Lecture 5 Script

1 Permutation Equivariance of Graph Filters

Slide 1: Permutation Equivariance of Graph Filters - Title Page

(1) Intuition indicates that graph filters and GNNs are permutation equivariant. We have encountered this possibility several times. In this lecture we formally define and prove the permutation equivariance of graph filters

Slide 2: Permutation Matrices

(1) We begin by introducing the notion of a permutation matrix to enable representation of permutations of graph signals and their shift operators.

(2) A square matrix P of dimension n is said to be a permutation matrix.

(3) If it has binary entries so that each P_i-j is either 0 or 1.

(4) And it further satisfies that the product of P and P transpose with the all one vector is also the all one vector.

(5) This definition is such that the product P transpose times x reorders the entries of the vector x. It doesn't change them.

(6) And it is also such that the product P transpose S P is a permutation of the rows and columns of matrix S. This permutation is consistent with the permutation of x induced by the product P transpose x.

(7) Since P is a binary matrix and the product of P and P transpose times the all ones vector is equal to the all ones vectors. It follows that P has exactly one entry equal to 1 in each of its rows and in each of its columns.

(8) Their definition also implies that permutation matrices are unitary. This means that the product of a permutation P with its transpose equals the identity. In turn, this fact implies that multiplication with P transpose undoes the reordering implemented with multiplication by P.

Slide 3: Relabeling of Graph Signals

(1) In the case of graph signals, permutation matrices implement relabeling. More to the point, recall that graph signals are pairs S comma x which include the shift operator S in which the signal x is supported. Although we often leave the shift operator implicit.

(2) Given this reminder consider now the graph signal P-transpose-times-S-times-P coma P -transpose times-x. This is a signal in which both, the graph and the signal are permuted. This signal is a relabeling of S-x.

(3) But it is otherwise the same signal.

(4) We just changed the names of the nodes.

(5) To be more clear, consider an illustration of a graph signal x supported on graph S. As usual, the colors signify different signal values.

(6) This other illustration is the signal P-transpose-x supported on the graph P-transpose-S-P. The only difference between the one and the other are the names we assign to nodes.

(7) Otherwise, they are still the same signal. Their names have changed. Their values have not.

(8) This creates an important requirement. If the signals are the same except for their labeling, their processing should be label independent. This is a requirement that graph filters and GNNs fulfill because of their permutation equivariance.

Slide 4: Graph Filters and the Permutation of Graph Signals

(1) To study permutation equivariance of graph filters recall the definition of graph filter H of S as a polynomial on the shift operator S with coefficients h sub k. The output of the filter H-of-S when presented with input signal x is the product H of S times x.

(2) The same filter can be run on different signals but also on different graphs. The latter fact is what we call transference. In this video we are interested in running the filter on the signal x supported on S and on the signal x hat supported on S hat. Where x hat and S hat are permutations of the signal x and its shift operator S.

(3) For the signal x supported on S, we have the filtering operation H of S times x represented as a polynomial on variable S multiplying x

(4) And for the for the permuted signal x-hat supported S-hat, we have the filtering operation H of S hat times x hat represented as a polynomial on variable S hat multiplying x hat.

(5) The filter H-of-S applied to x uses coefficients h_k. It takes input signal x. And instantiates the filter polynomial on shift S.

(6) The filter H-of-S-hat applied to x-hat uses the same coefficients h_k. But it takes the permuted signal x-hat as an input. And instantiates the filter polynomial on the permuted shift operator S-hat.

Slide 5: Permutation Equivariance of Graph Filters

(1) Given these formal definitions of a filter and its permutation, it is ready to prove a theorem stating the permutation equivariance of graph filters.

(2) Consider then consistent permutations of the shift operator S and the signal x. Defined by the products P-transpose-S-P and P-transpose-x.

(3) We then have that.

(4) The output of the filter applied to the permuted signal supported on the permuted shift

(5) Is equal to a permutation of the output of the same filter applied to the original non-permuted signal supported on the original non-permuted graph.

(6) We say that graph filters are equivariant to permutations. Because a permutation of the input and the shift operator induces the same permutation at the output. The use of the word **equi**-variant comes from the fact that it is **equi-**valent to permute a signal at the input of a filter, which implies permuting the signal and the shift, as it is to permute the signal at the output of a filter. Which also implies permuting the signal and the shift on which it is supported. Permuting and running a filter is equivalent to running the filter and permuting the output afterwards.

Slide 6: Proof or Permutation Equivariance of Graph Filters

(1) The proof of this theorem is very simple. Start with the definition of the filter applied to the permuted input.

(2) And substitute in this expression the definition of S hat. Which is equal to P transpose times S times P.

(3) And the definition of x hat. Which is equal to P transpose times x

(4) The key observation, if we can call it that, is that in the k-th power of P-transpose-S-P.

(5) The permutation matrices P and P transpose cancel each other out. This is because they are unitary. Their product is an identity. Which, conceptually, holds because P and P-transpose implement inverse permutations.

(6) This cancellation reduces the k-th power of P-transpose-S-P. To P-transpose times S-to-the-k times P.

(7) To complete the proof we substitute this identity into the filter's output expression. When we do that, another product of the form P-times-P-transpose appears because of the permutation of the input.

(8) We cancel that out since we know it is equal to an identity.

(9) And factor the matrix P transpose out of the sum.

(10) To identify the appearance of the filter applied to the original non-permuted input.

(11) Which is what was to be shown.

Slide 7: Signal Processing with Graph Filters is Independent of Labeling

(1) We requested the processing of graph signals to be independent from the labeling of the nodes. We have now shown that graph filters satisfy this request. To make sure this is clear we show on the left a graph signal x supported on S. The same one we considered a few minutes ago.

(2) On the right, we show the result of applying a permutation P to the signal and the shift operator. Which, as we have already explained, is a mathematical representation of node relabeling. It changes the names of the signal components. But it does not change their values or their relationships with each other.

(3) Suppose now that we apply a filter to the original input signal. The specific result is some sort of diffusion. But it could be any other transformation that is implementable with a graph filter.

(4) The requirement to have graph signal processing that is independent of labels, implies that the output of running the same filter on the permuted signal, should be the corresponding permutation of the same signal. This is what we illustrate on the right. It's

the same signal on the left but with different labels. We emphasize that this is not a result. This is a requirement. We want this to happen.

(5) The permutation equivariance theorem says that this requirement is fulfilled by graph filters.

(6) Indeed, we know that the result of running the filter on the permuted graph with the permuted signal as an input,

(7) Results in an output that is a consistent permutation.

(8) Of the output produced when running the filter on the original non-permuted signal and original non-permuted graph. That is, the output on the right has to be a relabeling of the output on the left. Which is the request we had made.

2 Permutation Equivariance of Graph Neural Networks

Slide 8: Permutation Equivariance of Graph Neural Networks - Title Page

(9) We move on to discuss permutation equivariance of graph neural networks. We will show that GNNs inherit the permutation equivariance of graph filters.

Slide 9: Graph Neural Networks and the Permutation of Graph Signals

(1) We consider GNNs with single features. The extension of the analysis to multiple features is straightforward. Start by recalling the definition of a GNN with L layers.

(2) This is an architectures that consists of L layers that recursively process outputs from previous layers. Each layer is made up of a perceptron, which is the composition of a pointwise nonlinearity with a graph filter with coefficients h_k

(3) The output of the GNN is the output of layer L. It is a function of the input signal, the shift operator, and the filter tensor calligraphic H which groups all of the filter coefficients used at each layer. We use Phi to denote the GNN operator.

(4) We consider here the result of running the GNN defined by the filter tensor H on the signal x supported on S and on the permuted signal x-hat supported on the consistently

permuted shift S-hat. That is, the permuted signal is P-transpose-x and the permuted shift is P-transpose-S-P.

(5) To be clear, we want to understand the relationship between the output of the GNN operator Phi that uses the filter tensor H. Takes the signal x as an input. And instantiates the filters of each layer on the shift operator S.

(6) And the output of the GNN operator Phi that uses the **same** filter tensor H. But takes as input the permuted signal x-hat. And instantiates the filters on the permuted shift operator S-hat.

Slide 10: Permutation Equivariance of Graph Neural Networks

(1) The relationship between these outputs is stated by the permutation equivariance theorem of graph neural networks.

(2) Consider then graph signal x supported on shift operator S and its permuted version x-hat supported on shift operator S-hat.

(3) It then holds that.

(4) The output of the GNN Phi that uses filter tensor H and runs on the permuted signal x-hat and permuted shift operator S-hat.

(5) Is the corresponding permutation of the output of the GNN Phi that uses the same filter tensor H and runs on the original non-permuted signal x and the original non-permuted shift operator S.

(6) This is the same theorem that holds for graph filters but restated for GNNs. Its implication is the analogous. Running a GNN after we permute the input and the shift is equivalent to first running the GNN and permuting the output afterwards.

Slide 11: Proof or Permutation Equivariance of Graph Neural Networks

(1) The theorem holds because we know that filters are equivariant to permutations. And since the nonlinearities are pointwise, there is nothing they can do to alter the equivariance of the filters. To formalize this argument recall that we are running two separate GNNs. At layer ell of the one that runs on shift operator S we have the output x_ell written as a perceptron applied to the signal x_ell-minus-one. This perceptron combines the pointwise nonlinearity sigma with the graph filter H_ell of S.

(2) In Layer ell of the GNN that runs on shift operator S-hat we have the output x_ell-hat written as a perceptron applied to the signal x-hat_ell-minus-one. This perceptron combines pointwise nonlinearity sigma with the graph filter H_ell of S-hat. The graph filter has the same coefficients.

(3) Proceed now to make the induction hypothesis that at layer ell, the inputs are permutations of each other.

(4) Write down the output of layer ell on the GNN that uses shift operator S-hat. The perceptron involves the filter H_ell of S-hat applied to signal x-hat_ell_minus one

(5) Given that the filter is permutation equivariant, which we have proven before, and that we are assuming by induction hypothesis that the inputs are permutations of each other, it follows that the output of this filter can be written as a permutation of the output of the filter H_ell of S applied to signal x_ell-minus-one.

(6) Furthermore, given that the nonlinearity is pointwise, the permutation matrix, that is currently applied before the nonlinearity, can be applied after the nonlinearity is taken.

(7) We can now identify that we are left with a permutation of the output of the perceptron that characterizes the Layer L output of the GNN that runs with shift operator S.

(8) We have proven that layer ell is permutation equivariant. More importantly for the purposes of a formal proof is that we have completed an induction step. We have proven that if the inputs to layers ell are related by a permutation P-transpose, their outputs are related by P-transpose as well. But this output is the input to layer ell-plus-one. Thus, inputs to layer ell-plus one are related by permutation P-transpose if inputs to layer ell are related by permutation P-transpose.

(9) To complete an induction proof, notice that at Layer 1 we have x hat equal to P transpose times x. This is the hypothesis of the theorem.

(10) The induction is now complete.

(11) The proof is done.

Slide 12: Signal Processing with GNNs is Independent of Labeling

(1) Graph neural networks, same as graph filters, perform label-independent processing. This is because the nonlinear function acts separately on each component. The implication is the exact same implication we discussed for graph filters.

(2) If we consider the signal on the right and a consistent permutation of the signal and the shift, all we have done is change the labels of the nodes.

(3) Thus, we **require** the processing to be impervious to this change. Whatever transformation happens on the non-permuted signal on the left, should be replicated on the right. GNNs, same as graph filters, **fulfill** this requirement. This is what the equivariance theorem proves.

Slide 13: Equivariance to Permutations and Signal Symmetries

(1) Another implication of permutation equivariance, which also holds for both, graphs filters and GNNs, is the exploitation of symmetries in graphs and graph signals. We have discussed this at an intuitive level, but we can now state it formally.

(2) A graph symmetry exists when the graph can be permuted onto itself. Some graphs are invariant under the action of a given permutation. This means it is possible to rewrite S as P-transpose-S-P for some permutation P. This is what we show in these two figures. The graphs are drawn to be literally the same. But we have applied a permutation that implements a specular symmetry that moves node 9 into node 12, node 6 into node 3 and so on.

(3) If we apply the equivariance theorem to this particular case.

(4) We know that a permutation of the shift and the signal applied at the input of the GNN.

(5) Is the same as a permutation of the signal (and the shift) applied at the output of the GNN.

(6) But we also know that the shift permutation is moot because the shift operator is being permuted onto itself.

(7) We have therefore shown that when a graph is symmetric, the GNN output associated with a permutation of the input x is equally to a permutation of the output of the GNN. But without permuting the shift. Which we don't have to permute because it is symmetric.

(8) We van therefore claim that a GNN learns to process the permuted signal P-transpose-x supported on S

(9) From learning the processing of the non-permuted signal x. Which is also supported on x. This effectively multiplies the size of the dataset. Because the signal x is doubling for the signal P-transpose-x. And for any other signal that has the same symmetry.

Slide 14: Symmetry is Rare but Quasi-Symmetry is Common

(1) An important point to make is that perfect symmetries are rare. Regular graphs are nice abstractions but do not appear often in practice. But we do have situations where graphs are close to being symmetric. As we illustrate here.

(2) The graph is a perturbed version of a permutation of itself. Thus, there is no interest in exploiting perfect symmetries. But there is interest in exploiting quasi-symmetries. If this were a recommendation system, our user similarity networks are **not identical.** But some of us will have user similarity networks that are **close.**

(3) The ability to exploit quasi-symmetries hinges on our ability to establish stability to deformations.

(4) So that we have approximate equivariance when we have graphs that are close to be permutations of each other. And so that we have approximate-equivariance for graphs that exhibit approximate-symmetries.

Slide 15: Operator Distance Modulo Permutation

(1) This brings to the notion of operator distance modulo permutation. To measure how far operators are to being permutation equivariant.

(2) Consider then given operators Psi and Psi hat.

(3) The operator distance modulo permutation between Psi and Psi hat.

(4) Is defined as: The minimum over permutations.

(5) Of the maximum over vectors x with unit norm.

(6) Of the norm of of the difference between:

(7) P transpose times Psi applied to x.

(8) And Psi hat applied to P transpose times x. This definition is a variation of the operator norm in which on top of comparing the action of the operator on different unit norm vectors, we also search for the permutation matrix that makes the operators as close to each other as possible.

(9) The operator distance modulo permutation permits rewriting our equivariance theorems. For the case of graph filters, we can restate the theorem as claiming that

(10) If the distance modulo permutation of S and S hat is zero

(11) Then the distance modulo permutation of H of S hat and H of S must be zero as well

(12) Likewise the equivariance theorem of GNNs can be restated as claiming that

(13) If the distance modulo permutations of S hat and S is zero

(14) Then the distance modulo permutations of the GNN operators Phi that run on graphs S and S-hat with the same tensor must be zero too.

(15) What happens now when the distance modulo permutations of S hat and S is small but not zero? Namely, when S and S-hat are not permutations of each other but **close** to permutations of each other?

(16) This is the problem we study under the banner of stability properties of graph filters and GNNs.

3 Lipschitz and Integral Lipschitz Filters

Slide 16: Lipschitz and Integral Lipschitz Filters - Title Page

(1) Graph filters are the learnable parameters of GNNs. We illustrate how different classes of filters may have different discriminability properties. We focus on Lipschitz and Integral Lipschitz filters.

Slide 17: Graph Convolutional Filters

(1) To start on the same page, recall that for given graph shift operator S and coefficients h_k, a graph filter is a polynomial on the shift operator modulated by coefficients h_k.

(2) Graph convolutional filters can also be characterized in the graph frequency domain by their frequency response tilde h of lambda. The frequency response is a polynomial on a scalar variable lambda that uses the same coefficients h_k. The variable lambda represents the graph frequency domain.

(3) An important property of the frequency response is that it is independent of the graph. It is completely characterized by the filter coefficients h_k. The effect of a particular graph

is to specify the frequencies on which the response is to be instantiated. This is determined by the eigenvalues of the graph.

(4) In any event, in this video we are studying filters in the abstract. Without reference to a specific graph. It suffices for us to look at their frequency responses. These are just analytic functions represented by a polynomial series on the scalar variable lambda, modulated by coefficients h_k

Slide 18: Lipschitz Filters

(1) The first class of filters that we introduce is that of Lipschitz filters.

(2) Given a graph filter with coefficients h_k.

(3) And a graph frequency response tide h of lambda, written as a polynomial with coefficient h_k and scalar variable lambda.

(4) We say that the filter is Lipschitz.

(5) If, there exists a positive finite constant C.

(6) Such that for **any** pair of arguments lambda 1 and lambda 2,

(7) The absolute value of the difference between the frequency response of the filter evaluated at frequencies lambda 1 and lambda 2 is bounded by the Lipschitz constant C multiplied by the the absolute value of the difference between lambda 1 and lambda 2.

(8) This definition entails that for Lipschitz filters, the change in values of the frequency response is at most linear with rate C.

(9) Taking the limit of the Lipschitz condition as the difference lambda 2 - lambda 1 approaches zero, we see that the Lipschitz condition implies that the derivative of the frequency response tilde-h-prime of lambda is bounded by C for all lambda. This derivative exists always because graph filters are analytic functions.

Slide 19: Discriminability of Lipschitz Filters

(1) As it is indicated by their names, the frequency response of a Lipschitz filter is Lipschitz continuous. With a slope that does not exceeds C. The figure illustrates a Lipschitz filter. But the figure doesn't say much. This is because almost any filter is Lipschitz. We just

need to avoid infinite slopes. Which really, it is more difficult to have an infinite slope than it is to avoid one.

(2) What really matters is the value of the Lipschitz constant. In this figure we highlight in different colors a couple of filters with a small constant C and a couple of filters with a large constant C.

(3) The filters with small C vary slowly

(4) And the filters with larger C vary more rapidly.

(5) A conclusion to draw from this picture is that the Lipschitz constant of the filter is related to its discriminability.

(6) If the filter has a small C. Like the filters in red, it allows passage of a number of frequency components. The filter is not very discriminative. It has low discriminability.

(7) If the filter has a large C. Like the filters in blue, it allows passage of a few frequency components. The filter is discriminative. It has high discriminability.

Slide 20: Lipschitz Frames

(1) The effect of the Lipschitz constant on the discriminability of a filter is clearer in the context of Lipschitz frames. Which are frames made up of Lipschitz filters, all of which have constant C. The figure illustrates the typical look of a Lipschitz frame. We have several filters of the same width located next to each other.

(2) If we increase the value of the Lipschitz constant, we make the filters of the frame thinner. But to maintain the frame condition that all frequencies have to pass through at least one filter, we also have to pack them tighter. This results in a larger number of filters in the frame. And therefore, in the possibility of discriminating more signals. In the ability to identify a larger number of frequency signatures.

(3) An observation that will be of interest when we study stability of filters and GNNs to deformations of the support, is that the discriminability of the frame is, or at least can be, the same at all frequencies. This is in contrast to what happens with the **integral** Lipschitz filters that we will define next.

Slide 21: Integral Lipschitz Filters

(1) To define an integral Lipschitz filter,

(2) Consider again a given filter with coefficients h_k and graph frequency response tilde h of lambda.

(3) The filter is said to be integral Lipschitz

(4) If there exists a positive constant C.

(5) Such that for **any** pair of arguments lambda 1 and lambda 2,

(6) The difference between the frequency response of the filter evaluated at frequencies lambda_1 and lambda_2 is bounded by the Lipschitz constant C multiplied by the difference between lambda 1 and lambda 2 divided by the average of lambda_1 and lambda_2. All of these differences are evaluated in absolute value.

(7) This definition is reminiscent of the Lipschitz filter definition except that in the right hand side we divide by the absolute value of the average lambda_1 plus lambda_2 divided by 2. This is tantamount to saying that an integral Lipschitz filter is Lipschitz in any interval. With a Lipschitz constant that is inversely proportional to the middle point of the interval.

(8) Letting lambda_2 approach lambda_1 the derivative of the filter's frequency response appears in the integral Lipschitz bound. We have that the product of lambda and the derivative of the filter's response is bounded by C. This is where the integral comes from in the name of the filter class. If this is true, the integral of the filter is a Lipschitz function. More importantly, it implies that the filter can't change for large lambda. Its derivative must vanish at least at a linear rate.

Slide 22: Discriminability of Integral Lipschitz Filters

(1) This fact, namely, that the derivative of the filter's frequency response vanishes for large lambda, indicates that integral Lipschitz filters are fundamentally different from Lipschitz filters.

(2) This is **not** the case at medium frequencies. For lambda neither too large nor too small, integral Lipschitz filters behave roughly as Lipschitz filters.

(3) But the behavior of the filter **is** very different at low frequencies. Since the maximum slope of the filter's response is inversely proportional to lambda, at low frequencies integral Lipschitz filters can be arbitrarily thin. They don't have to be. But they can. This is a good property to have. It implies the possibility of high discriminability.

(4) At large frequencies the behavior **is** very different as well. But in the opposite sense. Since the maximum slope of the filter's response is inversely proportional to lambda, at large frequencies integral Lipschitz filters must be flat. Their derivatives must vanish. No matter how large we make the constant C, the filter has no discriminability when the frequency argument lambda is large.

Slide 23: Integral Lipschitz Frames

(1) This manifests more clearly if we look at integral Lipschitz frames. Made up of integral Lipschitz filters with constant C. As in the case of integral Lipschitz filters.

(2) Increasing the constant C allows for thinner filters. Therefore for filters that are more tightly packed.

(3) And that consequently allows for the discrimination of more features.

(4) However, no matter C, integral Lipschitz frames can be very thin around lambda=0. This is true if C is large. But it is also true if C is small. As we illustrate with this frame. This translates to high discriminability.

(5) For large lambda, however, integral Lipschitz frames have to be wide, no matter how large we make C. Thus, integral Lipschitz frames cannot be discriminative at high frequencies. No matter how large we make C.

(6) This is bad news because, as we are going to see soon, filter stability to deformations of the shift operator requires filters that are integral Lipschitz. And it will therefore turn out that stability and discriminability are not only in conflict with each other, which we should expect them to be, but impossible to attain simultaneously. Which is an unexpected complication.

4 Stability of Graph Filters to Scaling

Slide 24: Stability of Graph Filters to Scaling

(1) In this section, we are going to analyze the stability of graph filters to scaling perturbations of the shift operator.

(2) The scaling of shift operators is a perturbation form that is not very realistic but it

nevertheless is useful to illustrate proof techniques and insights. Which will be used and discussed when analyzing more general perturbations

(3) Our focus is to prove the stability of graph filters to scaling. The bulk of the presentation is a proof. Which you are advised to follow with a pencil in your hand.

Slide 25: Stability of Graph Filters to Scaling of Shift Operators

(1) Shift operators describe the support of graph signals. Sometimes they are estimated from data. Sometimes they change over time. In either case, they are not god-given and immutable. This motivates the study of their deformations. In particular, the result of running the same graph convolutional filter on similar graphs.

(2) In this particular section we consider scalings of the edges of the graph. So that the shift operator S is deformed into the shift operator hat-S given by the product of S with the scaling factor 1 plus epsilon.

(3) This is a reasonable perturbation model in the sense that each edge undergoes the same relative scaling. Edges change proportional to their values.

(4) But the model is also unrealistic because all of the edges change by the same proportion.

(5) Unrealistic though it is, scaling opens the door to illuminating discussions. And we will see in forthcoming lectures that the stability proof for scaling contains the essential arguments that are needed in more generic proofs tied to more realistic perturbation models.

Slide 26: Stability of Graph Filters to Scaling of Shift Operators

(1) We are going to prove a theorem stating that integral Lipschitz graph filters are stable to scaling.

(2) In particular, given two graphs with shift operators S and S hat. The latter a scaling of S.

(3) Along with an integral Lipschitz filter with constant C.

(4) The operator norm difference between filters H of S and H of S hat is bounded by

(5) The product of the Lipschitz constant C and the scaling coefficient epsilon.

(6) This is a first order bound. There are higher order terms that we lump in a term of order epsilon squared.

(7) The theorem states that stability of graph filters to deformations of the graph is possible. No surprise here.

(8) But the important point to remark about the theorem, is that it requires the use of integral Lipschitz filters. This could be somewhat unexpected. Integral Lipschitz filters are not the first thing that comes to mind. More importantly, we know they have a limiting drawback: They can't discriminate high frequency features. Let us show a proof before we elaborate further on the conclusions of the theorem.

Slide 27: Proof Preliminaries

(1) The key arguments of the proof are in the graph Fourier transform domain. We state two facts about the GFT and the frequency response of the graph filter. We will use them in the proof.

(2) The first fact is about writing the signal x in terms of the components of its GFT.

(3) If we are given the GFT tilde x,

(4) We can recover x as the sum of tilde x_i times v_i.

(5) Where the v_i are the eigenvectors of the shift and the tilde x_i the entries of the GFT. The sum is over all eigenvector indexes.

(6) This fact is just an alternative way of writing the inverse GFT.

(7) Indeed, write the iGFT as V times tilde x.

(8) The columns of the matrix V are the eigenvectors v_i and the rows of tilde x are the entries of the GFT.

(9) Expand the product to obtain the sum of tilde x_i times v_i over eigenvector indexes.

(10) The second fact is about writing the derivative of the frequency response as a series.

(11) This derivative denotes as tilde h prime can be written as a the sum of k times h_k times lambda to the k-minus-1.

(12) As a consequence, the product of lambda with tilde h prime is the sum of k times h_k

times lambda to the k.

(13) To prove this fact just recall that the frequency response is a series.

(14)Where the derivative of each summand is k times h_k times lambda to the k-minus-1.

Slide 28: Proof Step 1: From Shift Perturbations to Filter Perturbations

(1) The first step in the proof of this theorem is to translate the perturbation of the shift operator into an expression for the perturbation of the filter operator.

(2) To do that, we consider the filter difference H-of-S-hat minus H-of-S.

(3) We write it explicitly in terms of the polynomials with coefficients h_k that represent the respective graph filters. We have a difference of two polynomials. One on S hat. One on S.

(4) We further write S hat as 1 plus epsilon times S. Which is the definition of the perturbed shift and regroup all of the terms in a single sum.

(5) We now expand the binomial term involving the perturbed shift. But we perform the expansion to first order only. Thus 1 plus epsilon to the power of k is replaced by 1 plus k epsilon as dictated by the first order approximation of a binomial.

(6) All other terms of the binomial expansion are grouped in the matrix O_k of epsilon

(7) Upon substitution of this expanded binomial into the expression we derived for the filter difference, the terms plus and minus S to the power of k cancel out. We are left with a simple expression where summands in the series are of the form h_k times k times epsilon times S to the power of k

(8) High order terms are grouped in the matrix O of epsilon. This matrix is of order 2 because the filters are analytic.

Slide 29: Proof Step 2: Evaluating and Reducing the Operator Norm

(1) We have thus reduced the filter difference to the sum of two terms. We repeat the result here for ease of reference. The second step of the proof involves evaluation of norms in this expression.

(2) To do so define the filter variation delta of S to represent the first summand. Which is the

one that includes terms that require further processing.

(3) Applying the triangle inequality to the equality in the first line, we bound the operator norm of the filter difference by the sum of the operator norm of the filter variation delta of S and the operator norm of the matrix O of epsilon. The one that contains the high order terms.

(4) We must now recall the definition of the operator norm of the filter variation delta of S as the maximum norm of the product delta of S times x over all vectors with unit norm.

(5) Consequently, the theorem follows if we manage to prove that the norm of the product between the filter variation delta of S and a vector x of unit norm is less than C times epsilon. With this fact holding for all vectors that have unit norm.

Slide 30: Proof Step 3: Shifting to the GFT Domain

(1) The next step in the proof is to shift the analysis to the GFT domain so that filter properties can be leveraged.

(2) Consider then the product of the filter variation delta of S

(3) With an arbitrary unit-norm input vector x. The norm of this vector is 1 because we are looking at the computation of an operator norm. Recalling this fact will be important later on.

(4) We now invoke Fact 1 in the proof preliminaries to write x as a sum of the eigenvectors v_i of the shift operator S multiplied by the respective entries of the GFT of x. This is where we move the analysis to the GFT domain.

(5) Upon substitution of x by its inverse-GFT expression, we reorder terms in the resulting sum to facilitate further manipulations.

(6) Since the v_i are eigenvectors of S, we have that S to the power of k times v_i can be simply written as lambda_i to the power of k times v_i. With lambda_i representing the eigenvalue associated with eigenvector v_i.

(7) Using this fact, the product delta S times x

(8) Which contains terms of the form S to the power of k times v_i

(9) Can be rewritten in a form that involves terms of the form lambda_i to the k times v_i. The shift operator S is replaced by an eigenvalue.

(10) The steps we have performed so far have led to a remarkable expression. Because the derivative of the filter's frequency response has appeared.

(11) Indeed, if we isolate the scalar terms that appear multiplying the eigenvector v_i in the innermost summation.

(12) We can invoke Fact 2 in the proof preliminaries to claim that this is the product of lambda_i with the derivative of the filter's response tilde h prime of lambda_i. The recognition that this factor appears is the most important step in the proof. All of the other pieces of the proof are either leading towards this recognition. Or building from this recognition.

Slide 31: Proof Step 4: Bounding the Filter's Response

(1) The proof is almost done. We just need a fourth and final step where we use the integral Lipschitz condition to bound the product of lambda_i with the derivative of the filter's response. To do so, rewrite the expression we have just derived, which is remarkably simple for such a complicated analysis, by the way

(2) This expression involves the quantity we bound with the integral Lipschitz condition. But to apply the bound we need to have an expression in which the product lamda-h-prime-lambda appears without a sign. We could do that by taking the absolute value. But a better bound is obtained if we compute energies.

(3) Proceed then to compute the energy of delta of S times x using the expression in the first line. And observe that the squares of the product lamda-h-prime-lambda appear.

(4) We use the integral Lipschitz bound on these squared terms

(5) Together with the fact that the input signal x has unit energy, which is preserved in the GFT domain. This leads to a bound of the form C-epsilon squared.

(6) We now take square root on both sides

(7) To complete the proof.

Slide 32: The Stability / Discriminability Non-Tradeoff

(1) The theorem says that Integral Lipschitz filters are necessary for stability to deformations of the supporting graph.

(2) The proof shows that this is not an artifact of the analysis. The result is tight. A series that represents the product of lambda_i with the derivative of the frequency response of the filter appears in the proof. This is the term that requires the use of integral Lipschitz filters.

(3) This is somewhat unexpected. When analyzing a learning parametrization it is natural to expect a stability vs discriminability tradeoff. That is, a less stable model is more discriminative. While a more stable model is less discriminative. But the appearance of integral Lipschitz filters means that, in a sense, we get a non-tradeoff.

(4) Graph filters can be stable only if they are integral Lipschitz. But integral Lipschitz filters have to be flat at high frequencies. Therefore, it is impossible for them to discriminate high frequency signals.

(5) It is impossible to use a graph filter to separate signals with high frequency features and be stable to deformations at the same time. There is no tradeoff. It is simply not possible.

5 Stability of Graph Neural Networks to Scaling

Slide 33: Stability of Graph Neural Networks to Scaling

(1) We have studied the stability of graph filters to scaling.

(2) Along the way, we saw that scaling is a reasonable perturbation model, albeit one that is not very realistic. It is however, a form of perturbation that offers valuable insights into stability properties and proof techniques.

(3) It is therefore worthwhile to study the stability of GNNs to scalings of the shift operator.

Slide 34: Normalizations

(1) To avoid appearance of meaningless constants, we introduce normalization assumptions on the filters and the nonlinearities that are used by the GNN.

(2) Our first normalization assumption is that at each layer of the GNN, the filters that compose the respective perceptrons have been normalized to unit operator norm.

(3) This is easy to achieve with scaling. If this norm is not one, we can scale the filter up or

down to satisfy this equality. Since graph filters are pointwise on the GFT domain, this condition is equivalent to having the maximum value of the frequency response equal to one. This makes the condition not only easy to satisfy, but easy to check. In any event, it is a normalization that we introduce for the sole purpose of simplifying bounds.

(4) Our second normalization assumption is that the nonlinearity sigma is Lipschitz with its Lipschitz constant normalized to one. That is, the absolute value of the difference of the nonlinearity evaluated at arguments x_1 and x_2 is bounded by the absolute value of the difference between the arguments themselves

(5) This is also easy to achieve with scaling and it holds for the standard ReLU, hyperbolic tangent, and absolute value nonlinearities.

(6) These two normalization put together imply that if the energy of the signal at the input of the GNN is not larger than one. The energy of layer outputs x_ell is also not larger than one. This holds at all layers and it is true because of the submultiplicative property of operator norms and because the Lipschitz constant of the nonlinearity is 1.

Slide 35: Stability of GNNs to Scaling

(1) We prove that GNNs whose layers are made up of integral Lipschitz filters are stable to scaling

(2) Formally, we consider two given graph shit operators S and S hat. With S hat corresponding to a scaling of S with factor 1 plus epsilon.

(3) We are also given a GNN operator Phi consisting of L single-feature layers. The operator is parametric on the choice of shift operator and filter tensor.

(4) We assume that the filters at each layer have unit operator norms and are integral Lipschitz with constant C.

(5) We further assume that the nonlinearity sigma is normalized Lipschitz.

(6) We then have that,

(7) The operator norm of the difference between the output of a GNN that runs on shift operator S, and the output of a GNN that runs on shift operator S hat.

(8) Is bounded by the product of the integral Lipschitz constant C, the number of Layers L and the scaling constant epsilon.

(9) This is a first order bound. There are higher order terms that we lump in a term which is of order epsilon squared.

(10) The theorem's claim is that GNNs inherit the stability of graph filters. The bound is essentially the same. And we will see in the proof that it is literally the same but propagated over the L layers. Notably, the statement also requires filters to be integral Lipschitz. Which we know engenders the challenge of discriminating high frequency components.

Slide 36: Proof Step 1: Characterizing the Effect of the Pointwise Nonlinearity

(1) The theorem is true because the nonlinearity is pointwise. It is unaware of the graph. There is nothing the nonlinearity does that is different on different graphs. The only difference between the GNN operators that run on different graphs is due to the difference on the graph filters they run at each layer. And we already know that these are stable. In any event, we are here for a formal proof. The first step of the proof is to characterize the effect of the pointwise nonlinearity.

(2) To to so, let x_ell be the output of the ell-th layer of the GNN defined on the shift operator S.

(3) Similarly, let x-hat_ell be the ell-th layer output of the GNN defined on the scaled shift operator S hat.

(4) With these definitions, we compare the output of layer ell of the GNN defined over S with the output of layer ell of the GNN defined over S hat.

(5) This difference can be rewritten in terms of the outputs of the previous layers, x_ell-minus-one and x-hat_ell-minus-one. It suffices for us to recall that the layer is a graph perceptron composing a pointwise nonlinearity sigma with a graph convolution.

(6) We now invoke our assumption that the nonlinearities in a graph neural network are normalized Lipschitz.

(7) That means that we can bound the difference on the output of the nonlinearity function in terms of its inputs. But the inputs to the nonlinearity function are the graph convolutions computed at layer ell. Thus, we can bound the difference between the outputs of the two layers elll by the difference of the graph convolutions themselves.

(8) This is the critical step of the proof. It eliminates the nonlinearity from the analysis and allows us to leverage our knowledge of the stability properties of graph filters that are

integral Lipschitz. In order to analyze the stability properties of graph neural networks. It is somewhat surprising that the nonlinearity can be so easily handled. But we are not going to complain that a proof is easy. In any event, the rest of the proof is made up of simple algebraic manipulations.

Slide 37: Proof Step 2: Implementing Norm Manipulations

(1) Start with the bound we obtained in the previous slide. Which we repeat here for reference.

(2) In the right hand side, we add and subtract

(3) The result of applying the graph convolution defined by S hat.

(4) On the signal that is produced by the GNN that runs on graph S at the output of layer ell-minus-one. The cross product across the different GNNs is important here.

(5) As the distance modulo permutation is a proper norm, we can proceed to use the triangle inequality to separate the norm into a term depending on the signal x_ell-minus-one, and another term depending on the difference between x_ell-minus and x-hat_ell-minus-one. The norm is also submultiplicative. Allowing us to further break down each of these two terms into the product between the norm of a filter and the norm of a signal.

(6) We now recall that we have assumed the filters to be normalized.

(7) From where we can say that the norm of the filter H_ell of S hat equals to one.

(8) And since the nonlinearities are also normalized, we can further say that the norm of the graph signal x_ell-minus-one is also bounded by 1.

(9) This leaves us with a term depending on the difference between the filters of layer ell.

(10) And a term depending on the difference between the signals received from the previous layer. The signals x and x hat at layer ell minus 1.

(11) Since we have already proven that integral Lipschitz graph filters are stable to scaling, the filter difference can be bounded. Indeed, we have proven that the operator norm of the difference between the filters, is bounded by the product of the scaling constant epsilon times the integral Lipschitz constant C. Plus some second order terms.

(12) Putting all bounds together we end up bounding the difference between signals at layer ell with the product epsilon-C and the difference between the signals at layer ell-minus-

one.

(13) This gives us a recursion that we can apply backwards from layer L up to the Layer 1.

(14) At each layer, the bound we just computed has the same form. We therefore add a distortion bounded by epsilon-C at each layer. From which we get the L factor in the stability bound of the GNN.

(15) This completes the proof.

Slide 38: The Stability / Discriminability Tradeoff of GNNs

(1) In the proof of the stability of GNNs to scaling, we see that GNNs inherit the same stability properties of the graph filters in their layers. The perturbation comes from the filters. The nonlinearity does not add to the distortion of the output. In particular, GNNs must have layers with integral Lipschitz filters if they are to be stable.

(2) We have seen already that integral Lipschitz filters must be flat at high frequency. This is a serious limitation. It precludes the discrimination of high frequency features.

(3) It is impossible for integral Lipschitz filters to separate signals with high frequency features and be stable to deformations simultaenously.

(4) On the flip side, integral Lipschitz filters can be very sharp at low frequencies. They have high discriminability when the frequency argument is close to zero.

(5) This means that for features that are located at low frequencies, filters can be very discriminative. And the same time, they can be very stable to deformations as well. We do not need a large constant C for an integral Lipschitz filter to be discriminative around frequency zero.

(6) The low discriminability at low frequencies and the high discriminability at large frequencies are properties of filters, that **layers** of the GNN inherit. But while the **GNN as a whole** inherits the stability of the filters, it doesn't have to inherit their discriminability limitations. In fact, avoiding this fate is the role of the nonlinearity. The nonlinearity is a low pass operation that demodulates high frequencies components into low frequencies.

(7) Where they can be discriminated sharply with a stable filter at the **next** layer.

(8) Thus, GNNs can be both, stable and discriminative. They can be stable if they use integral Lipschitz filters. And they can be discriminative by demodulating high frequency

components into low frequency components. They do that with low pass pointwise nonlinearities. In order to enable their stable discrimination in deeper layers. This is a tradeoff that linear filters cannot achieve. They are either discriminative or stable. But they **can't** be both. That GNNs **can** be both, stable and discriminative, explains their better performance relative to linear graph filters.