Lecture 1 Script

Slide 1

- Welcome to Penn's course on Graph Neural Networks.
- For those of you that don't know me, my name is Alejandro Ribeiro. I am Professor of Electrical and Systems Engineering and I will be in charge of teaching this class.
- If you need to reach me, send me an email at aribeiro@seas.upenn.edu.
- And if you want to know more about my teaching and research, visit my lab's webpage at alelab.seas.upenn.edu.

Slide 2

- If you allow me to begin on a personal note, let me tell you that I have never been more excited on my first day of school than I am today. Graph neural networks are a core component of my research program and you can't get this professor happier than giving him a captive audience to talk about his research. But you should be excited too!
- GNNs are tools with broad applicability and very interesting properties. There is a lot that can be done with them and a lot to learn about them.
- These are therefore the two objectives that I expect we can accomplish together in this course.
- You will learn how to use GNNs in practical applications. That is, you will develop the ability to formulate machine learning problems on graphs using Graph neural networks. You will learn to train them. And You will learn to evaluate them.
- But you will also learn that you cannot use them blindly. You will learn the fundamental principles that explain their good empirical performance. This knowledge will allow you to identify cases where GNN are applicable or not.
- Combining these two abilities, at the end of this course you will be able to identify situations where there is potential in the use of GNNs.

- You will be further able to formulate problems using GNNs to enable the realization of their potential.

- And you will build on your practical experience to develop their solutions.

Slide 3

- This is a terse statement of goals. But is it a relevant statement of goals?

- Well yeah. Most definitely! The reason to care about GNNs and about learning to identify, formulate, and solve problems with GNNs is that GNNs are the tool of choice for machine learning on graphs. And graphs are very common. I could spend several minutes describing problems in which graphs appear, but let me give you four examples that I have personally investigated.

- Graphs appear in authorship attribution problems where we want to identify the author of a text of unknown provenance. They appear because a graph can be used as a simple proxy for grammar.

- Graphs also appear in recommendations systems where the interest is to predict product ratings. And they appear here because a graph can describe similarities in customer preferences.

- A more organic example of a graph is the allocation of resources in a wireless communication network. It is not just that a graph appears. A graph represents a network and the network is itself the object of interest.

- A fourth and very exciting problem is the decentralized control of an autonomous system. In this case, graphs provide a description of the interactions between agents as well as the restrictions on the exchange of information.

Slide 4

- We will spend several weeks talking about machine learning on graphs and graph neural networks.

- But let me share something with you about my mom.

- On every first day of school she would ask me about my day. As I imagine all good mothers do.

- I would therefore hate it if you don't have anything interesting to tell your respective mothers when you give them a call later tonight.

- Thus, allow me to use the next half hour to tell you some interesting things about machine learning on graphs, graph convolutions, and graph neural networks.

Slide 5

- There are, to be concrete, four matters that I want to cover briefly today and that I think will be interesting for you and your moms.

- The first one, is the question of why. Namely, why are we interested in machine learning on graphs.

- I have already told you that this is because graphs appear in scores of problems. I gave you four concrete examples.

- But I want to elaborate on the notion of structure and on how graphs are used to model the structure of signals.

- The second one, is the question of how. How do we do machine learning on graphs?

- The obvious answer is that we should use a neural network. Not a graph neural network. Just a neural network.

- The drawback of this obvious answer is that fully connected neural networks do not scale beyond signals with a small number of entries. We will not dwell on why this happens. But I want you to end the day aware of this fact.

- A related truth is that the way to achieve scalability is to use convolutions. Whether they be in time or graphs. This has to do with leveraging symmetries induced by structure, something that convolutions attain by construction. Again, we are not going to dwell on this. But I want you to end the day aware of this truth.

- The third item that I want to cover today, is the definition of convolutional filters in Euclidean space and convolutional filters on graphs. Both of these are very simple to understand. I would be negligent if I didn’t explain them to you in our first lecture.

- The fourth matter I want to discuss is the definition of convolutional neural networks and graph neural networks. As in the case of filters, both of these are also very simple to understand. I would be equally negligent if I didn’t explain them to you in our first lecture.

Slide 6

- Let us dwell on the whys of machine learning on graphs. Why is it interesting? Why do we care? At the risk of being repetitive, we care because graphs are pervasive in information processing.

Slide 7

- This is perhaps then an appropriate reformulation of the question of why. Why is it that graphs are so pervasive in signal and information processing? Well, the reason why these models are so common is because graphs are generic models of signal structure, and this structure that the graph codifies can help learning in many practical situations.

- To exemplify the meaning of this statement, suppose we are interested in an authorship attribution problem where the goal is to identify the author of a text of unknown provenance.

- Or perhaps we are interested in a recommendation system where the goal is to predict the rating that a customer would give to a certain product.

- In both cases, and although this is not immediately apparent, there exist graphs that contain information that is meaningful about the problem that we want to solve.

Slide 8

- To further dwell on this statement, let's take a closer look at authorship attribution. This is a problem we can tackle with word adjacency networks.

- These word adjacency networks are graphs in which nodes represent different function words and edges represent how often a given author utilizes each pair of words together.

- The critical point to be highlighted here is that function words are those that don't carry meaning. They are words like prepositions or conjunctions, the words we use to put other words together. It is therefore not difficult to accept that they may serve as a good proxy for the different ways in which different authors leverage or exploit the grammar of the English language in their writing.

- For example, the figure on the left is the word adjacency network built from the cannon of William Shakespeare plays.

- The figure on the right is a word adjacency network built from the canon of Christopher Marlowe. The first thing we notice is that both of these word adjacency networks look a lot like each other. This is not surprising. Both of these are English playwrights of the same period, after all.

- However, if we squint more closely, we can see that there are sufficient differences in these two networks to tell the writing styles of Shakespeare and Marlowe apart. This is not only a fun fact but it enables us to objectively ascertain their collaboration in the writing of Henry the sixth.

- But this is not the point I want to illustrate today. The point that I want to drive is that even though it didn't look like that a priory, it turns out that a graph plays an enabling role in authorship attribution problems.

Slide 9

- The other problem we highlighted was a recommendation system. The particular approach we advocate is collaborative filtering.

- In collaborative filtering, we build a graph in which nodes represent different customers and edges are similarity scores between pairs of users that we build from the rating histories of customers.

- These user similarity graphs are the basis for predicting ratings in situations where a certain product has been rated by some customers but has not been rated by others.

- The diagram on the left represents original ratings in which some customers have rated the product, but some others have not. This figure is not a graph but a variation diagram, in which edge strengths are proportional to the product of the user similarity score and the difference between their ratings for the given product. The abundance of strong edges highlights the presence of large rating dissimilarities between customers that typically tend to give similar scores. This large variability is not an inherent property of the signal. It is a consequence of the fact that the product has not yet been rated by certain customers.

- We can therefore make rating predictions by eliminating large variability. We do that in the figure on the right.

- This figure shows a reduction in the variation energy pre and post prediction and the signal that it represents can be seen to provide good rating estimates. But, again, the

point here is not to discuss the merits of collaborative filtering. I am trying to illustrate that even though it didn't look like that a priory, it turns out that a graph plays an enabling role in rating predictions.

Slide 10

- In recommendation systems and authorship attribution, graphs represent a data structure. But graphs are more than data structures.

- In several applications, graphs are an inherent part of the system. This is the case of physical systems with multiple agents in which graphs model the locality of interactions between agents.

- To give an example, this video illustrates the decentralized control of an autonomous system where the goal is to coordinate a team of drones without relying on central coordination. The graph appears the moment we forgo central coordination and opt for decentralized control. When we do that, physical proximity becomes an integral part of the system because it determines the information that is available to particular agents. This physical proximity, which is inherent, indeed, central, to decentralized control, is modeled by a graph.

- Another example of a multiagent physical systems is a wireless communication network. The goal here is to manage interference when allocating bandwidth and power. Interference is affected by the radio propagation environment which determines how much of the transmitted power makes it to several intended and unintended receivers. This interference is inherent to wireless networks and is modeled with an interference graph.

- An important observation to make in both of these examples is that it's not only the graph instrumental in finding a solution, the graph is itself the source of the problem. The reason why decentralized control and wireless resource allocation are challenging is because the graph creates a tension between the acquisition of local information and the achievement of goals that are inherently global. This is a feature shared by all multiagent physical systems.

Slide 11

- We have seen four interesting examples of machine learning on graphs. Two of which involve graphs as data structures and two that involve graphs as models of multiagent physical systems. This answers the question of why we want to do machine learning on graphs. And leads to the question of how to do machine learning on graphs.

Slide 12

- The answer to the question of how is pretty easy: We should use a neural network. We should do this, because we have overwhelming empirical and theoretical evidence for the value of neural networks. Understanding this evidence is one of the objectives of this course. But before we are ready to do that, there is a dealbreaker challenge potentially lurking in the shadows.

- That challenge is the fact that we want to run a neural network over a structure like this graph

- But we have become good at running neural networks over images that look like this other graph.

- Generic neural networks, or fully connected neural networks to be more precise, do not scale as we grow the dimensionality of the input signal. If we have a signal made up of a small number of components, a generic neural network suffices for its processing. However, if we have a signal with a large number of components, which in the case of graphs means a signal supported on a large graph, a generic neural network will not work.

- Do we know how to overcome this issue? Well, in the case of images and signals in time, we know that convolutional neural networks succeed at scaling. That's what I meant when I said that we are good at running neural networks on images. We can make them scale.

Slide 13

- That we know how to scale is good news. Alas, we can process images with convolutional neural networks, but we cannot process graphs or graph signals. But if we look deeper at CNNs, an intellectual roadmap arises.

- CNNs are made up of layers, each of which is a composition of a convolutional filter bank with a pointwise nonlinearity. In this definition, the notion of layer and the notion of pointwise nonlinearity is not specific to images. They can easily be generalized to arbitrary graphs. It is the notion of a convolutional filter bank that we do not know how to generalize to graphs. Out of these observations we find our roadmap.

- If we generalize convolutions to graphs, we can easily create graph filter banks

- Which we can easily combine with pointwise nonlinearities.

- And which we can easily stack in layers to create a graph neural network.

- This yields the GNN architecture that we will use to process graphs and graphs signals.

- Which we hope will be as successful and scalable as CNNs are for the processing of images and time signals.

Slide 14

- Our intellectual path towards scalable machine learning on graphs begins from the construction of generalisations of the convolution operator to signals supported on graphs.

- We will build this generalization by observing that even though we do not often think of them as such, convolutions are operations on graphs.

Slide 15

- In order to express convolutions as operations on graphs, begin by observing that we can always describe discrete time and space using graphs that support either time or space signals.

- Indeed, consider a time signal **x** with components x_i, and associate individual components with individual nodes of a directed line graph. A directed line graph is a good description of the proximity and causality of time, and it is therefore also a good description for the underlying structure of the signal **x**.

- Likewise, consider an image defined as a signal **x** with components x_{ij} and associate each of these components with a node of a grid graph. The grid graph is a good description of the local structure of the plane, and it is therefore also a good description of the underlying structure of the signal **x**.

Slide 16

- That we can describe time and space using graphs is an almost trivial observation, but one that nonetheless has interesting conclusions. Out of this, the one that is germane to our current discussion is that we can use line and grid graphs to write convolutions as polynomials on their respective adjacency matrices.

- Let us begin with the case of a signal supported in time and suppose that we want to implement a convolutional filter with coefficients h_k. We know that the output of such a filter is a weighted linear combination of shifted versions of the input signal **x**.

- The first term of the convolution sum is just the signal **x** scaled by coefficient h_0.

- The second term of the convolution sum is a shifted version of the signal **x,** scaled by coefficient h_1**.** But we can obtain this shifted version of **x** by multiplication with the adjacency matrix **S** of the line graph.

- The third element of the convolution sum is a twice-shifted version of the signal **x** scaled by coefficient h_2. But we can obtain this twice-shifted version of **x** through multiplication by the square of the adjacency matrix **S** of the line graph.

- The next term is a thrice-shifted version of the signal **x** scaled by coefficient h_3. Which we obtain by multiplying the original signal **x** by the adjacency matrix of the line graph three times.

- We keep adding terms to the convolution sum until the order of the filter so that in the end we write the convolution as a premultiplication of the signal **x** by a polynomial on the adjacency matrix of the line graph modulated by coefficients h_k. This illustrates that we can write the convolution in time as a polynomial on the shift operator of the directed line graph.

- Pretty much the same holds true for images, except that now the polynomial is on the adjacency matrix of the grid graph. This is true because we can write a spatial filter as a linear combination of diffused versions of the input signal. And we can obtain these diffused versions by multiplying the original signal **x** by subsequent powers of the adjacency matrix of the grid graph.

- We begin by adding the signal modulated by coefficient h_0

- And we further add a diffused version of the signal **x** modulated by coefficient h_1. This diffused version of **x** can be obtained by multiplying the input **x** with the adjacency matrix **S** of the grid graph.

- The next entry of the convolution sum is a diffusion of the diffused signal, scaled by coefficient h_2. Which we can obtain through a second multiplication with the adjacency matrix **S.**

- The fourth term of the convolution sum is a diffusion of the diffusion of the diffused signal, scaled by coefficient h_3. We can obtain this thrice-diffused signal by multiplying **x** with the third power of **S**.

- We keep adding terms to this sum as required by the order of the filter. We end up with the convolution expressed as a premultiplication of the signal **x** by a polynomial on the adjacency matrix of the grid graph modulated by coefficients h_k.

Slide 17

- Pervasive and important though they are, time and space signals constitute a rather limited class.

- Their interpretation as graph signals, however, hints that we can use graphs as generic descriptors of signal structure.

- In which signal values

- Are associated to nodes of a graph

- And edges of the graph express an expectation of similarity between signal components.

- (Empty)

- For instance, nodes can represent customers. Signal values, product ratings. And edges are cosine similarities between past scores. This provides an appropriate description of the types of signals that appear in recommendation systems.

- Alternatively, nodes can be drones. Signal values, their velocities. And edges can represent sensing and communication ranges. This is an appropriate description of a decentralized autonomous system.

- Or we can have nodes representing transceivers. Signal values representing quality of service requirements. And edges representing wireless channel strength. This is an appropriate description of a wireless communication system. Graphs can therefore provide a significant expansion of the class of signals that we can process. But we have seen this already.

Slide 18

- Our concern here is on how to design convolutional processing of graphs signals. In doing so, we start from the observation that at least in two cases, we already know what to do.
- If we are given a graph signal supported on a line graph
- We know how to build
- A polynomial
- On that adjacency matrix
- That represents
- The time convolution.
- If we are given a signal supported on a grid graph,
- We know how to build
- A polynomial
- On this other adjacency matrix
- That represents
- The convolution of signals in space. But it is not only that we know how to build these polynomials. These polynomials are the same.
- (Empty)

Slide 19

- We can therefore proceed by analogy. If we are given a signal supported on a certain graph, we define a graph convolutional filter to process such a signal, as a polynomial on a matrix representation of the graph that supports the signal.
- To illustrate this idea, consider it a signal supported on the graph shown on the left and suppose that we want to run a convolutional graph filter with coefficients h_k. The output

of this convolutional graph filter is defined as a summation of diffused versions of the input signal **x**, scaled by respective coefficients.

- We begin by adding the signal **x** itself modulated by coefficient h_0.
- We then add a diffused version of the signal **x** modulated by coefficient h_1
- A twice-diffused version of **x** modulated by coefficient h_2
- A thrice-diffused version of the input **x** scaled by coefficient h_3. And so on.
- In the end, the graph convolution is expressed as a premultiplication of the signal **x** with a polynomial on a matrix representation of this graph modulated by coefficients h_k.
- What happens if we change the graph? This yields a different graph signal supported on a different graph, but the expression for the graph convolutional filter is the same.
- We begin by adding the signal itself.
- We then add a diffused version of the signal **x** modulated by a different coefficient.
- We further add a twice-diffused version of the signal **x**
- And a thrice-diffused version of the signal **x**. And whatever number of times are required by the convolutional filter order.
- In the end, we have the same expression as before in which the signal **x** is premultiplied by a polynomial on the matrix representation of the graph **S** modulated by coefficients h_k. It goes without saying that the expressions for the convolutions are the same in both graphs but the resulting convolutions can be quite different. This is, of course, because the graphs, represented by **S**, can be quite different.
- As highlighted by our illustrations, multiplication of a graph signal with a matrix representation of the graph on which it is supported is a local operation. This is a feature that graph convolutions share with conventional convolutions in time and space, and that underlies their practical value. As we will see throughout the next few weeks.

Slide 20

- To enable machine learning on graphs, we constructed an intellectual roadmap that began with a generalisation of convolutions to graphs and continued with a

generalisation of convolutional neural networks to graph neural networks. We have completed the first part of the roadmap.

- The second part of the roadmap is easier because CNNs and GNNs are minor variations of linear convolutional filters.

- We just need to add pointwise nonlinearities and compositions of several layers.

Slide 21

- Before we define convolutional versions, we define plain, or fully connected neural networks. A neural network is a composition of a cascade of layers.

- Each of which is itself the composition of a linear map **H** with a pointwise nonlinearity \sigma. The first layer takes a signal **x** as an input and it processes that with a linear function and a pointwise nonlinearity to produce an intermediate output. Intermediate layers, like layer two in the diagram, take the output of the previous layer as an input and again, they process that with a linear function and a pointwise nonlinearity to produce another intermediate output. This compositional process is repeated until the last layer where the output of the layer is declared to be the output of the neural network.

- Neural networks are generic information processing architectures in the sense that they can be applied to any type of signal **x**. However, as it often happens with generic architectures, it does not scale well with the dimensionality of the input signal.

Slide 22

- To design a scalable architecture, we resort to convolutions. A convolutional neural network is thus defined as a neural network in which the linear maps that are used at each layer are required to be convolutional filters. We then end up with an architecture that again composes a cascade of layers, but these layers are now different.

- They are a composition of a convolutional filter with a pointwise nonlinearity.

- Convolutional neural networks are the workhorse of deep learning. This provides ample empirical evidence that they do scale well with the dimensionality of the input signal **x**.

- Empirical evidence aside, the scalability of a CNN is suspected because a CNN is a minor variation of a convolutional filter.

- We just add nonlinearities and compositions.

- And we know that convolutions scale well. We have more than a century of evidence that this is true. But of course, we do pay a price. The price we pay is that this is not a generic architecture anymore. It is an architecture that is restricted to the processing of signals in time, if we use time convolutional filters. Or to the processing of images, if we use spatial convolutional filters. And as we have said before, time signals and images are pervasive and important, but nevertheless a limited class. To build a more generic architecture, we resort to the use of graph convolutions.

Slide 23

- We have seen already that the key to defining graph convolutions is to think of time signals as signals that are supported on a line graph. When we do that in a CNN, all of the convolutions that appear in different layers become polynomials on the adjacency matrix of the line graph. We end up with the architecture we show in this diagram in which we compose layers, each of which is a composition of a linear transformation with a pointwise nonlinearity. But the linear maps are now defined through polynomials on the adjacency matrix of the line graph.

- That is, this architecture replaces the convolutions we have in the CNN

- By polynomials on the adjacency matrix of the line graph.

- But this is not a substitution. This is just a different way of writing convolutions and CNNs,

- But one that lends itself to generalization to arbitrary graphs.

Slide 24

- This brings us to the introduction of graph neural networks. We simply get this alternative representation of CNNs and we let the matrix **S** be the representation of an arbitrary graph.

- The convolutional filters are now arbitrary graph convolutional filters, which depend on the specific graph. The most remarkable fact about this modification is that it is not a modification. It's just the same thing. A graph neural network is just a convolutional neural network with a different graph. This not modification, however, does make a lot of difference in practice. Because we have now extended the reach of scalable neural networks to signals that are supported on arbitrary graphs.

Slide 25

- This is then how a graph neural network looks like. It is the composition of a cascade of layers.

- Each of which is itself also a composition of a graph convolution with a pointwise nonlinearity. The graph convolution is a polynomial on a matrix representation of a certain given graph.

- We can therefore think of a GNN as a neural network in which the linear transformations that are used at each layer are restricted to be graph convolutional filters.

- As per this definition, we can also think of graph neural networks as generalisations of convolutional neural networks in the sense that CNNs can be recovered from GNNs if we particularize the graph to a line graph.

Slide 26

- Although the matter is not as settled as in the case of CNNs, there is growing evidence for the scalability of GNNs.

- But empirical evidence aside, the reason why GNNs are promising is because GNNs are minor variations of graph filters. In the exact same sense in which convolutional neural networks are minor variations of convolutional filters.

- We just add nonlinearities and layer compositions.

- As we will explore in this course, both of them, graph convolutional filters and graph neural networks, are expected to scale because they leverage the signal structure that is codified by the underlying graph.

Slide 27

- We are reaching the end of lecture 1. Let’s close with a summary of the road ahead.

Slide 28

- When we began the lecture I told you that there were two goals for this course.

- To learn how to use Graph Neural Networks

- And to understand their fundamental properties

- But there is obviously a third goal. You could call it Goal Zero.

- Which is to define GNNs. Or to define GNN architectures to use the lingo.

Slide 29

- In today's lecture I told you a lot about architectures. I defined convolutions in time and convolutions on graphs. And I explained how these convolutions can be used to construct CNNs and GNNs, which are the basis for scalable machine learning. This was just to give you a taste. As I said, for you to have something interesting to tell your mothers.

- If you didn't understand today's lecture, do not worry. We will revisit graph filters and graph neural networks. We will take it more slowly to explain them better.

- We will also introduce other related architectures. Notably, graph recurrent neural networks.

- In future lectures I will also dive on the fundamental properties of GNNs. This is because I oppose the blind use of tools. Using tools we don't understand, leads to unpleasant results more often than not. In the particular case of GNNs there are concrete properties that explain why they work. More importantly, these properties separate problems where GNNs are warranted and expected to work. From cases where GNNs are either unwarranted or expected to fail.

- These properties are permutation equivariance.

- Stability to deformations.

- And transferability across scales

- Here you have a recent paper if you want to start reading ahead.

Slide 30

- In parallel with lectures, we will have five labs focused on building practical skills. By which I mean developing the ability to set up and train GNNs for solving practical problems.

- Lab 1 is about Statistical and Empirical Risk Minimization. This is Just a warmup to get you used to the programming environment. It is focused on the notion of learning parameterizations and on how they are necessary in machine learning.

- Lab 2 illustrates the use of GNNs in recommendation systems. We will learn to predict ratings with fully connected neural networks, graph filters, and GNNs. To understand the differences between them. This will be a motivating example for the introduction of fundamental properties of GNNs.

- Lab 3 considers the Learning of Controllers in Distributed Collaborative Intelligent Systems. It is our first approach to the use of GNNs in multiagent physical systems. It will illustrate the role of GNNs in learning policies that have a natural distributed implementation.

- Lab 4 is concerned with Learning Resource Allocations in Wireless Communication Networks. The goal of this lab will be to illustrate situations in which the graph itself is an input to the GNN.

- Lab 5 is a sort of capstone project. We will solve a path planning problem in a multirobot system in which robots are given local maps of their environment and have to collaborate to learn trajectories to target destinations.

- Here you have respective references if you want to start reading ahead.

Slide 31

- This is all for today. I am looking forward to working with you. And don’t forget to give your mom a call!