrojaron los mejores resultados en las muestras de evaluación.
E. Determinar si hay overfitting. Mostrar cómo se determina. Si se observa overfitting, probar con distintas técnicas (dropout y/o regularización L2) para reducirlo.

Ejercicio 3: Comparación y Análisis

Tome del ejercicio 2 los valores de F, K y tasa de aprendizaje que arrojaron mejores resultados. Continúe usando dropout y/o regularización si los considera necesarios.

A. Cree una GNN con dos capas convolucionales (y la capa de dropout). Para la segunda capa use el doble de features que para la primera, y el mismo valor de K.
B. Cree un perceptrón multicapa que tenga el mismo número de hidden units que la GNN.
C. Cree un CNN que tenga el mismo número de features que la GNN.
D. Entrene todas estas redes neuronales, obtenga las evaluaciones y compare los resultados. ¿Cuál ofrece el mejor resultado? ¿Hay overfitting?
E. Cambie el orden de los nodos (i.e. una permutación del vector de señales, junto con la correspondiente permutación de la matriz del grafo). Evalúe las redes neuronales nuevamente, sin re-entrenar. ¿Observa alguna diferencia?
F. Reentrene las redes y evalúelas. ¿Observa alguna diferencia?