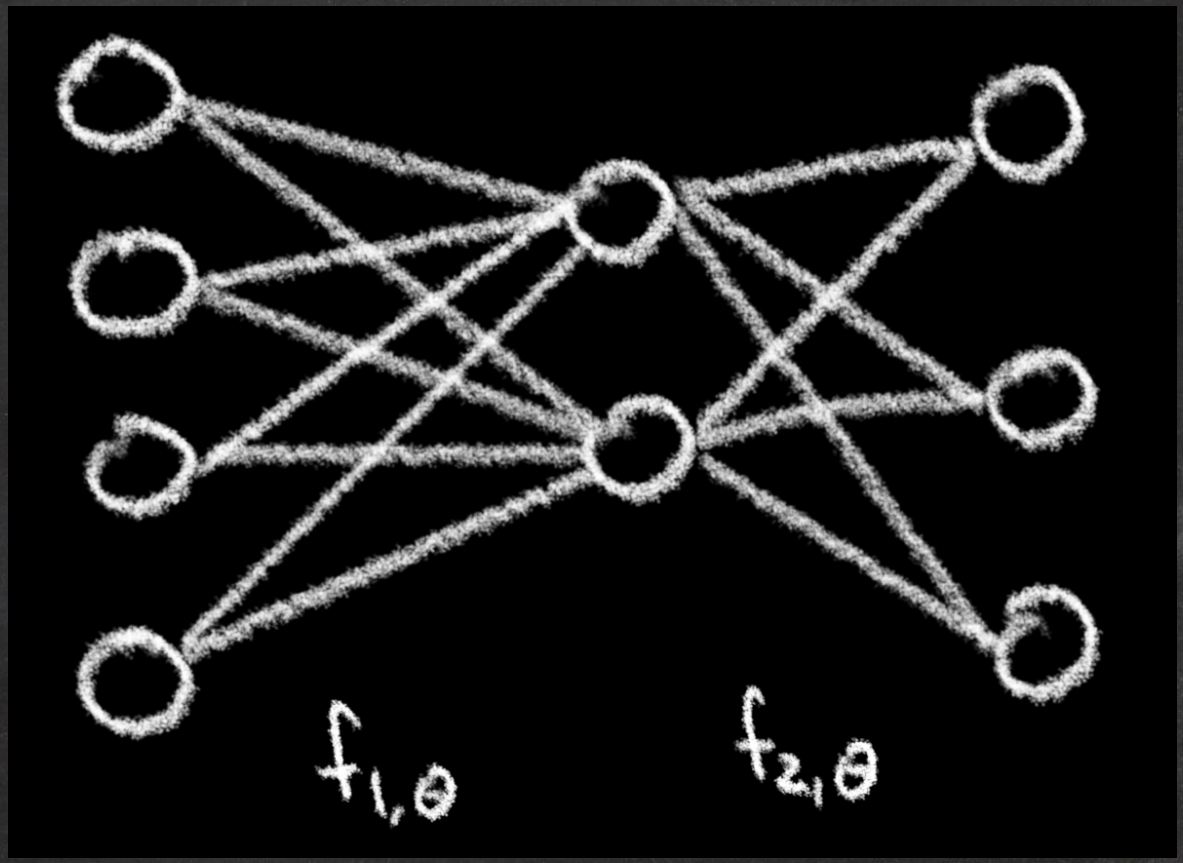
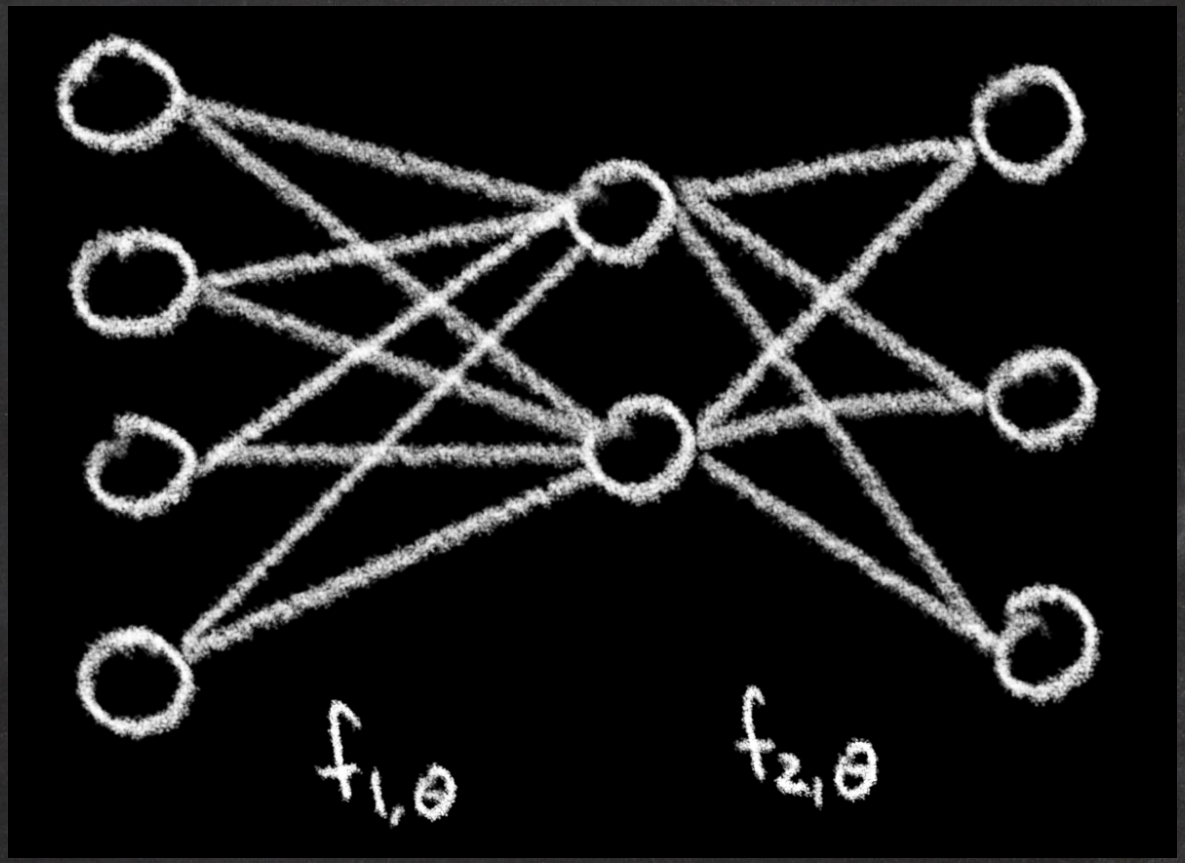
Convolutional Tensors

Kathlén Kohn

feedforward neural networks



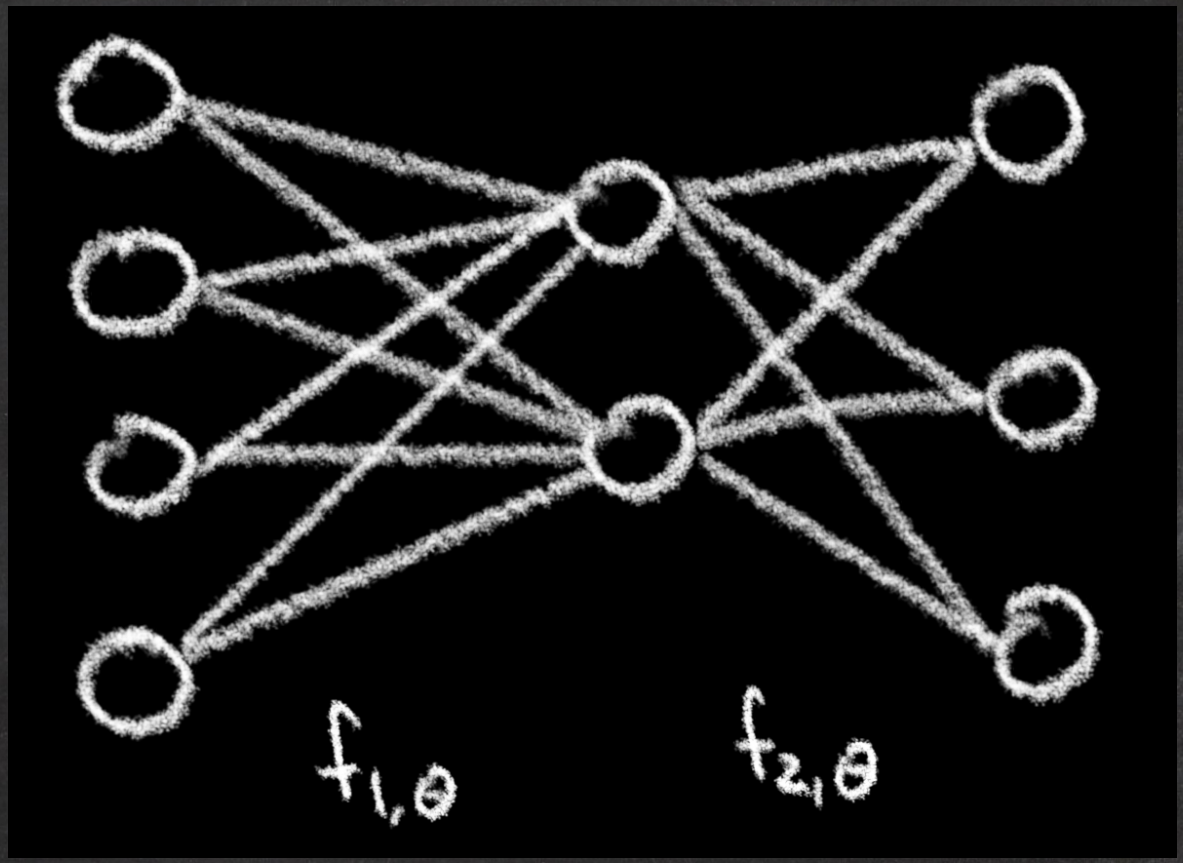
feedforward neural networks



are parametrized families of functions

$$\mu : \mathbb{R}^N \longrightarrow \mathcal{M},$$
$$\theta \longmapsto f_{L,\theta} \circ \ldots \circ f_{1,\theta}$$

feedforward neural networks



are parametrized families of functions

$$\mu : \mathbb{R}^N \longrightarrow \mathcal{M},$$
$$\theta \longmapsto f_{L,\theta} \circ \ldots \circ f_{1,\theta}$$

$\mathcal{M} =$ function space / neuromanifold, $L = \#$ layers

training a network

Given training data $\mathcal{D}$, the goal is to minimize the loss

$$\mathbb{R}^N \xrightarrow{\mu} \mathcal{M} \xrightarrow{\ell_{\mathcal{D}}} \mathbb{R}.$$

training a network

Given training data $\mathcal{D}$, the goal is to minimize the loss

$$\mathbb{R}^N \xrightarrow{\mu} \mathcal{M} \xrightarrow{\ell_{\mathcal{D}}} \mathbb{R}.$$

Geometric questions:

- How does the network architecture affect the geometry of the function space?
- How does the geometry of the function space impact the training of the network?

training a network

Given training data $\mathcal{D}$, the goal is to minimize the loss

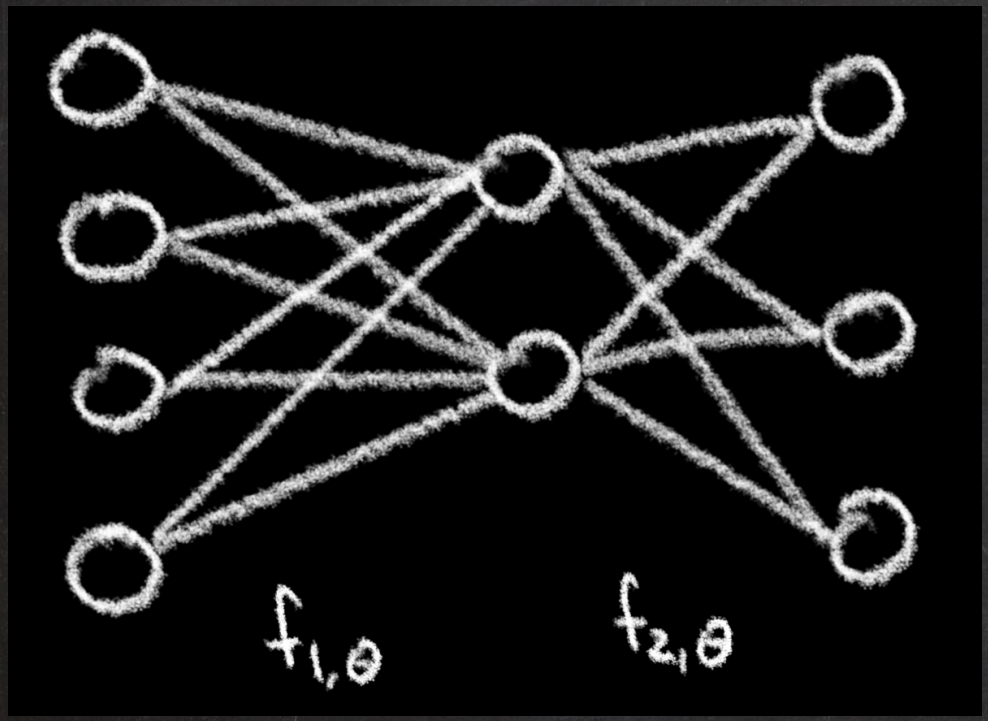
$$\mathbb{R}^N \xrightarrow{\mu} \mathcal{M} \xrightarrow{\ell_{\mathcal{D}}} \mathbb{R}.$$

Geometric questions:

- How does the network architecture affect the geometry of the function space?
- How does the geometry of the function space impact the training of the network?

In this talk:
What is the impact of changing from dense layers to convolutional layers?

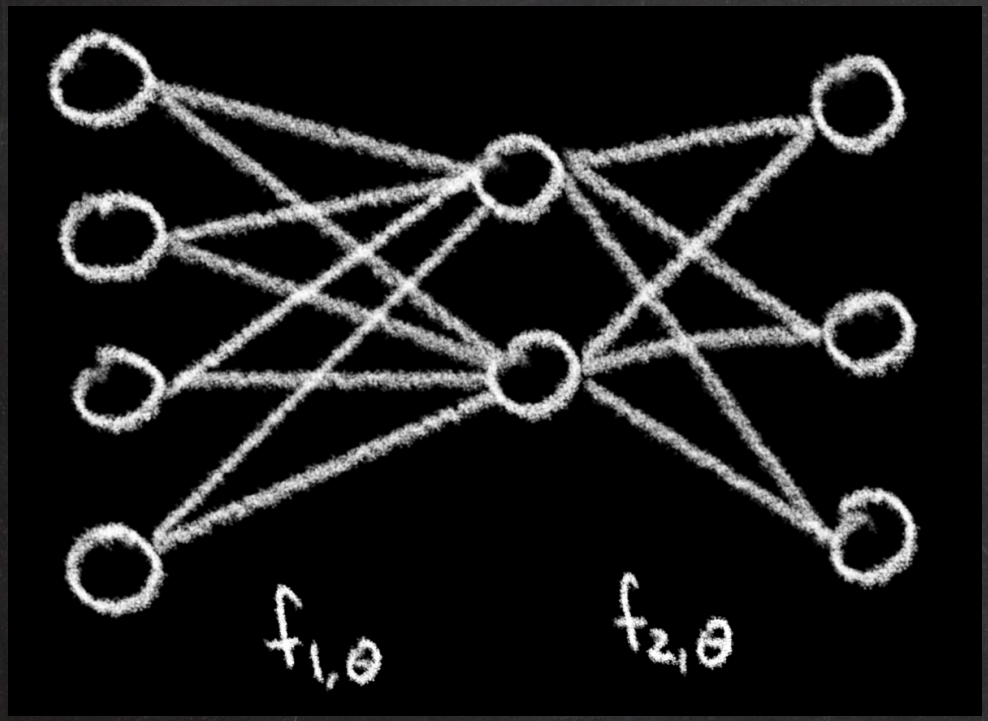
linear dense networks



In this example:

$$\mu : \mathbb{R}^{2\times 4} \times \mathbb{R}^{3\times 2} \longrightarrow \mathbb{R}^{3\times 4},$$
$$(W_1, W_2) \longmapsto W_2 W_1.$$

linear dense networks

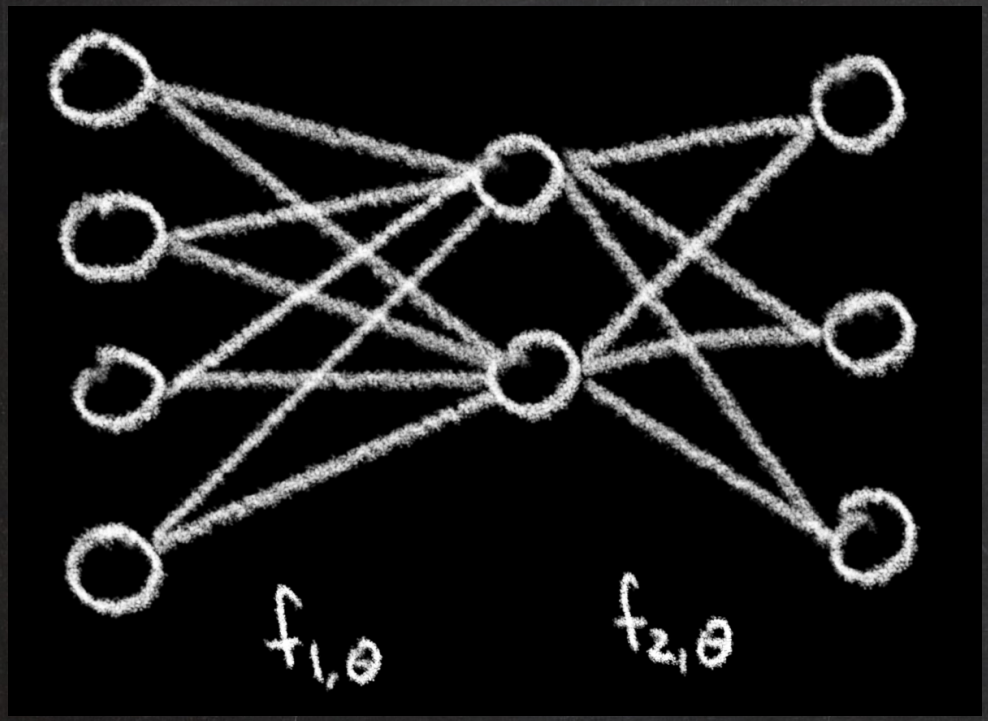


In this example:

$$\mu : \mathbb{R}^{2\times 4} \times \mathbb{R}^{3\times 2} \longrightarrow \mathbb{R}^{3\times 4},$$
$$(W_1, W_2) \longmapsto W_2 W_1.$$

$$\mathcal{M} = \{W \in \mathbb{R}^{3\times 4} \mid \operatorname{rank}(W) \leq 2\}$$

linear dense networks



In this example:

$$\mu : \mathbb{R}^{2\times 4} \times \mathbb{R}^{3\times 2} \longrightarrow \mathbb{R}^{3\times 4},$$
$$(W_1, W_2) \longmapsto W_2 W_1.$$

$$\mathcal{M} = \{W \in \mathbb{R}^{3\times 4} \mid \operatorname{rank}(W) \leq 2\}$$

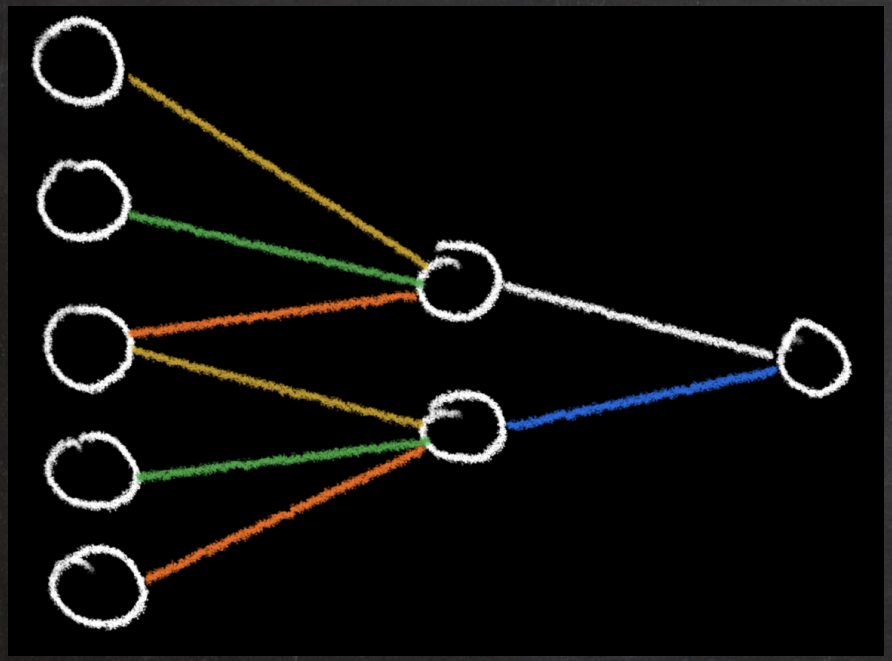
In general:

$$\mu : \mathbb{R}^{k_1\times k_0} \times \mathbb{R}^{k_2\times k_1} \times \ldots \times \mathbb{R}^{k_L\times k_{L-1}} \longrightarrow \mathbb{R}^{k_L\times k_0},$$
$$(W_1, W_2, \ldots, W_L) \longmapsto W_L \cdots W_2 W_1.$$

$\mathcal{M} = \{W \in \mathbb{R}^{k_L\times k_0} \mid \operatorname{rank}(W) \leq \min(k_0, \ldots, k_L)\}$ is an algebraic variety and we know its singularities etc.

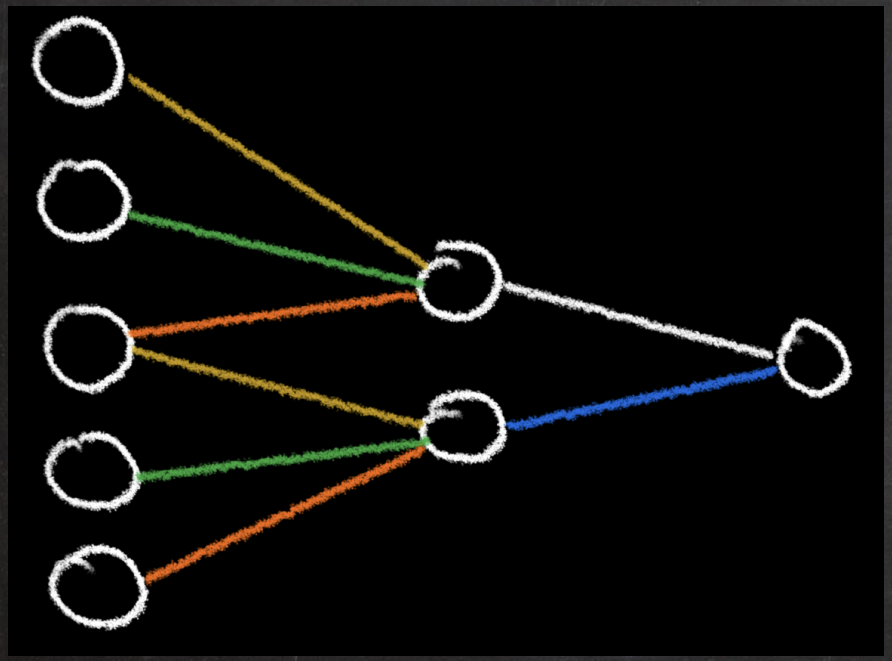
Linear Convolutional Networks (LCNs)

with 1D-convolutions



Linear Convolutional Networks (LCNs)

with 1D-convolutions



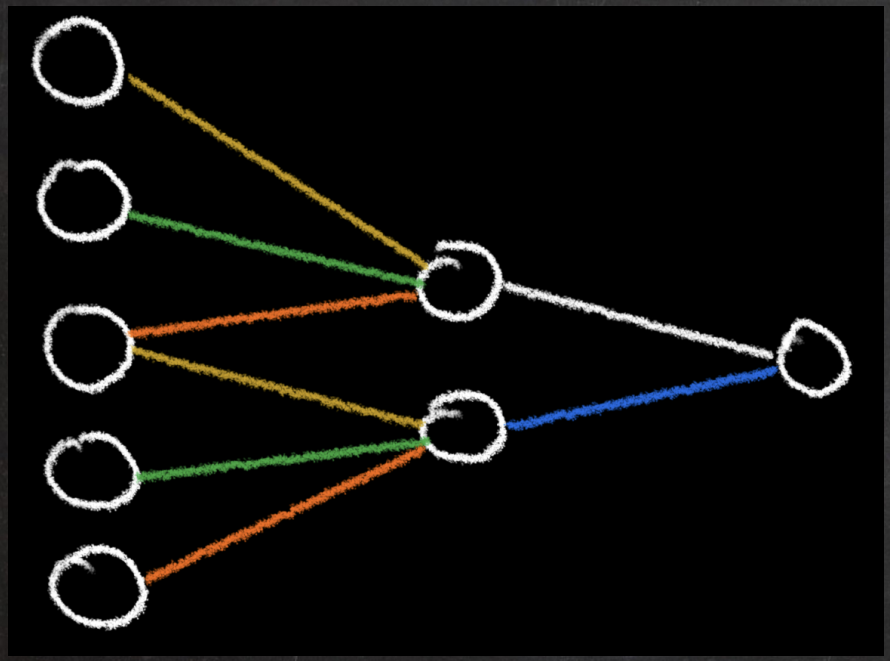
$$\mu : \mathbb{R}^3 \times \mathbb{R}^2 \longrightarrow \mathbb{R}^5,$$
$$(u, v) \longmapsto T_{v,1}\, T_{u,2}, \text{ where}$$

$$T_{u,2} = \begin{bmatrix} u_0 & u_1 & u_2 & 0 & 0 \\ 0 & 0 & u_0 & u_1 & u_2 \end{bmatrix}$$

$$T_{v,1} = \begin{bmatrix} v_0 & v_1 \end{bmatrix}$$

Linear Convolutional Networks (LCNs)

with 1D-convolutions



$$\mu : \mathbb{R}^3 \times \mathbb{R}^2 \longrightarrow \mathbb{R}^5,$$
$$(u, v) \longmapsto T_{v,1}\, T_{u,2}, \text{ where}$$

$$T_{u,2} = \begin{bmatrix} u_0 & u_1 & u_2 & 0 & 0 \\ 0 & 0 & u_0 & u_1 & u_2 \end{bmatrix}$$

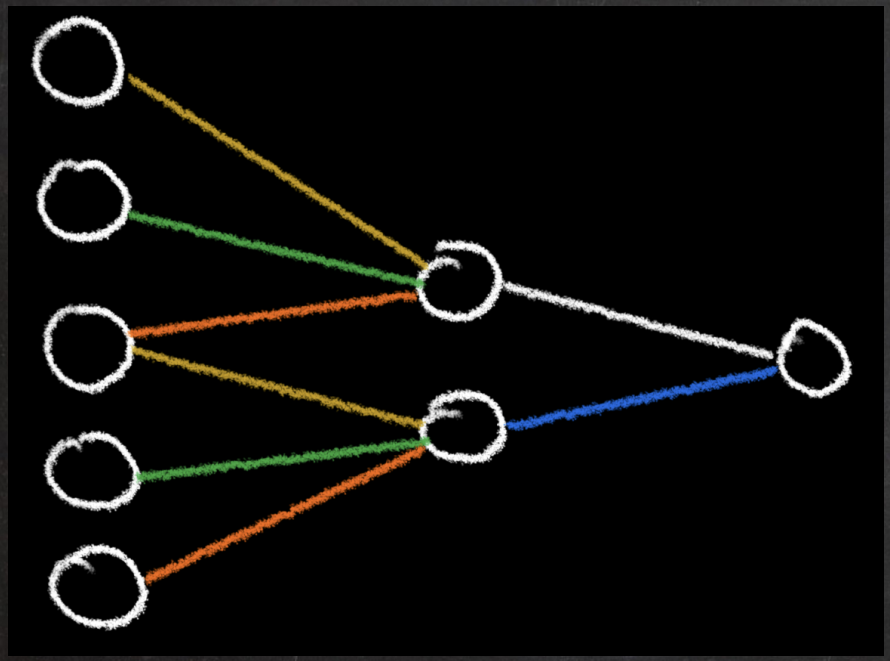
$$T_{v,1} = \begin{bmatrix} v_0 & v_1 \end{bmatrix}$$

In general: $\mu : (w_1, \ldots, w_L) \mapsto T_{w_L,s_L} \cdots T_{w_1,s_1}$, where

$$T_{w,s} = \begin{bmatrix} w_0 & \cdots & w_s & \cdots & w_{k-1} & & \\ & & w_0 & & \cdots & & w_{k-1} \\ & & & \ddots & & & \\ & & & & w_0 & \cdots & w_{k-1} \end{bmatrix}$$

Linear Convolutional Networks (LCNs)

with 1D-convolutions



$$\mu : \mathbb{R}^3 \times \mathbb{R}^2 \longrightarrow \mathbb{R}^5,$$
$$(u, v) \longmapsto T_{v,1}\, T_{u,2}, \text{ where}$$

$$T_{u,2} = \begin{bmatrix} u_0 & u_1 & u_2 & 0 & 0 \\ 0 & 0 & u_0 & u_1 & u_2 \end{bmatrix}$$

$$T_{v,1} = \begin{bmatrix} v_0 & v_1 \end{bmatrix}$$

In general: $\mu : (w_1, \ldots, w_L) \mapsto T_{w_L,s_L} \cdots T_{w_1,s_1}$, where

$$T_{w,s} = \begin{bmatrix} w_0 & \cdots & w_s & \cdots & w_{k-1} & & & \\ & & w_0 & & \cdots & & w_{k-1} & \\ & & & \ddots & & & & \ddots \\ & & & & & w_0 & \cdots & w_{k-1} \end{bmatrix}$$

is a convolutional matrix of stride s with filter w

LCNs & sparse polynomial factorization

Observation: $\mu(w_1, \ldots, w_L) = T_{w_L,s_L} \cdots T_{w_1,s_1}$ is again a convolutional matrix of stride $s_1 \cdots s_L$.

LCNs & sparse polynomial factorization

Observation: $\mu(w_1, \ldots, w_L) = T_{w_L,s_L} \cdots T_{w_1,s_1}$ is again a convolutional matrix of stride $s_1 \cdots s_L$. Its filter can be computed via polynomial multiplication:

LCNs & sparse polynomial factorization

Observation: $\mu(w_1, \ldots, w_L) = T_{w_L,s_L} \cdots T_{w_1,s_1}$ is again a convolutional matrix of stride $s_1 \cdots s_L$. Its filter can be computed via polynomial multiplication:

For $S \in \mathbb{Z}_{>0}$, let

$$\pi_S : \mathbb{R}^k \longrightarrow \mathbb{R}[x^S]_{\leq k-1},$$
$$v \longmapsto v_0 x^{S(k-1)} + v_1 x^{S(k-2)} + \ldots + v_{k-2} x^S + v_{k-1}$$

LCNs & sparse polynomial factorization

Observation: $\mu(w_1, \ldots, w_L) = T_{w_L,s_L} \cdots T_{w_1,s_1}$ is again a convolutional matrix of stride $s_1 \cdots s_L$. Its filter can be computed via polynomial multiplication:

For $S \in \mathbb{Z}_{>0}$, let

$$\begin{aligned} \pi_S : \mathbb{R}^k &\longrightarrow \mathbb{R}[x^S]_{\leq k-1}, \\ v &\longmapsto v_0 x^{S(k-1)} + v_1 x^{S(k-2)} + \ldots + v_{k-2} x^S + v_{k-1} \end{aligned}$$

and $\pi_S(T_{w,s}) := \pi_S(w)$. Then:

LCNs & sparse polynomial factorization

Observation: $\mu(w_1, \ldots, w_L) = T_{w_L,s_L} \cdots T_{w_1,s_1}$ is again a convolutional matrix of stride $s_1 \cdots s_L$. Its filter can be computed via polynomial multiplication:

For $S \in \mathbb{Z}_{>0}$, let

$$\begin{aligned} \pi_S : \mathbb{R}^k &\longrightarrow \mathbb{R}[x^S]_{\leq k-1}, \\ v &\longmapsto v_0 x^{S(k-1)} + v_1 x^{S(k-2)} + \ldots + v_{k-2} x^S + v_{k-1} \end{aligned}$$

and $\pi_S(T_{w,s}) := \pi_S(w)$. Then:

$$\pi_1(\mu(w_1, \ldots, w_L)) = \pi_{S_L}(w_L) \cdots \pi_{S_1}(w_1), \text{ where } S_i := s_1 \cdots s_{i-1}.$$

LCNs & sparse polynomial factorization

Observation: $\mu(w_1, \ldots, w_L) = T_{w_L,s_L} \cdots T_{w_1,s_1}$ is again a convolutional matrix of stride $s_1 \cdots s_L$. Its filter can be computed via polynomial multiplication:

For $S \in \mathbb{Z}_{>0}$, let

$$\begin{aligned} \pi_S : \mathbb{R}^k &\longrightarrow \mathbb{R}[x^S]_{\leq k-1}, \\ v &\longmapsto v_0 x^{S(k-1)} + v_1 x^{S(k-2)} + \ldots + v_{k-2} x^S + v_{k-1} \end{aligned}$$

and $\pi_S(T_{w,s}) := \pi_S(w)$. Then:

$$\pi_1(\mu(w_1, \ldots, w_L)) = \pi_{S_L}(w_L) \cdots \pi_{S_1}(w_1), \text{ where } S_i := s_1 \cdots s_{i-1}.$$

Hence, we reinterpret μ as

$$\begin{aligned} \mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} &\longrightarrow \mathbb{R}[x]_{\leq d_1 S_1 + \ldots + d_L S_L}, \\ (P_1, \ldots, P_L) &\longmapsto P_L \cdots P_1 \end{aligned}$$

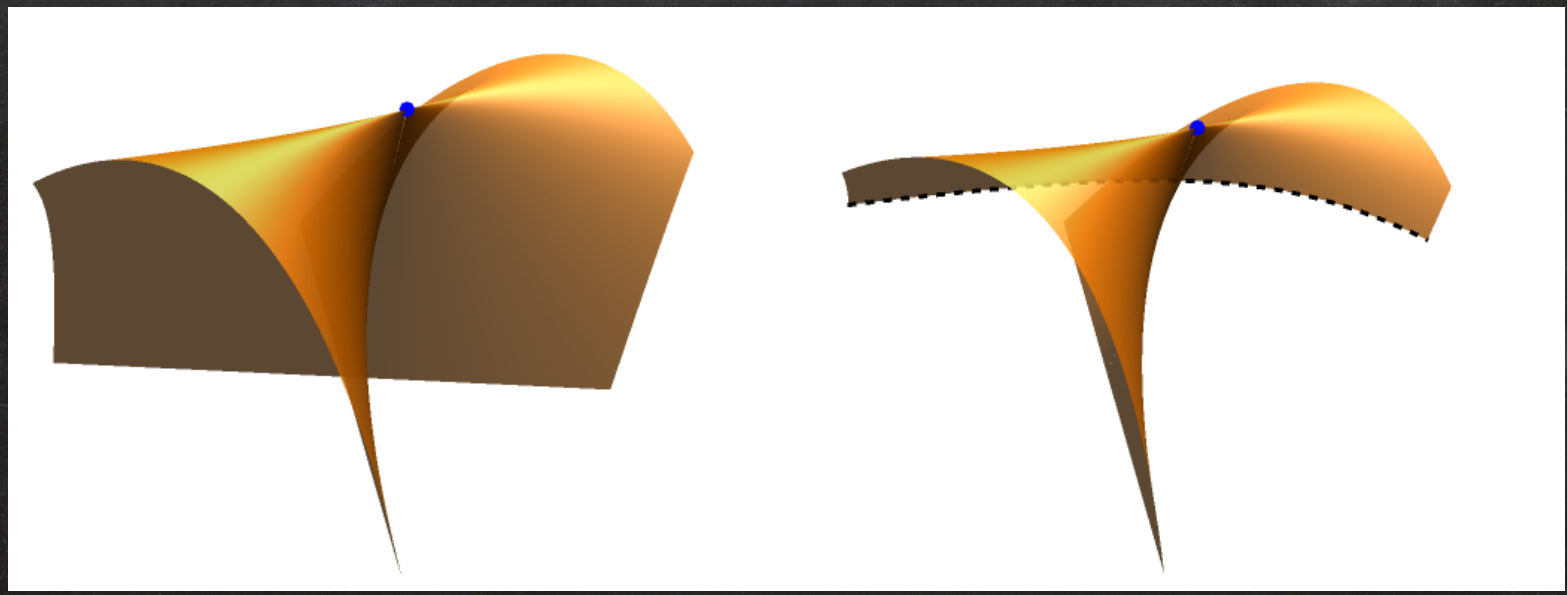
LCN function spaces

$$\mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} \longrightarrow \mathbb{R}[x]_{\leq d}, \text{where } d := \sum_i d_i S_i$$
$$(P_1, \ldots, P_L) \longmapsto P_L \cdots P_1,$$

LCN function spaces

$$\mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} \longrightarrow \mathbb{R}[x]_{\leq d}, \text{where } d := \textstyle\sum_i d_i S_i$$
$$(P_1, \ldots, P_L) \longmapsto P_L \cdots P_1,$$

Theorem: The function space $\mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu)$ is a semi-algebraic, Euclidean-closed subset of $\mathbb{R}[x]_{\leq d}$ of dimension $d_1 + \ldots + d_L + 1$.



$\mu : \mathbb{R}[x]_{\leq 2} \times \mathbb{R}[x^2]_{\leq 1} \to \mathbb{R}[x]_{\leq 4}$

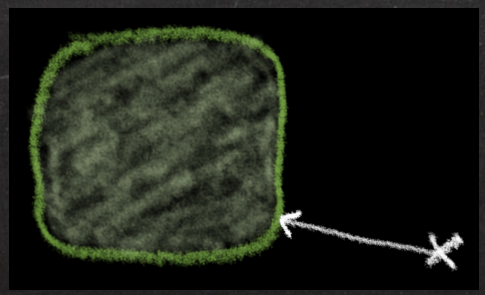
$\mu : \mathbb{R}[x]_{\leq 1} \times \mathbb{R}[x]_{\leq 1} \times \mathbb{R}[x^2]_{\leq 1} \to \mathbb{R}[x]_{\leq 4}$

LCN function spaces

$$\mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} \longrightarrow \mathbb{R}[x]_{\leq d}, \text{where } d := \sum_i d_i S_i$$
$$(P_1, \ldots, P_L) \longmapsto P_L \cdots P_1,$$

Theorem: The function space $\mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu)$ is a semi-algebraic, Euclidean-closed subset of $\mathbb{R}[x]_{\leq d}$ of dimension $d_1 + \ldots + d_L + 1$.

Corollary: $\mathcal{M}_{\boldsymbol{d},\boldsymbol{S}}$ is full-dimensional in $\mathbb{R}[x]_{\leq d}$ if and only if all strides $s_i = 1$.



LCN function spaces

$$\mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} \longrightarrow \mathbb{R}[x]_{\leq d}, \text{where } d := \sum_i d_i S_i$$
$$(P_1, \ldots, P_L) \longmapsto P_L \cdots P_1,$$

Example: $\boldsymbol{s} = (1, 1)$ and $\boldsymbol{d} = (1, 1)$.
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) =$

LCN function spaces

$$\mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} \longrightarrow \mathbb{R}[x]_{\leq d}, \text{where } d := \sum_i d_i S_i$$
$$(P_1, \ldots, P_L) \longmapsto P_L \cdots P_1,$$

Example: $\boldsymbol{s} = (1, 1)$ and $\boldsymbol{d} = (1, 1)$.
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) = \{$ quadratic polynomials with 2 real roots $\}$

LCN function spaces

$$\mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} \longrightarrow \mathbb{R}[x]_{\leq d}, \text{where } d := \sum_i d_i S_i$$
$$(P_1, \ldots, P_L) \longmapsto P_L \cdots P_1,$$

Example: $\boldsymbol{s} = (1, 1)$ and $\boldsymbol{d} = (1, 1)$.
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) = \{$ quadratic polynomials with 2 real roots $\}$

Example: $\boldsymbol{s} = (1, 1)$ and $\boldsymbol{d} = (1, 2)$
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) =$

LCN function spaces

$$\mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} \longrightarrow \mathbb{R}[x]_{\leq d}, \text{where } d := \sum_i d_i S_i$$
$$(P_1, \ldots, P_L) \longmapsto P_L \cdots P_1,$$

Example: $\boldsymbol{s} = (1, 1)$ and $\boldsymbol{d} = (1, 1)$.
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) = \{$ quadratic polynomials with 2 real roots $\}$

Example: $\boldsymbol{s} = (1, 1)$ and $\boldsymbol{d} = (1, 2)$
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) = \{$ all cubic polynomials $\}$

LCN function spaces

$$\mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} \longrightarrow \mathbb{R}[x]_{\leq d}, \text{where } d := \sum_i d_i S_i$$
$$(P_1, \ldots, P_L) \longmapsto P_L \cdots P_1,$$

Example: $\boldsymbol{s} = (1,1)$ and $\boldsymbol{d} = (1,1)$.
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) = \{$ quadratic polynomials with 2 real roots $\}$

Example: $\boldsymbol{s} = (1,1)$ and $\boldsymbol{d} = (1,2)$
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) = \{$ all cubic polynomials $\}$

Proposition: $\boldsymbol{s} = (1,1,\ldots,1)$ and $\boldsymbol{d} = (d_1, d_2, \ldots, d_L)$.
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) = \mathbb{R}[x]_{\leq d}$ if and only if

LCN function spaces

$$\mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} \longrightarrow \mathbb{R}[x]_{\leq d}, \text{where } d := \sum_i d_i S_i$$
$$(P_1, \ldots, P_L) \longmapsto P_L \cdots P_1,$$

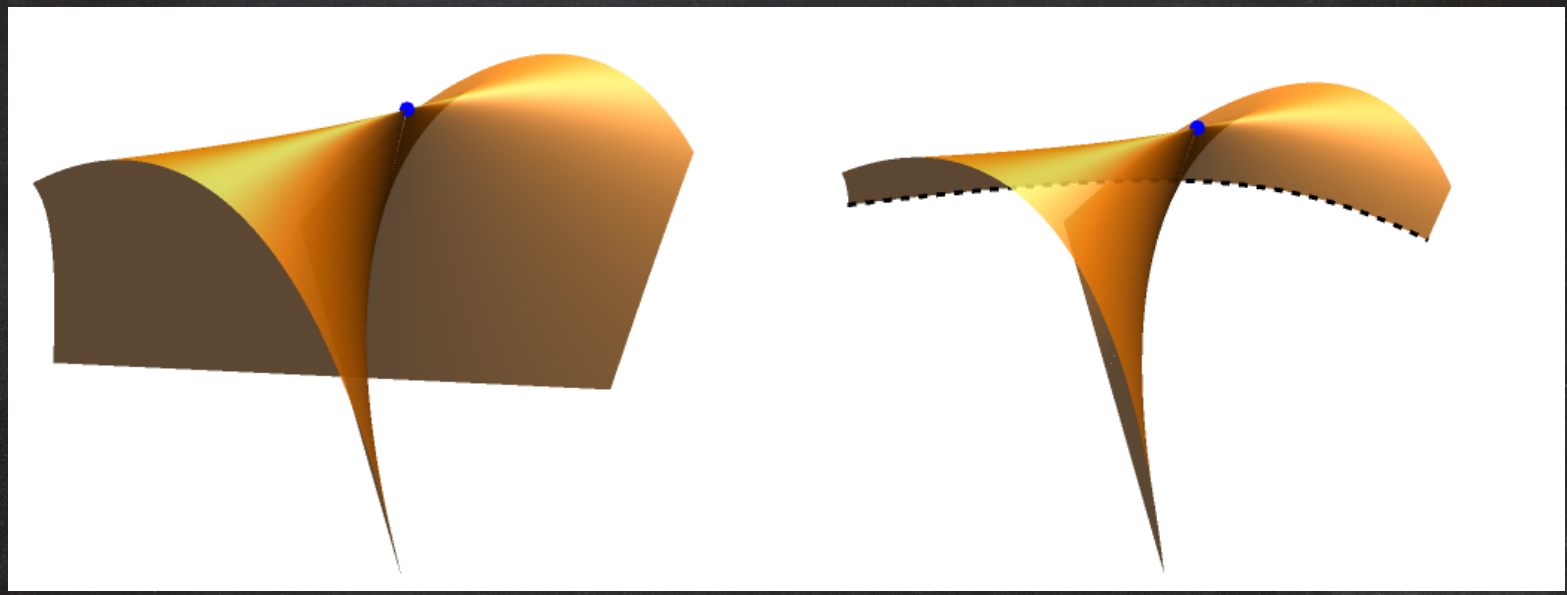
Example: $\boldsymbol{s} = (1,1)$ and $\boldsymbol{d} = (1,1)$.
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) = \{$ quadratic polynomials with 2 real roots $\}$

Example: $\boldsymbol{s} = (1,1)$ and $\boldsymbol{d} = (1,2)$
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) = \{$ all cubic polynomials $\}$

Proposition: $\boldsymbol{s} = (1,1,\ldots,1)$ and $\boldsymbol{d} = (d_1, d_2, \ldots, d_L)$.
$\Rightarrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \mathrm{im}(\mu) = \mathbb{R}[x]_{\leq d}$ if and only if at most one d_i is odd.

LCN function spaces

$$\mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} \longrightarrow \mathbb{R}[x]_{\leq d}, \text{where } d := \sum_i d_i S_i$$
$$(P_1, \ldots, P_L) \longmapsto P_L \cdots P_1,$$



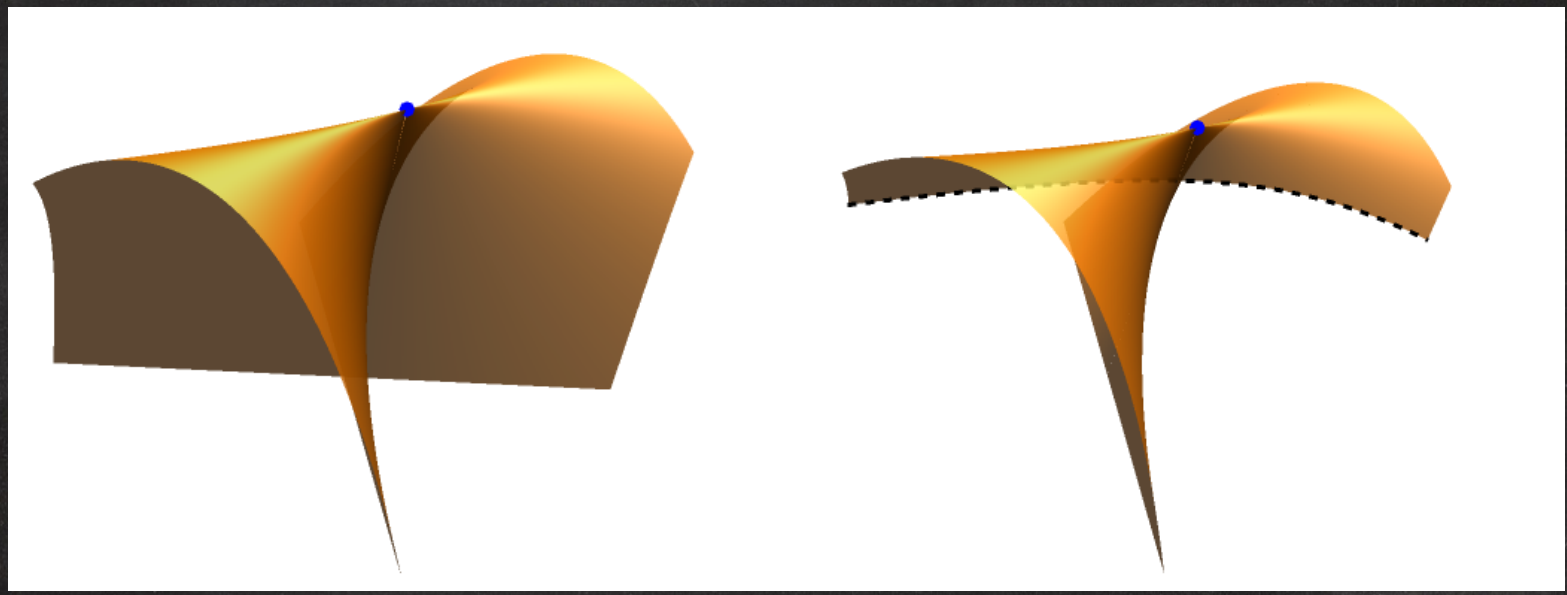
$\mu : \mathbb{R}[x]_{\leq 2} \times \mathbb{R}[x^2]_{\leq 1} \to \mathbb{R}[x]_{\leq 4}$

$\mu : \mathbb{R}[x]_{\leq 1} \times \mathbb{R}[x]_{\leq 1} \times \mathbb{R}[x^2]_{\leq 1} \to \mathbb{R}[x]_{\leq 4}$

$\uparrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \{(a + bx + c^2)(d + ex^2) \mid a, \ldots, e \in \mathbb{R}\}$

LCN function spaces

$$\mu : \mathbb{R}[x^{S_1}]_{\leq d_1} \times \ldots \times \mathbb{R}[x^{S_L}]_{\leq d_L} \longrightarrow \mathbb{R}[x]_{\leq d}, \text{where } d := \sum_i d_i S_i$$
$$(P_1, \ldots, P_L) \longmapsto P_L \cdots P_1,$$



$\mu : \mathbb{R}[x]_{\leq 2} \times \mathbb{R}[x^2]_{\leq 1} \to \mathbb{R}[x]_{\leq 4}$

$\mu : \mathbb{R}[x]_{\leq 1} \times \mathbb{R}[x]_{\leq 1} \times \mathbb{R}[x^2]_{\leq 1} \to \mathbb{R}[x]_{\leq 4}$

$$\uparrow \mathcal{M}_{\boldsymbol{d},\boldsymbol{S}} = \{(a + bx + c^2)(d + ex^2) \mid a, \ldots, e \in \mathbb{R}\}$$
$$= \{A + Bx + Cx^2 + Dx^3 + Ex^4 \mid AD^2 + B^2E = BCD,\ C^2 \geq 4AE\}$$

Convolutional Tensors

A convolution on D-dimensional signals is defined using a filter w of format

$$k^{(1)} \times \cdots \times k^{(D)}$$

and a stride tuple $s = (s^{(1)}, \ldots, s^{(D)}) \in \mathbb{Z}^D_{\geq 0}$.

Convolutional Tensors

A convolution on D-dimensional signals is defined using a filter w of format

$$k^{(1)} \times \cdots \times k^{(D)}$$

and a stride tuple $s = (s^{(1)}, \ldots, s^{(D)}) \in \mathbb{Z}^D_{\geq 0}$.

The convolution is a linear map α_w that produces an output tensor of format $d^{(1)} \times \cdots \times d^{(D)}$ from an input tensor X of format $(s^{(1)}(d^{(1)} - 1) + k^{(1)}) \times \cdots \times (s^{(D)}(d^{(D)} - 1) + k^{(D)})$:

$$(\alpha_{w,s}(X))_{i_1,\ldots,i_D} = \sum_{j_1=0}^{k^{(1)}-1} \cdots \sum_{j_D=0}^{k^{(D)}-1} w_{j_1,\ldots,j_D} X_{i_1 s^{(1)}+j_1,\ldots,i_D s^{(D)}+j_D}$$

for $i_m = 0, 1, \ldots, d^{(m)} - 1$.

Convolutional Tensors

A convolution on D-dimensional signals is defined using a filter w of format

$$k^{(1)} \times \cdots \times k^{(D)}$$

and a stride tuple $s = (s^{(1)}, \ldots, s^{(D)}) \in \mathbb{Z}^D_{\geq 0}$.

The convolution is a linear map α_w that produces an output tensor of format $d^{(1)} \times \cdots \times d^{(D)}$ from an input tensor X of format $(s^{(1)}(d^{(1)} - 1) + k^{(1)}) \times \cdots \times (s^{(D)}(d^{(D)} - 1) + k^{(D)})$:

$$(\alpha_{w,s}(X))_{i_1,\ldots,i_D} = \sum_{j_1=0}^{k^{(1)}-1} \cdots \sum_{j_D=0}^{k^{(D)}-1} w_{j_1,\ldots,j_D} X_{i_1 s^{(1)}+j_1,\ldots,i_D s^{(D)}+j_D}$$

for $i_m = 0, 1, \ldots, d^{(m)} - 1$.

This linear map is represented by a $2D$-dimensional convolutional tensor T of format $d^{(1)} \times \cdots \times d^{(D)} \times (s^{(1)}(d^{(1)} - 1) + k^{(1)}) \times \cdots \times (s^{(D)}(d^{(D)} - 1) + k^{(D)})$

Convolutional Tensors

Example: $D = 2, \quad s = (1, 1), \quad$ filter w of format 2×2:

The convolution of the input

$$X = \begin{bmatrix} * & * \\ *\circ & *\circ \\ \circ & \circ \end{bmatrix}$$

with filter w yields an output of size 2×1.

The associated convolutional tensor $T \in \mathbb{R}^{2\times 1\times 3\times 2}$ has the slices:

$$T_{00::} = \begin{bmatrix} w_{00} & w_{01} \\ w_{10} & w_{11} \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad T_{10::} = \begin{bmatrix} 0 & 0 \\ w_{00} & w_{01} \\ w_{10} & w_{11} \end{bmatrix}$$

LCNs

A linear convolutional network composes L such convolutions, yielding another convolution!

The end-to-end convolution has:

- Stride: $(s_1^{(1)} \cdots s_L^{(1)}, \ldots, s_1^{(D)} \cdots s_L^{(D)})$

LCNs

A linear convolutional network composes L such convolutions, yielding another convolution!

The end-to-end convolution has:

- Stride: $(s_1^{(1)} \cdots s_L^{(1)}, \ldots, s_1^{(D)} \cdots s_L^{(D)})$
- Filter computed via polynomial multiplication:

  For

  - $\mathrm{x} = (x_1, \ldots, y_D)$,
  - $S = (S^{(1)}, \ldots, S^{(D)})$,,
  - $k = (k^{(1)}, \ldots, k^{(D)})$,

  we denote by

  $$\mathbb{R}[\mathrm{x}^S]_{\leq \mathrm{k}-1}$$

  the vector space of polynomials of degree at most $k^{(m)} - 1$ in $x_m^{S^{(m)}}$.

LCNs & sparse polynomial factorization

We identify the space $\mathbb{R}[x^S]_{\leq k-1}$ with the tensor space $\mathbb{R}^{k^{(1)} \times \cdots \times k^{(D)}}$:

$$\pi_S : v \mapsto \sum_{i_1=0}^{k^{(1)}-1} \cdots \sum_{i_D=0}^{k^{(D)}-1} v_{i_1,\ldots,i_D} \, x_1^{S^{(1)} i_1} \cdots x_D^{S^{(D)} i_D}$$

LCNs & sparse polynomial factorization

We identify the space $\mathbb{R}[x^S]_{\leq k-1}$ with the tensor space $\mathbb{R}^{k^{(1)} \times \cdots \times k^{(D)}}$:

$$\pi_S : v \mapsto \sum_{i_1=0}^{k^{(1)}-1} \cdots \sum_{i_D=0}^{k^{(D)}-1} v_{i_1,\ldots,i_D} \, x_1^{S^{(1)} i_1} \cdots x_D^{S^{(D)} i_D}$$

Proposition: $\pi_{(1,\ldots,1)}$ of the end-to-end filter is

$$\pi_{S_L}(W_L) \cdots \pi_{S_1}(W_1),$$

where $S_i^{(m)} := s_1^{(m)} \cdots s_{i-1}^{(m)}$.

LCNs & sparse polynomial factorization

We identify the space $\mathbb{R}[x^S]_{\leq k-1}$ with the tensor space $\mathbb{R}^{k^{(1)} \times \cdots \times k^{(D)}}$:

$$\pi_S : v \mapsto \sum_{i_1=0}^{k^{(1)}-1} \cdots \sum_{i_D=0}^{k^{(D)}-1} v_{i_1,\ldots,i_D} \, x_1^{S^{(1)} i_1} \cdots x_D^{S^{(D)} i_D}$$

Proposition: $\pi_{(1,\ldots,1)}$ of the end-to-end filter is

$$\pi_{S_L}(w_L) \cdots \pi_{S_1}(w_1),$$

where $S_i^{(m)} := s_1^{(m)} \cdots s_{i-1}^{(m)}$.

This is factorization is typically unique!

https://doi.org/10.1007/s44426-025-00002-2

ARTICLE

Check for updates

Algebraic complexity and neurovariety of linear convolutional networks

Vahid Shahverdi[1]

Received: 10 April 2025 / Accepted: 28 April 2025

What about higher dimensions?

Abstract

In this paper, we study linear convolutional networks with one-dimensional filters and arbitrary strides. The neuromanifold of such a network is a semialgebraic set, represented by a space of polynomials admitting specific factorizations. Introducing a recursive algorithm, we generate polynomial equations whose common zero locus corresponds to the Zariski closure of the corresponding neuromanifold. Furthermore, we explore the algebraic complexity of training these networks employing tools from metric algebraic geometry. Our findings reveal that the number of all complex critical points in the optimization of such a network is equal to the generic Euclidean distance degree of a Segre variety. Notably, this count significantly surpasses the number of critical points encountered in the training of a fully connected linear network with the same number of parameters.

Keywords Function space description of neural networks · Linear networks · Euclidean distance degree · Critical points