

Social Network Analysis
Prof. Tanmoy Chakraborty
Department of Computer Science and Engineering
Indraprastha Institute of Information Technology, Delhi


Chapter - 09
Lecture - 04

Let us start with our a new chapter this is chapter 9 and this is a very exciting chapter because this is something that has been happening since last 5 6 years in a post deep learning right the advent of deep learning in 2015 2016 this trend has started. Now this is on graph representation learning a very exciting chapter I am not sure how much I can explain because this is a separate course altogether and this really requires a significant background on deep learning machine learning etcetera.

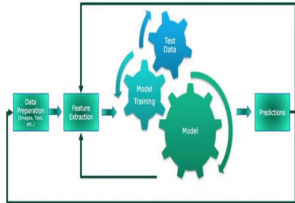
So, the t of this course Shivani has given you know some ideas at least you know the basic ideas about graphs grab I mean the deep learning right GNN sorry CNN RNN right attention. So, hopefully you know I will try to give you a very high level overview of this I will also point you to some of the important materials right if you really want to know graph representation learning well, but this is something which is a state of thought these days.

(Refer Slide Time: 01:35)

Machine Learning Pipelines



A Standard Machine Learning Pipeline



- Data preparation:**
 - collecting and annotating data according to requirements
- Data pre-processing:**
 - collected data is often noisy and unstructured
 - mandatory cleaning and organizing of the data
- Feature extraction:**
 - extract relevant features from our processed data
 - several statistical measures (mean, standard deviation, entropy, etc.) are used as features
 - domain specific features also extracted
- Learning algorithm:**
 - features sent as input to ML algorithm for prediction
 - with ground-truth labels (supervised learning)
 - without ground-truth labels (unsupervised learning)

<https://www.datacamp.com/2018/09/05/how-to-build-a-better-machine-learning-pipeline/>



So, well so, since this course is for students who do not have any background may not have any background for I mean may not have any background on machine learning. So, let me

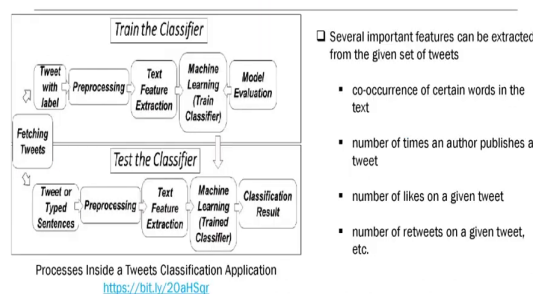
briefly discuss what a machine learning pipelines basically mean right a standard machine learning machine learning pipeline you know constitute of this modules. So, data preprocessing you are given a data either in terms of images text or time series data and so on.

You process the data then you extract features from the data, then you design some model. You train the model test the model right you fine tune the model and then you use this model for prediction ok. So, data preparation collecting and annotating data according to the requirement data pre-processing collected data is often noisy and unstructured mandatory cleaning is needed.

Feature extraction you extract features manually extracted you manually extract features handcrafted features right based on your intuition your hypothesis and so on. And then if you have domain specific knowledge then that would be great that would help you identify appropriate features and then you use learning algorithms ml algorithms may be supervised or unsupervised semi supervised for predicting labels ok.

(Refer Slide Time: 03:11)

Example: Feature Extraction from Texts



- Several important features can be extracted from the given set of tweets
 - co-occurrence of certain words in the text
 - number of times an author publishes a tweet
 - number of likes on a given tweet
 - number of retweets on a given tweet, etc.



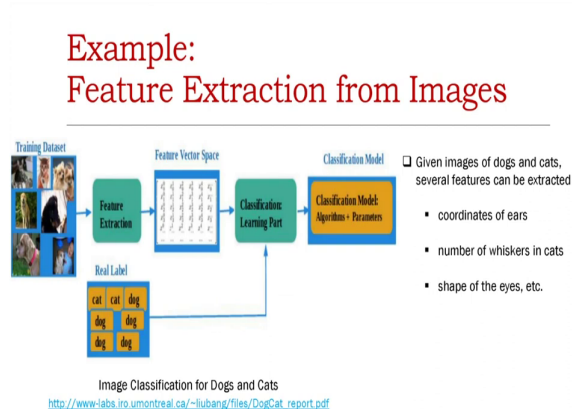
So, feature extraction is the pain why because you really need to know the domain properly. Say for example, you are working on fraud detection right for a naïve user for a naïve researcher right if he or she does not have any idea about the domain past experience. You would not be able to understand what types of features people generally talk about people generally consider right. In case of text say I mean in NLP Natural Language Processing you

have a sentence right and say the task is to predict the sentiment or determine the sentiment of the sentence.

So, you can identify manually identify some words which are which basically indicate positive sentiments right say extremely good or marvelous right. Whereas, you can also come up with a vocabulary of negative words for example, awful or say bad or these kind of words right and then you can also right you can also use Ingram kind of features right.

Consecutive words to a consecutive two words consecutive three words part of speech and so on and so forth. So, but you may not exactly know that what kind of features are required for this task the task is sentiment analysis right.

(Refer Slide Time: 04:46)



So, this is basically a problem right this is really a problem sentimental analysis is a very simple task if you think of complicated task for we say for example, legal document classification right. You really need to know the legal aspect right and depending on that you may be able to identify how what kind of you know vocabulary, what kind of tokens types are used in a legal document right, how what is are distributed, how what is former sentence and so on and so forth.

So, given an say let us think of an image classification problem right what you do say given an image of say dog and cat right you can extract features like coordinates of ears right number of you know whiskers in cat right shape of eyes and so on. Now, these are something

that you manually identify ok. I mean manually I mean what do you mean by manually identify you basically manually determine that these set of features need to be extracted.

(Refer Slide Time: 05:53)

Feature Extraction: Challenges



- Given a situation, there are a large number of possible features you can extract
- How should one choose which features to select from this set?
- Should she take all the features from the pool? +
- Should she take only few out of them?
- How to make a decision in such a situation?
- Is it possible to **encapsulate** the feature extraction process with the learning algorithm?
- Can it be ensured to extract features that **give the best possible results**?
- The answer is **Representation Learning**



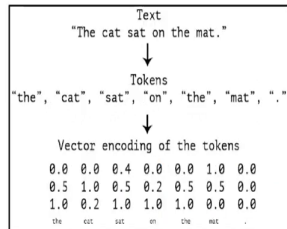
And depending on that you come up with your algorithms to extract it. So, what are the challenges of feature extraction manual feature extraction? So, given a situation there are large number of possible features right. That you can extract. How should one choose? Which features to select? Right.

Should see take all the features from the pool should she take only few out of them right? How to make a decision in such a situation is it possible to encapsulate the feature extraction process with the learning algorithm right what happens if we replace this manually handcrafted feature extraction process by an automation automated engine.

So, that automated engine will automatically extract features it automatically identifies important features and let the classifier do the task the end task right. So, this is the task of representation learning you basically you feed image you feed text you feed time series data you feed any unstructured data and your representation learning model produces features automatically and those features are being fed to the learning models the classification models in tasks for prediction ok.

(Refer Slide Time: 07:18)

Representation Learning



Representation of words in texts
<https://bit.ly/39yFDs>

- The field of machine learning, concerned with automatic computation of features from a given data
- The representation can further be used with various machine learning models
- Also known as **feature learning**
- Most representation learning algorithms depend on learning a vector
- Representations are mostly **task-dependent**

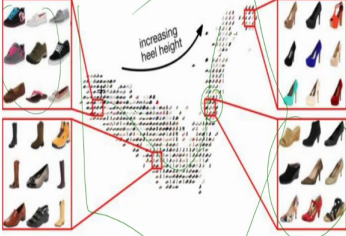


So, if you think of text again. Text is my favorite that is why I am always giving example of text right. So, let us say the gad the cat sat on the mat right. So, you can tokenize words you can tokenize the sentence into words right the cat sat etcetera. And then for each word you can come up with a vector right. This vector can be one hot encoding right or the vector can be can be taken from a predefined vector representation right.

And then you essentially come up with a concatenated version of these vectors which represent the sentence you feed it to the task and the task the basically the model behind the task runs, takes into account this one and predicts the this one. So, representations are generally task dependent ok, but we will also talk about representations which are task independent.

(Refer Slide Time: 08:35)

Similarity between Entities



- ❑ What makes images similar?
- ❑ Images are typically embedded in a feature vector space
- ❑ Their distance in feature space preserve the relative dissimilarities
- ❑ Notion of cosine similarity in vector spaces is good metric
- ❑ The above is the key to representation learning

<https://vision.cornell.edu/se3/embeddings-and-metric-learning/>

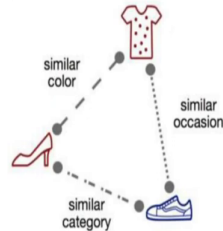


So, the major problem or the major issue is that how do we come up with similarity, similarity between entities while doing representation, while you know coming up with a beta representation. So, now, similarity can you basically want that based on some sort of notions of similarity of products you can have representations right.

For example, you see here this is basically different tools and these are representations. So, this is an embedding space and these are representations and these entities are arranged in an increasing heel height right heel of the shoe right you see that these items are grouped here these items are grouped here and so on and so forth right.

(Refer Slide Time: 09:43)

Similarity Assumption



- Simplified assumption regarding similarity is often required to be made
- Images are usually compared against a unique measure of similarity in a situation
- Fine-grained categorization suffers due to
 - lack of training data
 - large number of fine-grained categories
 - high intra-class vs. low inter-class variance

<https://vision.cornell.edu/se3/embeddings-and-metric-learning/>



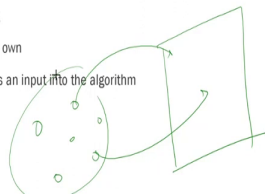
So, when it comes to similarity right think of items right. Say for example, you have a shoe and a cloth right these two items are similar in terms of color right. You have a sports shoe and a cloth these two items are similar in terms of occasion. You have two shoes these are similar in terms of some categories right. So, how do we come up with exact similarity measures to understand the similarity or proximity between a pair of items right.

(Refer Slide Time: 10:31)

Graph Representation Learning



- Graph-theoretic algorithms require to manually tune certain attributes
- Graph Representation Learning is all about employing machine learning algorithms which reduce human intervention significantly
- Has the same end goal as representation learning
- Help us solve most of the graph problems on their own
- Need to devise a method to incorporate a graph as an input into the algorithm



So, in graph representation learning the idea is that you are given a graph and you map the graph meaning that the nodes or the entities the edges of the graph to an embedding space

right. So, that the proximity between a pair of nodes in the graph is preserved on the embedding space right. So, we will try to formalize this notion in the later part of the slide.

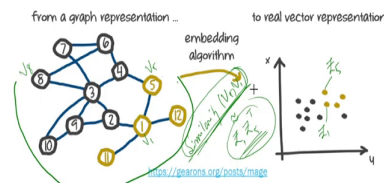
(Refer Slide Time: 11:07)

Graph Representation Learning



Example:

- map different components of a graph (nodes, edges, sub-graphs, communities) to an embedding space
- embeddings of similar types of nodes come closer
- embeddings of dissimilar nodes move away from each other



So, what is the purpose of a graph representation learning the purpose is that from a graph like this we basically map all the nodes to an embedding space such that nodes which are closer in the graph their corresponding embeddings also come closer in the embedding space. So, let us say this is. So, say this is vertex five vertex one.

So, the similarity; so, similarity of vertex 5 vertex 1 should be captured and let us say this is the embedding of vertex 5 Z_5 and this is say Z_1 embedding of vertex 1. This should be similar to the dot product of the embeddings of vertex 1 and vertex 5 right.

So, we I mean. So, here several questions can be arrived for example, how do we capture the similarity between two nodes in a graph right it can be a simple distance shortest path distance between two nodes it can be; it can be number of number of similar number of common neighbours we can use Jaccard similarity something like that.

And on the embedding space how do we measure the similarity between two embeddings we can take normal dot product or we can also think of other types of similarity measures between two vectors right. So, here the question is how to map different components of a graph. Now in this example I basically mentioned that how can we map a node to an embedding space, but it can be an edge it can be a sub graph community and so on.

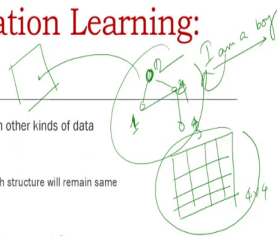
Embeddings of similar types of nodes should come closer embeddings of dissimilar nodes move away from each other. So, it is also it should also preserve the case that vertex 8 and say vertex 5 right since they are far apart their similarities should be less and their dot product should also be right basically different right. So, this is the idea right.

So, we need to capture the similarity of a pair of entities say pair of nodes in the original graph to the embedding space. So, that the such notion of similarity can be captured in a in an appropriate manner.

(Refer Slide Time: 13:43)

Graph Representation Learning: Challenges

- However, it is more interesting and challenging than other kinds of data
 - Graphs are not sequential in nature
 - Rename the nodes in a graph
 - Entries in the adjacency matrix will change, but the graph structure will remain same
- Graphs can represent enormously complex data
- Given the complex structure of graphs, what should one encode?
- Roughly define the problem as learning vector representations of various components of a graph
- Depending on use cases, it is possible to find
 - vectors that encode nodes of a graph
 - vectors that encode edges of a graph
 - vectors that encode the entire graph
 - vectors that encode paths in a graph, and so on



So, what is the challenge? The major challenge is if we look at the literature right embedding method I think it have it had started mostly in the area of computer vision their language then graph right. The problem in graph is that you know graph does not have any sequence right. For example, if we think of a graph like this 1 2 3 4 and you have an adjacency matrix right.

4 cross 4, 4 cross 4 right. Now if we rename the vertices say let us say now I call this as two this as 1 this as 4 this as 3 the graph structure will remain same, but the adjacency matrix will change right. But think of a; think of an image right image is represented by this pixel matrix right RGB matrix for example. Now if we change pixel values right the image will change completely. So, the unique pixel corresponds to an unique image.

But here as you see if you change the adjacency matrix this is essentially reordering the vertex ids right, but that would not change the topological structure. So, therefore, graph

representation itself is very tough because there is no uniqueness about the node how do we how do we call that ok this is a unique node how do we capture the uniqueness of a node right we cannot capture it using simple vertex id the vertex id can be changed right.


But if you think of say a language right say I am a boy, I am a boy in this sentence if we change the relative order of words right the sentence the meaning of the sentence will change completely. So, here the sequence matters a lot, but in case of graph there is no sequence as such in case of image there is a particular ordering right if we change the order the image will change, but in case of graph there is no sequence as such.

So, if you rename nodes and edges in a graph it will basically change the it will not change the topological structure of a graph. So, therefore, graphs do not have any what should I say any unique way of representation right. Therefore, it is complex it is a complex data structure itself therefore, roughly we roughly define the problem of learning vector representations of various components of a graph as graph representation learning ok.

We will discuss so, since this is the problem right we do not know how to represent a node even write appropriately uniquely we will see how we address these problems in the later part, right. So, when we say that we are embedding a graph to an embedding space we are essentially talking about either embedding a node either embedding edges or even embedding entire graph.

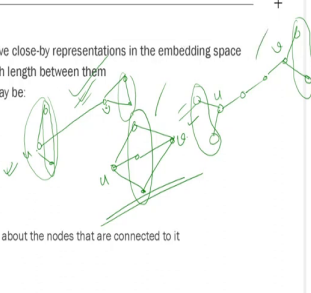
It may be possible that an entire graph is embedded to an embedding space right by a simple single vector. Even we can also think of embedding sub graph or community and so on and so forth.

(Refer Slide Time: 17:27)



Graph Representation Perspectives

- **Node Similarity**
 - Nodes that are similar in the graph should have close-by representations in the embedding space
 - Closeness of two nodes may refer shorter path length between them
 - Closeness in the embedding vectors space may be:
 - Euclidean distance
 - cosine similarity, or
 - any other suitable similarity metrics
- **Neighbourhood structure Similarity**
 - node representations would contain information about the nodes that are connected to it



So, as I mentioned there are two major perspectives how do we capture the similarity between nodes in a graph right we can use either Euclidean distance or say cosine similarity or any other matrices to capture the similarity of you know no representations in an embedding space.

In case of graph you can use say shortest path distance or neighbourhood similarity or two HOPE neighbour similarity and so on and so forth. So, let us say you have a graph like this ok u and v right. So, in this case u and v are directly connected therefore, you can say that ok they are similar because the shortest path distance is 1 in case of, but if you see the neighbours right use neighbours are arranged in a different manner right.

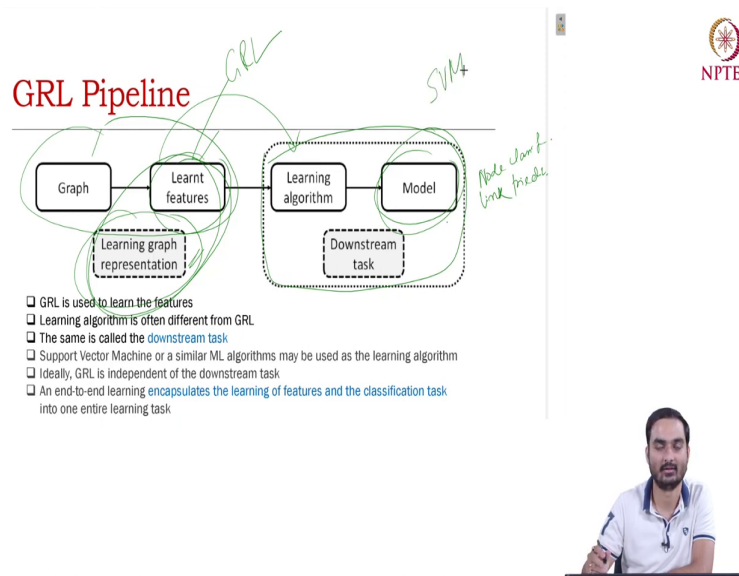
It basically looks like a star whereas, vs neighbour are a basically form a clique right. So, their neighbourhood structures are different, but they are directly connected. So, if you just take the shortest path distance then u and v are similar, but if you take the neighbourhood similarity they may not be similar. Now let us look at here in this case u and v their shortest path distance is 2 right higher than this 1, but if you see the neighbours all their neighbours are common right.

So, one may say that ok u and v are more similar in this case compared to this case right. Think of another cases another example, let us say like this. u and v they are far u and v do not have any common neighbours, but if you look at the neighbourhood structure arrangement both of them have kind of star like arrangement of their neighbours. So, in the

structural if you look at the neighbourhood structure in that aspect I think u and v would be similar.

So, essentially what I am trying to say is that u and v I mean when we take I mean when we consider measuring similarity between a pair of nodes in a graph it is difficult it is very difficult to capture the similarity right sometimes we use shortest path distance. Sometimes we use neighbourhood similarity some sometime we use the distribution of neighbours right and so on and so forth.

(Refer Slide Time: 20:15)



So, this is the overall pipeline of the graph representation. In general if we do not use graph representation learning technique we are given a graph we use manual feature extraction and then we use some model in the downstream task the downstream task can be say node classification right or say link prediction and so on and so forth right.

But if we have if we incorporate Graph Representation Learning GRL this part is now replaced by this one and this module will automatically extract features and those features will be then fed to the next model for the final prediction right. So, this remember this model and this model they are different here essentially this model is GRL right and this model is basically the model that we use for the downstream task. This can be a simple support vector machine or a logistic regression and so on and so forth.

(Refer Slide Time: 21:25)

GRL Pipeline: Components



- Input to GRL pipeline:
 - Homogeneous or heterogeneous graphs
 - Auxiliary information about nodes and edges
- Output to GRL pipeline:
 - Node Embedding
 - Edge Embedding
 - Graph Embedding: only makes sense when we have more than one graph to embed
 - Hybrid Embedding: complex representations of hybrid combinations of nodes and edges

+



So, what is the input to a GRL? So, the input would be either a homogeneous graph or an heterogeneous graph or if you have more information about the nodes it can be auxiliary information like the node feature and edge feature and the output would be either node embedding or edge embedding or graph embedding. In fact, hybrid embedding you can embed a node and a sub graph and so on and so forth in an hierarchical manner on the same embedding space that is also possible.

So, we stop here in the next lecture we start discussing about the actual algorithms for graph representation learning.

Thank you.