COMP 642 — Machine Learning Feb 22, 2022

Lecture 14: Graph and Node Classification

Lecturer: Anshumali Shrivastava *Scribe By: Pengfei Qiao, Yuxuan Kou, Ying Xue, Yufei Li*

1 What is attention mechanism in Machine Learning

The implementation of the attention mechanism in artificial neural networks does not necessarily track the biological and psychological mechanisms of the human brain. Rather, it is the ability to dynamically highlight and use the salient parts of the information at hand, in a similar manner as it does in the human brain, that makes attention such an attractive concept in machine learning.

1.1 How does attention mechanism works

An attention-based system is thought to consist of three components:

1. A process that "reads" raw data (such as source words in a source sentence), and converts them into distributed representations, with one feature vector associated with each word position.

2. A list of feature vectors storing the output of the reader. This can be understood as a "memory" containing a sequence of facts, which can be retrieved later, not necessarily in the same order, without having to visit all of them.

3. A process that "exploits" the content of the memory to sequentially perform a task, at each time step having the ability put attention on the content of one memory element (with a different weight).

Let's take the encoder-decoder framework as an example, since it was within such a framework that the attention mechanism was first introduced. If we are processing an input sequence of words, then this will first be fed into an encoder, which will output a vector for every element in the sequence. This corresponds to the first component of our attention-based system, as explained above.

A list of these vectors (the second component of the attention-based system above), together with the decoder's previous hidden states, will be exploited by the attention mechanism to dynamically highlight which of the input information will be used to generate the output.

At each time step, the attention mechanism, then, takes the previous hidden state of the decoder and the list of encoded vectors, and uses them to generate unnormalized score values that indicate how well the elements of the input sequence align with the current output. Since the generated score values need to make relative sense in terms of their importance, they are normalized by passing them through a softmax function to generate the weights. Following the softmax normalization, all of the weight values will lie in the interval [0, 1] and will add up to 1, which means that they can be interpreted as probabilities. Finally, the encoded vectors are scaled by the computed weights to generate a context vector. This attention process forms the third component of the attention-based system above. It is this context vector that is, then, fed into the decoder to generate a translated output.

1.2 Why using attention mechanism

By adopting attention mechanism, neural network could automatically detect the relationship between words, so that it can put words in a specific context and complete the analysis based on that context but not treat each word in different documents as the same. Thus, it can significantly improve the accuracy of translation or text classification. However, using attention mechanism is just like boost our model on a large data set, which can provide us with a really high accuracy with a cost of calculation complexity. If the task on hand does not need to be really accurate, then it is fine to just use the embedded table.

1.3 An example of encoding text using attention mechanism

Let's have an example to illustrate how this mechanism works. Assume that we have a sequence of words: "Rice is planning to teach CS-Ethics". After pre-processing, we can get the word list as: ['Rice', 'is', 'planning', 'to', 'teach', 'CS-Ethics']. By feeding the word list into an encoder, which is the embedded table we're using in this scenario, we can get a list of vector v^1 to v^6 in Figure 1.

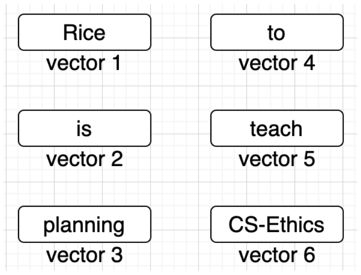
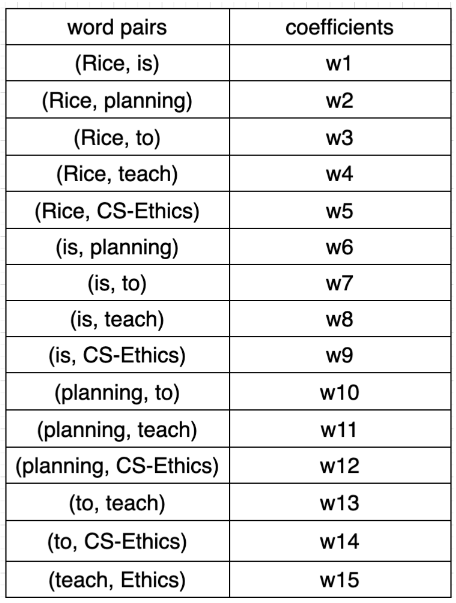


Figure 1: Vector of each word

Then we initialize the weight matrix in Figure 2 which is so called interaction coefficient. It represents the attention between any pair of words in our word list. This coefficient shows the relationship between two words. Like In our example, the coefficient between 'teach' may enhance the meaning of 'Rice' as a University but not a food. So the encoding process after adopting the attention mechanism will be like for each word, we sum up the product of the vector of this word and the coefficient between this word and every word in the sequence of words. After this, each word vector will not only be determined by this word in the embedded table but also be influenced by other words in the sequence.



word pairs	coefficients
(Rice, is)	w1
(Rice, planning)	w2
(Rice, to)	w3
(Rice, teach)	w4
(Rice, CS-Ethics)	w5
(is, planning)	w6
(is, to)	w7
(is, teach)	w8
(is, CS-Ethics)	w9
(planning, to)	w10
(planning, teach)	w11
(planning, CS-Ethics)	w12
(to, teach)	w13
(to, CS-Ethics)	w14
(teach, Ethics)	w15

Figure 2: Interaction coefficients matrix

In this scenario, the result using embedded table would be:

$$result = \sum_{i=1}^{6} v^i \tag{1}$$

When using attention mechanism, for every training iteration:

$$v^i = \sum_{n} w^n v^i \tag{2}$$

$$result = \sum_{i=1}^{6} v^i \tag{3}$$

2 Graph Neural Network

In recent years, due to the great expressive power of graphs, the research of graphs analysis based on machine learning has attracted more and more attention. Graph Neural Networks (GNNs) are neural models that capture the dependence of graphs via message passing between the nodes of graphs. Graph Neural Network is a unique non-Euclidean data structure for machine learning, and GNN is mainly used for node classification, link prediction, clustering and other tasks.[1]
As we learned in data structure, graph is consisting of nodes and edges. In a graph, the nodes represent objects and the edges represent relationships between these objects.
In later paragraphs, we denote a graph as $G = (V, E)$, where $|V| = N$ is the number of nodes in this graph and $|E| = N^e$ is the number of edges in this graph. To build a Graph Neural Network, there are 4 steps to follow: find the graph structure, specific graph type and scale, design loss function and build model using computational modules.
The method of transmitting our graph information as the input of the neural network is the focus of our class.

2.1 The Graph Structure in GNN

The graph structure with nodes and edges can be represented as Figure 3. The features of node can be represented by matrix or vector. The edges of graph could also be written as matrix or vector by using one-hot encoding. First, we define a neighbor vector V^n for each node to record the edge information for this node. In this neighbor vector V^n, we use one-hot encoding to show the relationship of the nodes' edges, which use 0 to represent there is no edge between the current node and the index node and use 1 to represent the edge is exist between these two nodes.
For the node 6 in Figure 3, the edge vector is as follow, which means node 2, 4, 7, 8 is the neighbors of node 6:

$$V_6^n = [0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0] \tag{4}$$

2.2 Different Task of Graph Neural Network

There are generally three types of prediction tasks on the graph: graph level, node level and edge level.

In graph level tasks, we predict individual attributes of the entire graph. For node level tasks, we predict some attributes of each node in the graph. For edge level tasks, we want to predict the properties or existence of edges in the graph.

For the above three levels of prediction problems (graph level, node level and edge level), we will show that all the following problems can be solved by a Graph Neural Network.

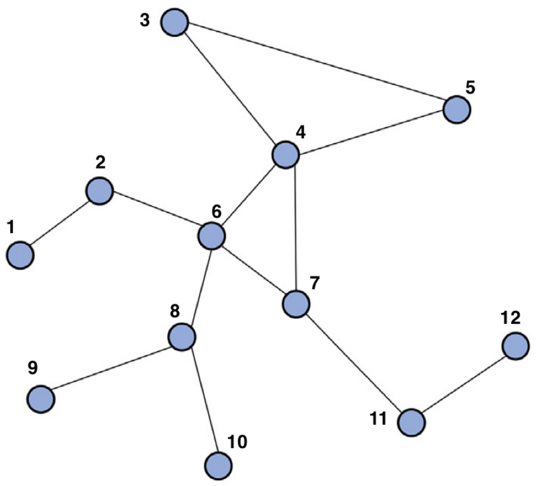


Figure 3: Graph with Non-Euclidean Space

2.3 Edge Level Task of Graph Neural Network

For the edge level task of graph neural network, we are given nodes that represent the objects in the image, we wish to predict which of these nodes share an edge or what the value of that edge is. If we wish to discover connections between entities, we could consider the graph fully connected and based on their predicted value prune edges to arrive at a sparse graph. In the class, we discussed about the edge level task of Graph Neural Network by preferred movie prediction.

3 PageRank Algorithm

In real life applications, lots of data exist in the form of graphs. For example, the Internet and social networks can be regarded as a graph. Machine learning on graph data has important theoretical and practical significance. The PageRank algorithm is a representative algorithm for link analysis of graphs, and belongs to the unsupervised learning method on graph data.

3.1 Directed graph and PageRank

Assuming that the Internet is a directed graph, a RandomWalk model is defined on its basis. That is a Markov chain, which represents the process of web browsers randomly browsing web pages on the Internet. Suppose that the browser jumps to the next page with equal probability according to the connected hyperlinks on each web page, and continues to make such random jumps on the Internet. This process forms a first-order Markov chain. PageRank represents the stationary distribution of this Markov chain. The PageRank value of each web page is the stationary probability.

Supposing a simplified Internet example as the directed graph below. Nodes A, B, C and D represent web pages, and directed edges between nodes represent hyperlinks between web pages , and the weight on the edge represents the probability of random jumping between web pages. Suppose there is a viewer who walks randomly on the Internet. If the viewer is on the web page A, the next step is to move to the web page B, C and D with the probability of 1/3. If the viewer is on the web page B, the next step is to transfer to the web page A and D with the probability of 1/2. If the viewer is on the web page C, the

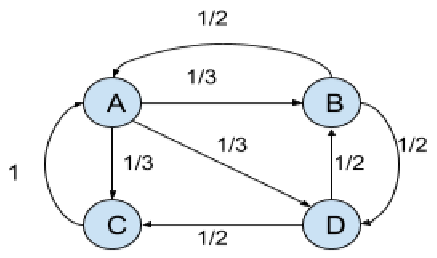


Figure 4: Example of directed graph

next step is to transfer to the web page A with probability 1. If the viewer is on the web page D, the next step is to transfer to the web page B and C with the probability of 1/2.

For a webpage, if there are more hyperlinks pointing to the webpage, the higher the probability of randomly jumping to the webpage, the higher the PageRank value of the webpage, and the more important the webpage is.

3.2 RandomWalk Model

Given a directed graph containing n nodes, define a random walk model on the directed graph, that is, a first-order Markov chain, where nodes represent states, and directed edges represent states between states. The transition between nodes assumes that the transition probability from one node to all nodes connected by directed edges is equal. Specifically, the transition matrix is a n order matrix M.

$$M = [m_{ij}]_{n*n} \tag{5}$$

The value rule of element m_{ij} in row i and column j is as follows: if node j has k directed edges, and node i is a node connected by its point, then $m_{ij} = 1/k$; otherwise $m_{i,j} = 0, i, j = 1, 2, ..., n$. Note that the transition matrix has the property:

$$m_{ij} \geq 0 \tag{6}$$

$$\sum_{i=1}^{n} m_{ij} = 1 \tag{7}$$

That is, each element is non-negative, and the sum of the elements in each column is 1, that is, the matrix M is a stochastic matrix. The probability distribution of random walks visiting each node at a certain moment t is the state distribution of the Markov chain at moment t , which can be represented by a n dimension column vector R_t. Then at moment $t + 1$, the probability distribution R_{t+1} of visiting each node satisfies:

$$R_{t+1} = MR_t \tag{8}$$

3.3 Definition of PageRank

Given a strongly connected and aperiodic directed graph containing n nodes, define a random walk model based on it. Assuming that the transition matrix is M, the probability distribution of visiting each node at time $0, 1, 2. \ldots .t. \ldots$ is:

$$R_0, MR_0, M^2R_0, ..., M_tR_0, ... \tag{9}$$

The limit is:

$$\lim_{t\to\infty} M^t R_0 = R \tag{10}$$

Therefore exists, the limit vector R represents the stationary distribution of the Markov chain, satisfying:

$$MR = R \tag{11}$$

Given a strongly connected and aperiodic directed graph containing n nodes $v_1, v_2, ..., v_n$, define a random walk model, a first-order Markov chain, on the directed graph. The characteristic of random walk is that the transition probability from a node to all nodes connected by directed edges is equal, and the transition matrix is M . This Markov chain has a stationary distribution R.

$$MR = R \tag{12}$$

The stationary distribution R is called the PageRank of this directed graph. Each component of the R is called the PageRank value of each node.

$$R = \begin{bmatrix} PR(v_1) \\ PR(v_2) \\ \vdots \\ PR(v_n) \end{bmatrix}$$

Figure 5: The PageRank value of each node

In the equation above $PR(V_i), i = 1, 2, ..., n$, represents the PageRank value of the node.
It is obvious that:

$$PR(V_i) \geq 0, i = 1, 2, ..., n \tag{13}$$

$$\sum_{i=1}^{n} PR(V_i) = 1 \tag{14}$$

$$PR(v_i) = \sum_{v_j \in M(v_j)} \frac{PR(v_j)}{L(v_j)}, i = 1, 2, ..., n \tag{15}$$

Here $M(V_i)$ represents the set of nodes pointing to the node v_i, and $L(v_j)$ represents the number of directed edges connected by the node v_j.

3.4 General Definition of PageRank

Given an arbitrary directed graph with n nodes $v_i, i = 1, 2, ..., n$. Suppose to consider a random walk model on the graph, that is, a first-order Markov chain, whose transition matrix is M, the transition probability from a node to all its connected nodes is equal. This Markov chain does not necessarily have a stationary distribution. Suppose to consider another completely random walk model, the elements of its transition matrix are all $1/n$, that is to say, the transition probability from any node to any node is $1/n$. The linear combination of the two transition matrices in turn forms a new transition matrix on which a new Markov chain can be defined. It is easy to prove that this Markov chain must have a stationary distribution, and the stationary distribution satisfies:

$$R = (dM + \frac{1-d}{n}E)R = dMR + \frac{1-d}{n}1 \tag{16}$$

where $0 \leq d \leq 1$ is a coefficient, called damping factor, E is a matrix with all 1, R is a n dimensional vector, and 1 is a 1 with all components of n dimensional vector. R represents the general PageRank of a directed graph.

Then we can write the general function of PageRank on the basis of equation (12), which is:

$$PR(v_i) = d(\sum_{v_j \in M(v_j)} \frac{PR(v_j)}{L(v_j)}) + \frac{1-d}{n}, i = 1, 2, ..., n \tag{17}$$

The second term is called smooth I page. Due to the smooth term, the PageRank value of all nodes will not be 0, and it has the following properties:

$$PR(V_i) \geq 0, i = 1, 2, ..., n \tag{18}$$

$$\sum_{i=1}^{n} PR(V_i) = 1 \tag{19}$$

3.5 Summary of PageRank

Given an arbitrary directed graph with n nodes, define a general random walk model, a first-order Markov chain, on the directed graph. The transition matrix of the general random walk model is composed of a linear combination of two parts, one is the basic transition matrix of the directed graph M, which means that the transition probability from a node to all the nodes connected to it is equal, and the other part is the basic transition matrix of the directed graph. It's a completely random transition matrix, indicating that the transition probability from any node to any node is $1/n$, and the linear combination coefficient is the damping factor d ($0 \leq d \leq 1$). This general random walk Markov chain has a stationary distribution, denoted by R . Define the stationary distribution vector R as the general PageRank of this directed graph.R is decided by equation:

$$R = dMR + \frac{1-d}{n} 1 \tag{20}$$

where 1 is a n dimensional vector with all components 1.

The general definition of PageRank means that Internet browsers walk randomly on the Internet according to the following methods: On any web page, the browser decides to jump randomly according to hyperlinks with probability d, and then with equal probability The link jumps to the next web page; or the probability$(1-d)$ determines a completely random jump, then jumps to any web page with an equal probability $1/n$. The second mechanism ensures that pages from unconnected hyperlinks can also be jumped out. This can ensure a stationary distribution, that is, the existence of general PageRank, so general PageRank is suitable for any structure of the network.

4 Node Classification

A social network is essentially a representation of the relationships between social entities, such as people, organizations, governments, political parties, etc. The interactions among these entities generate unimaginable amounts of data in the form of posts, chat messages, tweets, likes, comments, shares, etc. This opens up a window of opportunities and use cases we can work on.

Node classification or link prediction is one of the most important research topics in the field of graphs and networks. The objective of link prediction is to identify which nodes belongs to the same group, or identify pairs of nodes that will either form a link or not in the future.

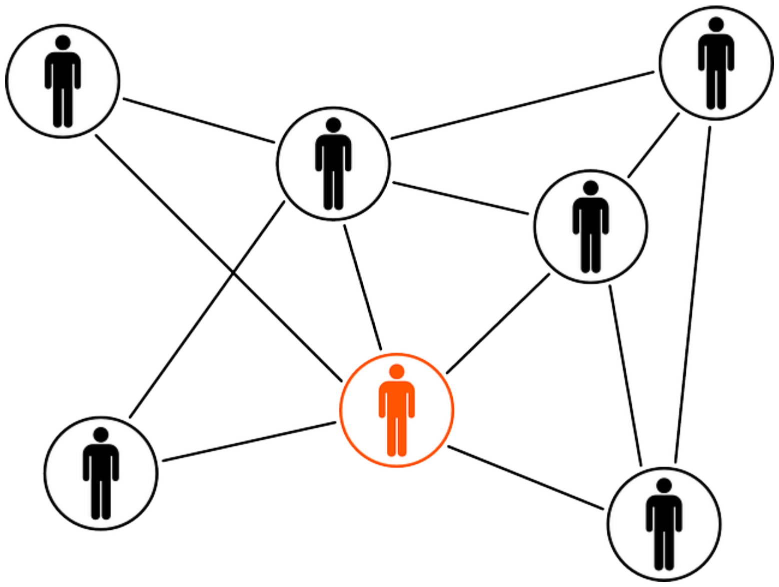


Figure 6: Social network

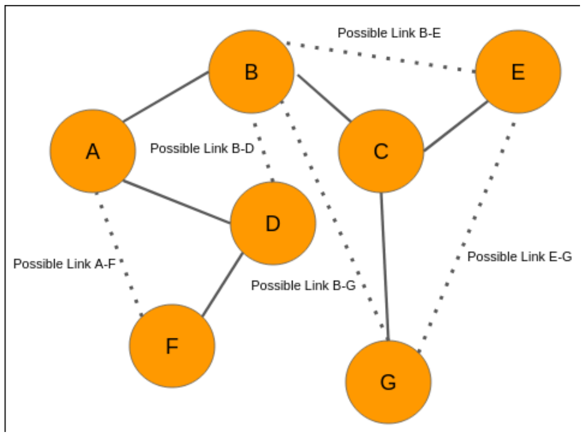


Figure 7: Link prediction

If we can somehow represent a graph in the form of a structured dataset having a set of features, then maybe we can use machine learning to predict the formation of links between the unconnected node-pairs of the graph.

The most important thing is to create features for each and every pair of nodes.

Referring to the previous idea of embedding, we can use an embedding model to convert a node into a vector. The good news is that there are several techniques to represent a node in vector. Let's say we use the one-hot code as the simplest way. Then how can we represent the connection between nodes?

Here is our thought. As we have discussed, we not only want our neighbor information, we want to store more information, like the neighbors of my neighbors and so on. So the algorithm is listed as below:

(1) Embedding

Use some techniques, for instance the node2vec[2] algorithm, to convert a node into a vector, to facilitate the utilizing of the machine learning model, since machine learning model could only handle

vector input.
(2) Dimension expanding

As you can see in Fig 8. Each node would take the vector from their neighbor and multiple the Wi, and then sum up the results of all neighbor nodes as a new vector for its own node. The whole process would take several iterations to generate the final node vector.

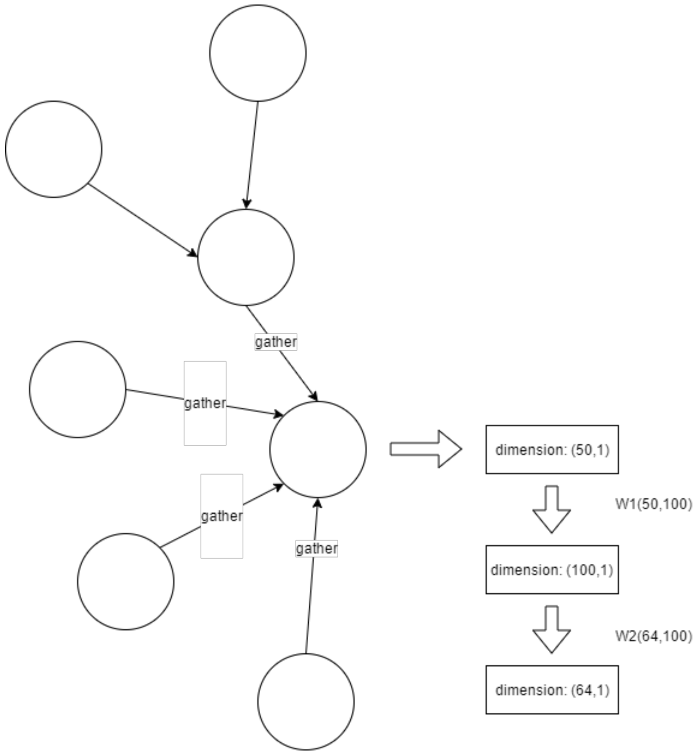


Figure 8:

W_i is used on each node on the same step. Every time a matrix operation is performed on a node, this weight parameter could reduce some useless information after expanding dimensions. By using this W_i parameter, since each step each node could store some information from their neighbor, each node is able to gather the information of neighbors of their neighbors.

Thus by training the W_i, we could find a perfect way to represent graphs by vectors.

References

[1] "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[2] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," 2016.