Algebraic Neural Networks: Stability to Deformations

Alejandro Ribeiro and Alejandro Parada Mayorga

Dept. of Electrical and Systems Engineering

University of Pennsylvania

Email: aribeiro@seas.upenn.edu

Web: alelab.seas.upenn.edu

March 24, 2021

Introduction

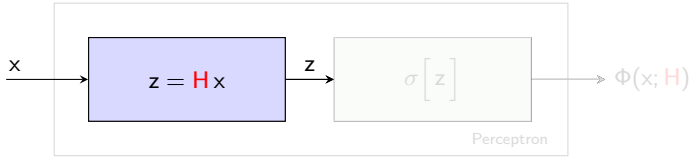
- We are given some input signal (data) and want to extract some information of interest

- These images are are made up of individual pixel color and luminance that represent the Eiffel Tower

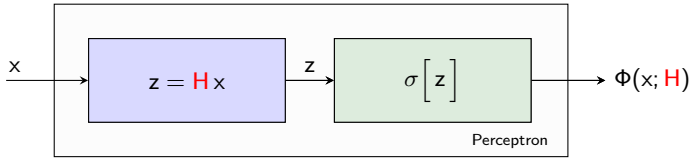
- Each drone senses its own velocity and the distances to neighboring drones

- They want to determine accelerations that would lead them to coordinate their velocities (globally)

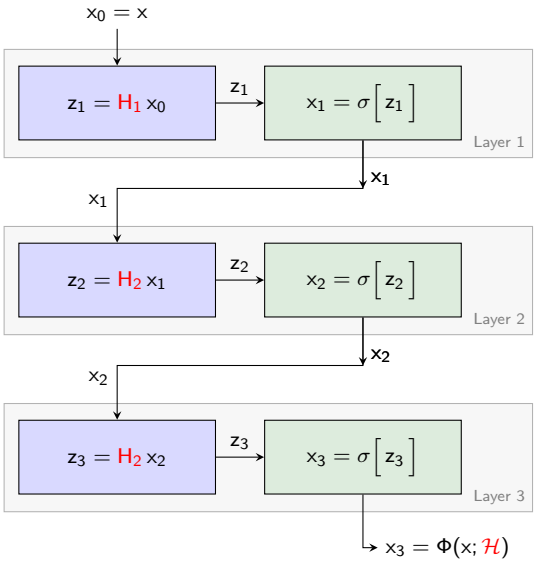
- Given input/output pairs in a training set $\Rightarrow$ Find a function approximation that generalizes well
- Learning parametrization $\Rightarrow$ function class that restricts the space of allowable maps $\Phi(x)$



- Given input/output pairs in a training set $\Rightarrow$ Find a function approximation that generalizes well
- Learning parametrization $\Rightarrow$ function class that restricts the space of allowable maps $\Phi(x)$



- Stack a few perceptrons to build a fully connected neural network
- Nonlinearities are pointwise.
  $\Rightarrow$ They do not mix components
- Neural networks are not that different from linear transformations

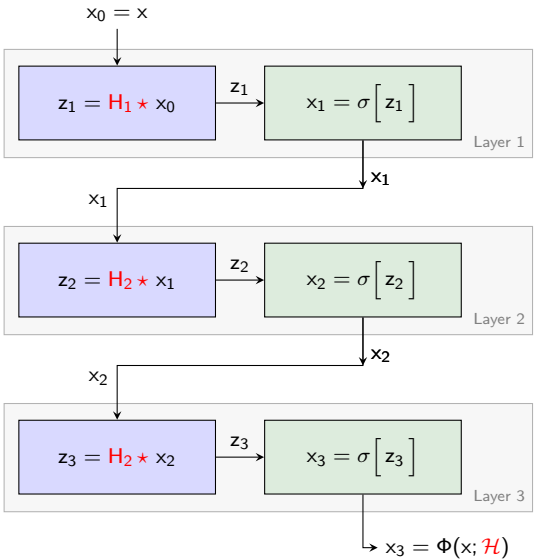


Fully connected neural networks do not scale to high dimensional inputs

Convolutional neural networks are used for scalable processing / learning

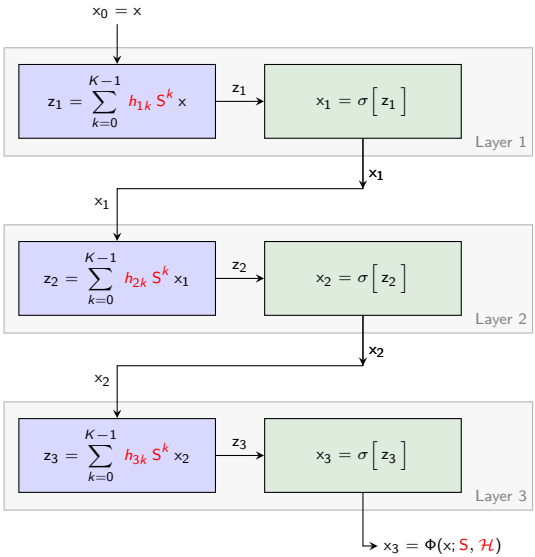
To process images...

- Require the linear transformations of all layers to be Euclidean convolutional filters



To process graphs and signals on graphs...

- Require the linear transformations of all layers to be Graph convolutional filters



Ruiz-Gama-Ribeiro, *Graph Neural Networks: Architectures, Stability and Transferability,* PIEEE 2021, http://arxiv.org/abs/2008.01767

- Euclidean and graph convolutional filters leverage signal (data) symmetries

  $\Rightarrow$ Translation equivariance for Euclidean filters and permutation equivariance for graph filters

- CNNs and GNNs leverage quasi quasi symmetries better than Euclidean and graph filters

  $\Rightarrow$ CNNs are more stable to diffeomorphisms (warping) of Euclidean space

  $\Rightarrow$ CNNs are more stable to multiplicative perturbations of the graph

Stéphane Mallat, *Group Invariant Scattering*, CPAM 2012, http://arxiv.org/abs/1101.2286

Gama-Bruna-Ribeiro, *Stability Properties of Graph Neural Networks*, TSP 2020, http://arxiv.org/abs/1905.04497

CNNs and GNNs are particular cases of an abstract algebraic architecture $\Rightarrow$ AlgNN

- Which includes other known types of convolutional architectures $\Rightarrow$ Group NNs, Graphon NNs
- And a large number of unexplored architectures

Parada Mayorga-Ribeiro, *Algebraic Neural Networks: Stability to Deformations*, TSP 2020, http://arxiv.org/abs/2009.01433

The shared stability properties of CNNs and GNNs relative to deformantions are a consequence of their shared algebraic structure.

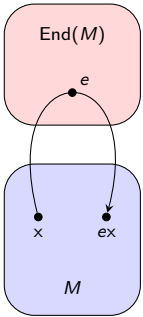
- Which are also shared by other known types of convolutional architectures $\Rightarrow$ Group, Graphon NNs
- And are also shared by a large number of unexplored architectures

Parada-Mayorga-Ribeiro, *Algebraic Neural Networks: Stability to Deformations*, TSP 2020, http://arxiv.org/abs/2009.01433

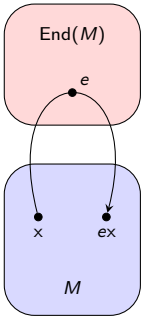
Algebraic (Convolutional) Signal Processing

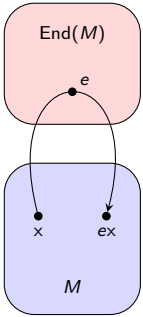
- Consider a vector space M. Signals (data) are elements $x \in M$
- Associated to M we have the space of endomorphisms $\text{End}(M)$

  $\Rightarrow$ The set of all linear maps e from M to M



- Signals $M = \mathbb{R}^n \Rightarrow$ Square matrix multiplications $\Rightarrow \mathrm{y} = \mathrm{E}\mathrm{x}$
- Signals in $M = L_2([0,1]) \Rightarrow$ Linear functionals $\Rightarrow \mathrm{y}(u) = \int_0^1 E(u,v)\mathrm{x}(v)\,dv$





- Signals $M = \mathbb{R}^n$ $\Rightarrow$ Square matrix multiplications $\Rightarrow$ $\mathrm{y} = \mathrm{E}\mathrm{x}$
- Signals in $M = L_2([0,1])$ $\Rightarrow$ Linear functionals $\Rightarrow$ $\mathrm{y}(u) = \int_0^1 E(u,v)\mathrm{x}(v)\,dv$
- End(M) is the set of all linear maps that can be applied to a signal x that lives in M

  $\Rightarrow$ Learning in End(M) is not scalable $\Rightarrow$ Search over All Matrices. Or over all linear functionals

- Scalable learning $\Rightarrow$ Restrict allowable linear maps

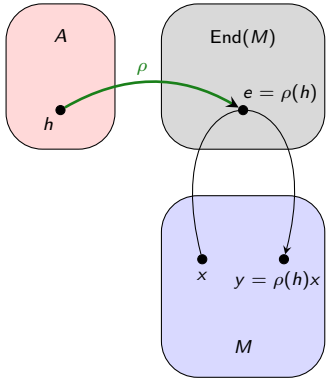
  $\Rightarrow$ To those that represent another (more restrictive) algebra

- The representation is defined by a homomorphism

$$\rho : A \to \text{End}(M)$$

- Map abstract filters $a \in A$ into concrete endomorphisms $\rho(a)$

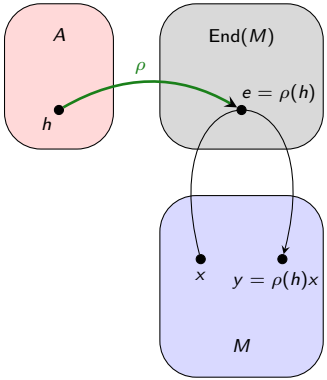
  $\Rightarrow$ Convolutional filters yield outputs $\Rightarrow y = \rho(a)x$



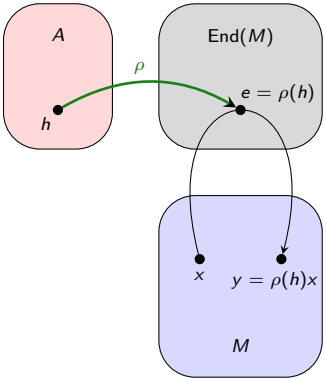
An Algebraic SP model is a triplet (A, M, ρ)

- A is an Algebra with unity where filters $h \in A$ live

  $\Rightarrow$ It defines the rules of convolutional signal processing



An Algebraic SP model is a triplet (A, M, ρ)



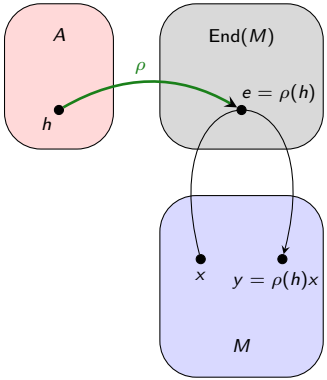
- M is a vector space

  $\Rightarrow$ The space containing the signals x we want to process

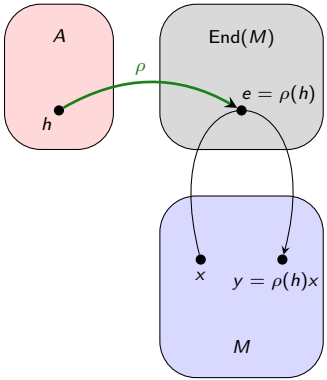
An Algebraic SP model is a triplet (A, M, ρ)

- ρ is a homomorphism from A to the endomorphisms of M

  $\Rightarrow$ Instantiates the abstract filter h in the space End(M)



An Algebraic SP model is a triplet (A, M, ρ)

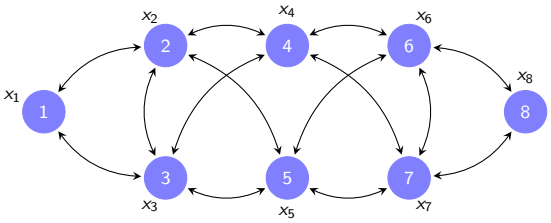


- ρ is a homomorphism from A to the endomorphisms of M

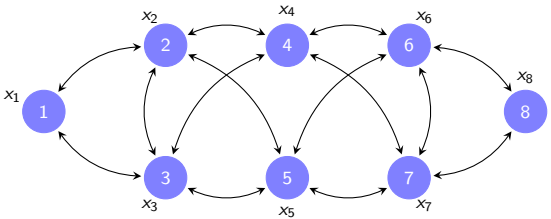
  $\Rightarrow$ Instantiates the abstract filter h in the space End(M)

- Any $h \in A$ is a filter which operates on signals according to the homomorphism $\Rightarrow y = \rho(h)x$

- If this sounds complicated it is because of the level of abstraction. But it is in fact very easy
- In GSP signals $x \in \mathbb{R}^n$ are supported on the nodes of a graph with matrix representation S
- Convolutional filters are polynomials on the matrix representation $\Rightarrow y = \sum_{k=0}^{K-1} h_k S^k x$

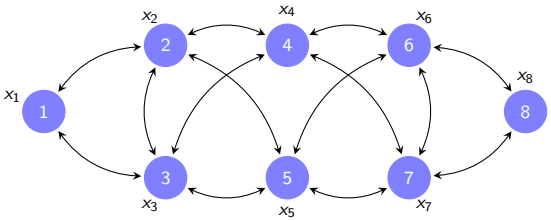


- If this sounds complicated it is because of the level of abstraction. But it is in fact very easy
- In GSP signals $x \in \mathbb{R}^n$ are supported on the nodes of a graph with matrix representation S
- Convolutional filters are polynomials on the matrix representation $\Rightarrow y = \sum_{k=0}^{K-1} h_k S^k x$



Signals x live in the vector space $M = \mathbb{R}^n$

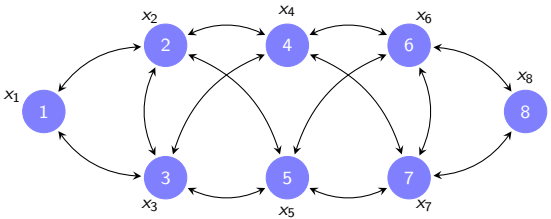
- If this sounds complicated it is because of the level of abstraction. But it is in fact very easy
- In GSP signals $\mathbf{x} \in \mathbb{R}^n$ are supported on the nodes of a graph with matrix representation $\mathbf{S}$
- Convolutional filters are polynomials on the matrix representation $\Rightarrow \mathbf{y} = \sum_{k=0}^{K-1} h_k \mathbf{S}^k \mathbf{x}$



Filters are elements of the algebra

of polynomials $\Rightarrow a = \sum_{k=0}^{K-1} h_k t^k$

- If this sounds complicated it is because of the level of abstraction. But it is in fact very easy
- In GSP signals $x \in \mathbb{R}^n$ are supported on the nodes of a graph with matrix representation S
- Convolutional filters are polynomials on the matrix representation $\Rightarrow y = \sum_{k=0}^{K-1} h_k S^k x$



The homomorphism maps filter h

to the matrix $\Rightarrow \rho(a) = \sum_{k=0}^{K-1} h_k S^k$

- The purpose of the abstraction is, of course, to make it very general

Graph Signal Processing on a Different Graph

- Change vector space to appropriate graph dimension
- Keep the algebra of polynomials
- Replace the homomorphism to the shift operator of the new graph

- The purpose of the abstraction is, of course, to make it very general

Discrete Time Signal Processing

- Replace the vector space by the space of square summable sequences
- Keep the algebra of polynomials
- Replace the homomorphism to compositions of the translation (time shift) operator

- The purpose of the abstraction is, of course, to make it very general

Image Convolutions

- Replace the vector space by the space of matrices
- Replace the algebra with the algebra of polynomials of two variables
- Replace the homomorphism to compositions of horizontal and vertical translation operators

- The purpose of the abstraction is, of course, to make it very general

Group Convolutions

- Vector space made up of functions supported on the group
- The algebra is the algebra of the group
- The homomorphism is the identity

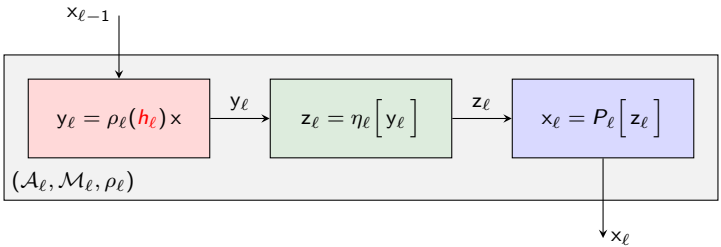
- The purpose of the abstraction is, of course, to make it very general

Whatever Convolutions

- Square summable functions. Functions on manifolds. Finite fields
- Lie Algebras
- Homomorphism chosen to match algebra to vector space

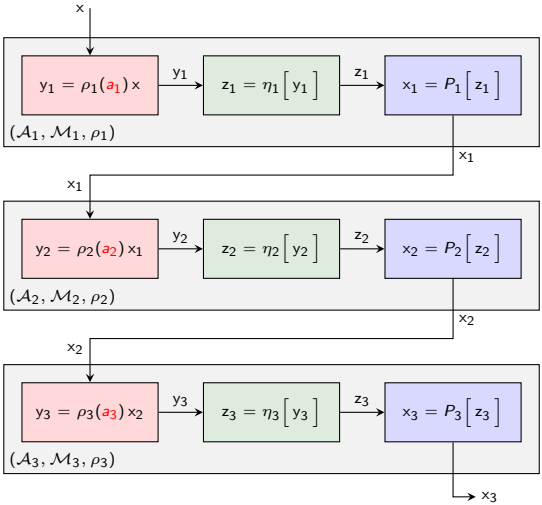
Algebraic Neural Networks

- Layers defined by an ASP model $\Rightarrow (\mathcal{A}_\ell, \mathcal{M}_\ell, \rho_\ell)$ that specifies the type of convolutional processing
- Add pointwise nonlinearity η_ℓ and pooling operator P_ℓ to match vector spaces $\mathcal{M}_\ell$ and $\mathcal{M}_{\ell+1}$



- Trainable parameters are the filters h_ℓ $\Rightarrow$ Numerically, we train directly on $\rho_\ell(h_\ell)$

- Stack several layers to build and AlgNN



Deformations

- The algebraic models $(\mathcal{A}_\ell, \mathcal{M}_\ell, \rho_\ell)$ determine the equivariance properties of the Algebraic NN

  $\Rightarrow$ Equivariance to translations in CNNs

  $\Rightarrow$ Equivariance to permutations in GNNs and Graphon NNs

  $\Rightarrow$ Equivariance to actions of the group in Group NNs

- These are properties of the filters that the Algebraic NN inherits

  $\Rightarrow$ Algebraic NNs outperform Algebraic filters. Why? $\Rightarrow$ Stability to Deformations

- To define model deformations we need the notion of generator of an algebra

Generators: The set $\mathcal{G} \subseteq \mathcal{A}$ generates $\mathcal{A}$ if all $a \in \mathcal{A}$ are polynomial functions of elements of $\mathcal{G}$

Shift Operators: The set $\mathcal{S}$ of homomorphism images $\mathbf{S} = \rho(g)$ is the set of shift operators

- Definitions of generators and shift operators allows writing filters as polynomials on shift operators

$$\rho(a) = p_{\mathcal{M}}\big(\rho(\mathcal{G})\big) = p_{\mathcal{M}}\big(\mathcal{S}\big) = p\big(\mathcal{S}\big)$$

- We define perturbations of Algebraic models as perturbations of shift operators $\Rightarrow \tilde{\mathbf{S}} = \mathbf{S} + \mathbf{T}(\mathbf{S})$
- The ASP model $(\mathcal{A}, \mathcal{M}, \rho)$ is consequently perturbed to the ASP model $(\mathcal{A}, \mathcal{M}, \tilde{\rho})$ such that

$$\tilde{\rho}(a) = p_{\mathcal{M}}\big(\tilde{\rho}(g)\big) = p_{\mathcal{M}}\big(\tilde{\mathcal{S}}\big)$$

  That is, the polynomials that define filters are the same. But they use the perturbed shift operator

- Graphs $\Rightarrow$ Shift operator $\mathbf{S}$ represents a graph $\Rightarrow$ $\tilde{\mathbf{S}}$ represents different graph
- Time $\Rightarrow$ $\mathbf{S}$ represents translation equivariance $\Rightarrow$ $\tilde{\mathbf{S}}$ represents quasi-translation equivariance

- We analyze a first order perturbation model of the form $\Rightarrow \mathbf{T}(\mathbf{S}) = \mathbf{T}_0 + \mathbf{T}_1\mathbf{S}$

- The operators $\mathbf{T}_0$ and $\mathbf{T}_1$ are compact normal with norms satisfying $\|\mathbf{T}_0\| \leq 1$ and $\|\mathbf{T}_1\| \leq 1$

- $\mathbf{T}_r$ and $\mathbf{S}$ do not commute. Write $\mathbf{S}\mathbf{T}_r = \mathbf{T}_r\mathbf{S} + \mathbf{S}\mathbf{P}_r$. and define commutation factor $\delta = \max_r \dfrac{\|\mathbf{P}_r\|}{\|\mathbf{T}_r\|}$

Stability to Deformations

Stable Operators: We say operator $p(\mathbf{S})$ is stable if there exist constants C_0, $C_1 > 0$ such that

$$\left\| p(\mathbf{S})\mathbf{x} - p(\tilde{\mathbf{S}})\mathbf{x} \right\| \leq \left[C_0 \sup_{\mathbf{S}\in\mathcal{S}} \|\mathbf{T}(\mathbf{S})\| + C_1 \sup_{\mathbf{S}\in\mathcal{S}} \left\| D_{\mathbf{T}}(\mathbf{S}) \right\| + \mathcal{O}\left(\|\mathbf{T}(\mathbf{S})\|^2 \right) \right] \|\mathbf{x}\|$$

for all $\mathbf{x} \in \mathcal{M}$ and $D_{\mathbf{T}}(\mathbf{S})$ denoting the Fréchet derivative of $\mathbf{T}$.

- $\left\| p(\mathbf{S})\mathbf{x} - p(\tilde{\mathbf{S}})\mathbf{x} \right\|$ is bounded by the size of the deformation. Measured by value and rate of change
- Staility is not a given $\Rightarrow$ Counter examples in GNN and processing of time signals.

- Filters are polynomials on shift operators $\Rightarrow$ Isomorphic to polynomials with complex variables

Lipschitz Filter: Polynomial $p : \mathbb{C} \to \mathbb{C}$ is Lipschitz if $\|p(\lambda) - p(\mu)\| \leq L_0 \|\lambda - \mu\|$ for some L_0

Integral Lipschitz: Polynomial $p : \mathbb{C} \to \mathbb{C}$ is Integral Lipschitz if $\left\| \lambda \frac{dp(\lambda)}{d\lambda} \right\| \leq L_1$ for some L_1

- Restricted attention to algebras with a single generator. Generalizations are cumbersome but ready

Stability of Algebraic Filters

A filter that is Lipschitz and Integral Lipschitz is stable

$$\left\| p(\mathbf{S})\mathbf{x} - p(\tilde{\mathbf{S}})\mathbf{x} \right\| \leq \left[(1+\delta) \left(L_0 \sup_{\mathbf{S}} \|\mathbf{T}(\mathbf{S})\| + L_1 \sup_{\mathbf{S}} \|D_{\mathbf{T}}(\mathbf{S})\| \right) + \mathcal{O}(\|\mathbf{T}(\mathbf{S})\|^2) \right] \|\mathbf{x}\|$$

- Good news $\Rightarrow$ Algebraic filters can be made stable to perturbations
- Alas, We either have stability or discriminability. Integral Lipschitz Filter $\Rightarrow$ $\left\| \lambda \frac{dp(\lambda)}{d\lambda} \right\| \leq L_1$
- Commutativity factor affects stability constant but does not generate instability

Stability of Algebraic Filters

Let $\Phi_\ell(\mathbf{S},\mathbf{x})$ and $\Phi_\ell(\tilde{\mathbf{S}},\mathbf{x})$ be the operators associated with layer ℓ of an Algebraic NN. If the layer filters are Lipschitz and Integral Lipschitz,

$$\left\|\Phi_\ell(\mathbf{S},\mathbf{x}) - \Phi_\ell(\tilde{\mathbf{S}},\mathbf{x})\right\| \leq \left[(1+\delta)\left(L_0 \sup_{\mathbf{S}} \|\mathbf{T}(\mathbf{S})\| + L_1 \sup_{\mathbf{S}} \|D_{\mathbf{T}}(\mathbf{S})\|\right) + \mathcal{O}(\|\mathbf{T}(\mathbf{S})\|^2)\right] \|\mathbf{x}\|$$

- Good news $\Rightarrow$ Algebraic NNs can be made stable to perturbations. It's the same bound
- Individual layers loose discriminability. Integral Lipschitz Filter $\Rightarrow$ $\left\|\lambda \frac{dp(\lambda)}{d\lambda}\right\| \leq L_1$
- Nonlinearity mixes frequency components $\Rightarrow$ Recover discriminability in subsequent layers

Frequency Representations

Definition (Frequency Representation)

In an algebra A with generators $g_i \in \mathcal{G}$ we are given the filter h expressed as the polynomial

$$h = \sum_{k_1,\dots k_r} h_{k_1,\dots,k_r} {g_1}^{k_1} \dots {g_r}^{k_r} = p_A(\mathcal{G})$$

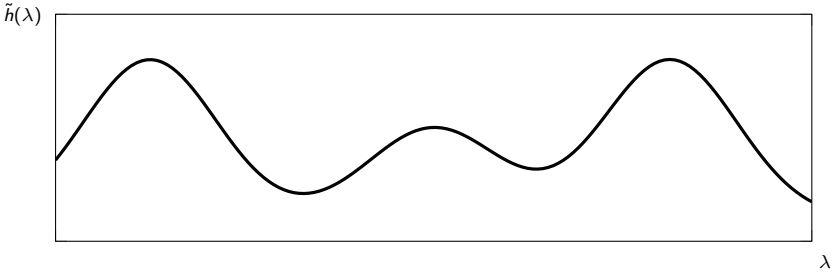
The frequency representation of h over the field F[1] is the polynomial function with variables $\lambda_i \in \mathcal{L}$

$$\tilde{h}(\mathcal{L}) = \sum_{k_1,\dots k_r} h_{k_1,\dots,k_r} {\lambda_1}^{k_1} \dots {\lambda_r}^{k_r}$$

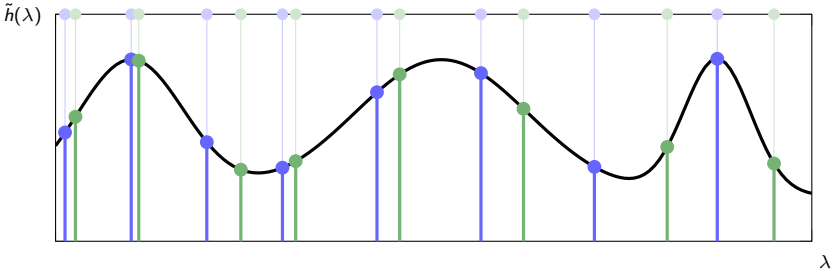
[1] The field is unspecified in the definition. But unless otherwise noted F refers to the field over which the vector space M is defined

- Frequency representation $\equiv$ polynomial over the field. $\Rightarrow$ E.g., a function over the reals

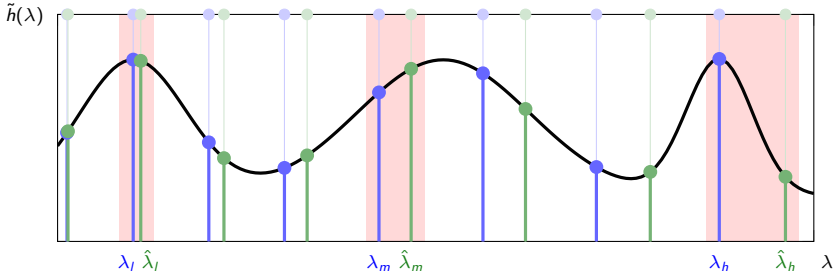
- It is decoupled the homomorphism. From the shift operator that specifies actions on x



- Shift $\mathbf{S}$ has eigenvalues λ_i $\Rightarrow$ The response is instantiated at these eigenvalues $\tilde{h}(\lambda_i) = \sum_{k=1}^{\infty} h_k \lambda_i^k$
- Shift $\hat{\mathbf{S}}$ has eigenvalues $\hat{\lambda}_i$ $\Rightarrow$ The response is instantiated at these eigenvalues $\tilde{h}(\hat{\lambda}_i) = \sum_{k=1}^{\infty} h_k \hat{\lambda}_i^k$



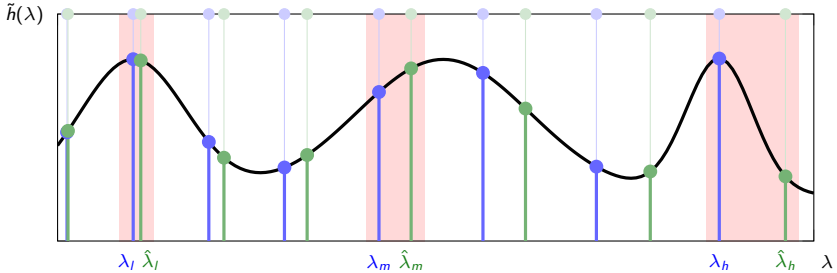
- Conceptually, we can learn all there is to be learnt from shift operator dilations $\Rightarrow \hat{\mathbf{S}} = \mathbf{S} + \epsilon \mathbf{S}$
- Eigenvalues dilate $\lambda_i \to \hat{\lambda}_i = (1+\epsilon)\lambda_i$. Frequency response instantiated on dilated eigenvalues



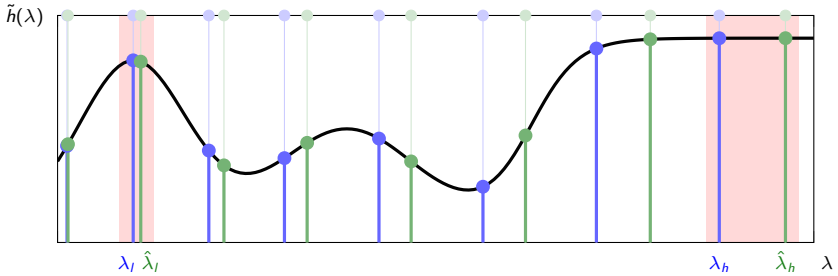
- Large eigenvalues move more. Signals with high frequencies are more difficult to process

  $\Rightarrow$ Even small perturbations yield large differences in the filter values that are instantiated

  $\Rightarrow$ We think we instantiate $h\Big(\lambda_i\Big)$ $\Rightarrow$ But in reality we instantiate $h\Big(\hat{\lambda}_i\Big) = h\Big((1+\epsilon)\,\lambda_i\Big)$

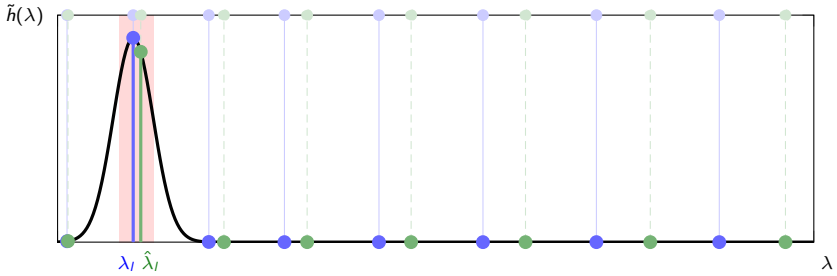


- To attain stable algebraic signal processing we need integral Lipschitz filters $\Rightarrow \left|\lambda \tilde{h}'(\lambda)\right| \leq C$
- Either the eigenvalue does not change because we are considering low frequencies
- Or the frequency response does not change when we are considering high frequencies



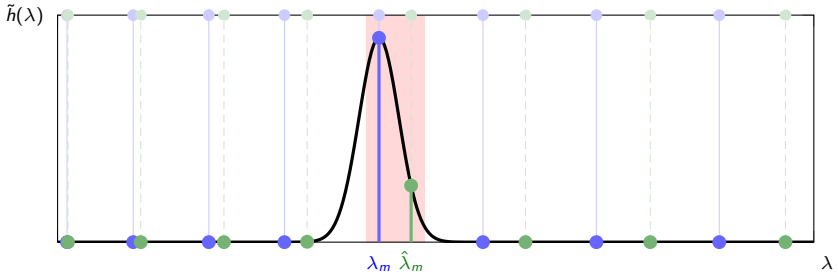
- At low frequencies a sharp highly discriminative filter is also highly stable

$\Rightarrow$ Ideal response $h\Big(\lambda_l\Big)$ is very close to perturbed response $h\Big(\hat{\lambda}_l\Big) = h\Big((1+\epsilon)\,\lambda_l\Big)$



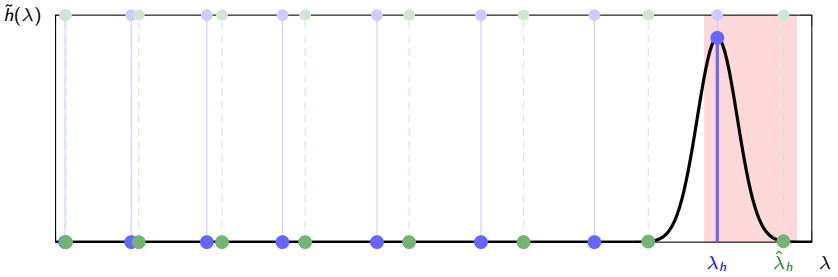
- At intermediate frequencies a sharp highly discriminative filter is somewhat stable

$\Rightarrow$ Ideal response $h\Big(\lambda_m\Big)$ is somewhat close to perturbed response $h\Big(\hat{\lambda}_m\Big) = h\Big((1+\epsilon)\,\lambda_m\Big)$



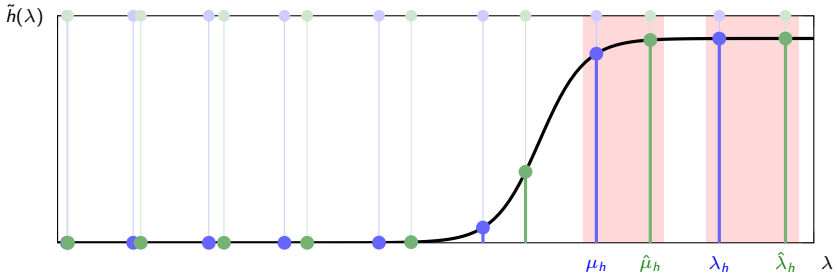
- At high frequencies a sharp highly discriminative filter is unstable. It becomes useless

$\Rightarrow$ Ideal response $h\Big(\lambda_h\Big)$ is very different from perturbed response $h\Big(\hat{\lambda}_h\Big) = h\Big((1+\epsilon)\,\lambda_h\Big)$



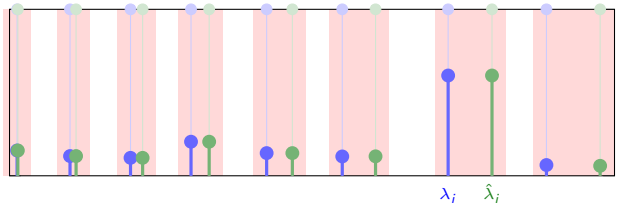
- We can have stability to deformations if we use an integral Lipschitz filters $\Rightarrow \left| \lambda \tilde{h}'(\lambda) \right| \leq C$

  $\Rightarrow$ But this precludes the discrimination of high frequency components

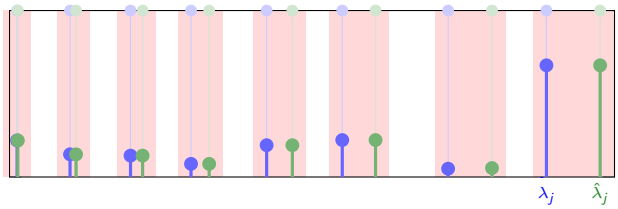


- Nonlinearities $\sigma(\mathbf{v}_i)$ and $\sigma(\mathbf{v}_j)$ spread energy across all frequencies
- Some energy where it used to be
- Some energy at other high frequencies
- Some energy at medium frequencies
- Some energy at low frequencies
- Where it can be discriminated with a stable filter in Layer 2

Spectrum of nonlinearity applied to $\mathbf{v}_i \Rightarrow \mathbf{V}^H\sigma(\mathbf{v}_i)$



Spectrum of nonlinearity applied to $\mathbf{v}_j \Rightarrow \mathbf{V}^H\sigma(\mathbf{v}_j)$



Stability Properties of Algebraic Neural Networks

AlgNNs can be simultaneously discriminative and stable to deformations. Algebraic filters cannot.

Stability Properties of Algebraic Neural Networks

For the same sensitivity to deformations, AlgNNs are more discriminative than algebraic filters

Closing Comments

- Provide a common language for Convolutional-, Graph-, Graphon-, Group-, Unknown- NNs
- Uncover a shared stability vs discriminability tradeoff that stems from a shared algebraic structure
  - $\Rightarrow$ Algebraic linear filters can be either stable **OR** discriminative. But not both
  - $\Rightarrow$ Algebraic neural networks can be both, stable **AND** discriminative
- Theory wishlist: Nonconmmutative algebras. Banach Algebras
- Application wishlist: Lie Algebras. Manifolds. Markov Random Fields

- arxiv.org/abs/2009.01433

  $\Rightarrow$ *Algebraic Neural Networks: Stability to Deformations* A. Parada Mayorga and A. Ribeiro

- gnn.seas.upenn.edu

  $\Rightarrow$ Video and scripts for our course on graph neural networks at Penn. Lecture 12

- aribeiro@seas.upenn.edu and alejopm@seas.upenn.edu

  $\Rightarrow$ Send us an email if you have questions or ideas