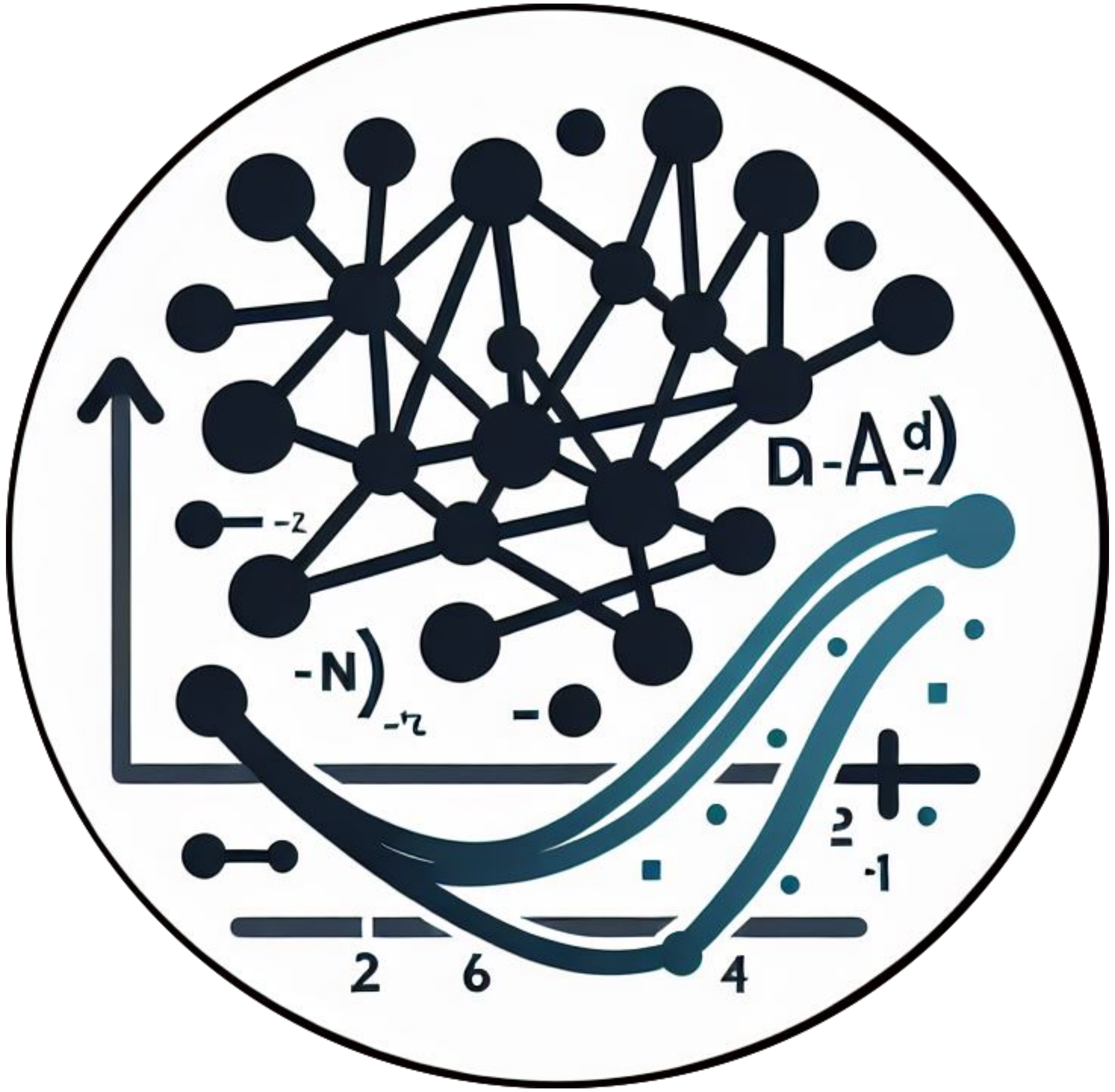
Kolmogorov Arnold Networks



Deep Learning for Engineers

Andrew Ning
aning@byu.edu

submitted April 2024

KAN: Kolmogorov–Arnold Networks

Ziming Liu[1,4*] **Yixuan Wang**[2] **Sachin Vaidya**[1] **Fabian Ruehle**[3,4]
James Halverson[3,4] **Marin Soljačić**[1,4] **Thomas Y. Hou**[2] **Max Tegmark**[1,4]

[1] Massachusetts Institute of Technology
[2] California Institute of Technology
[3] Northeastern University
[4] The NSF Institute for Artificial Intelligence and Fundamental Interactions

Abstract

Inspired by the Kolmogorov-Arnold representation theorem, we propose Kolmogorov-Arnold Networks (KANs) as promising alternatives to Multi-Layer Perceptrons (MLPs). While MLPs have *fixed* activation functions on *nodes* ("neurons"), KANs have *learnable* activation functions on *edges* ("weights"). KANs have no linear weights at all – every

Kolmogorov-Arnold Theorem

multivariable function f can be expressed as composition of univariate functions and addition

$$f(x) = \sum_{q=1}^{2n+1} \psi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$$

$$y = f(x_1, x_2, x_3, \ldots, x_n)$$

multivariate function

$$y = f(x_1, x_2, x_3, \ldots, x_n)$$ multivariate function

$$\phi_1(x_1) + \phi_2(x_2) + \ldots + \phi_n(x_n) = \sum_{p=1}^{n} \phi_p(x_p)$$ sum of univariate functions

$$y = f(x_1, x_2, x_3, \ldots, x_n)$$

multivariate function

$$\phi_1(x_1) + \phi_2(x_2) + \ldots + \phi_n(x_n) = \sum_{p=1}^{n} \phi_p(x_p)$$

sum of univariate functions

$$\psi\left(\sum_{p=1}^{n} \phi_p(x_p)\right)$$

pass through another univariate function:

$$y = f(x_1, x_2, x_3, \ldots, x_n)$$

multivariate function

$$\phi_1(x_1) + \phi_2(x_2) + \ldots + \phi_n(x_n) = \sum_{p=1}^{n} \phi_p(x_p)$$

sum of univariate functions

$$\psi\left(\sum_{p=1}^{n} \phi_p(x_p)\right)$$

pass through another univariate function:

$$y = \sum_{q=1}^{2n+1} \psi_q\left(\sum_{p=1}^{n} \phi_{qp}(x_p)\right)$$

sum over many such functions:

$$
\begin{aligned}
&\psi_1(\phi_{11}(x_1) + \phi_{12}(x_2) + \ldots + \phi_{1n}(x_n)) \quad + \\
&\psi_2(\phi_{21}(x_1) + \phi_{22}(x_2) + \ldots + \phi_{2n}(x_n)) \quad + \\
&\qquad\qquad \vdots \\
&\psi_m(\phi_{m1}(x_1) + \phi_{m2}(x_2) + \ldots + \phi_{mn}(x_n))
\end{aligned}
$$

Two Layers

$$y = \sum_{q=1}^{2n+1} \psi_q \left(\sum_{p=1}^{n} \phi_{qp}(x_p) \right)$$

$$[n, 2n + 1, 1]$$

Layer 1

$$z = \begin{bmatrix} \phi_{11}(x_1) + \phi_{12}(x_2) + \ldots + \phi_{1n}(x_n)) \\ \phi_{21}(x_1) + \phi_{22}(x_2) + \ldots + \phi_{2n}(x_n)) \\ \vdots \\ \phi_{m1}(x_1) + \phi_{m2}(x_2) + \ldots + \phi_{mn}(x_n) \end{bmatrix}$$

Layer 2

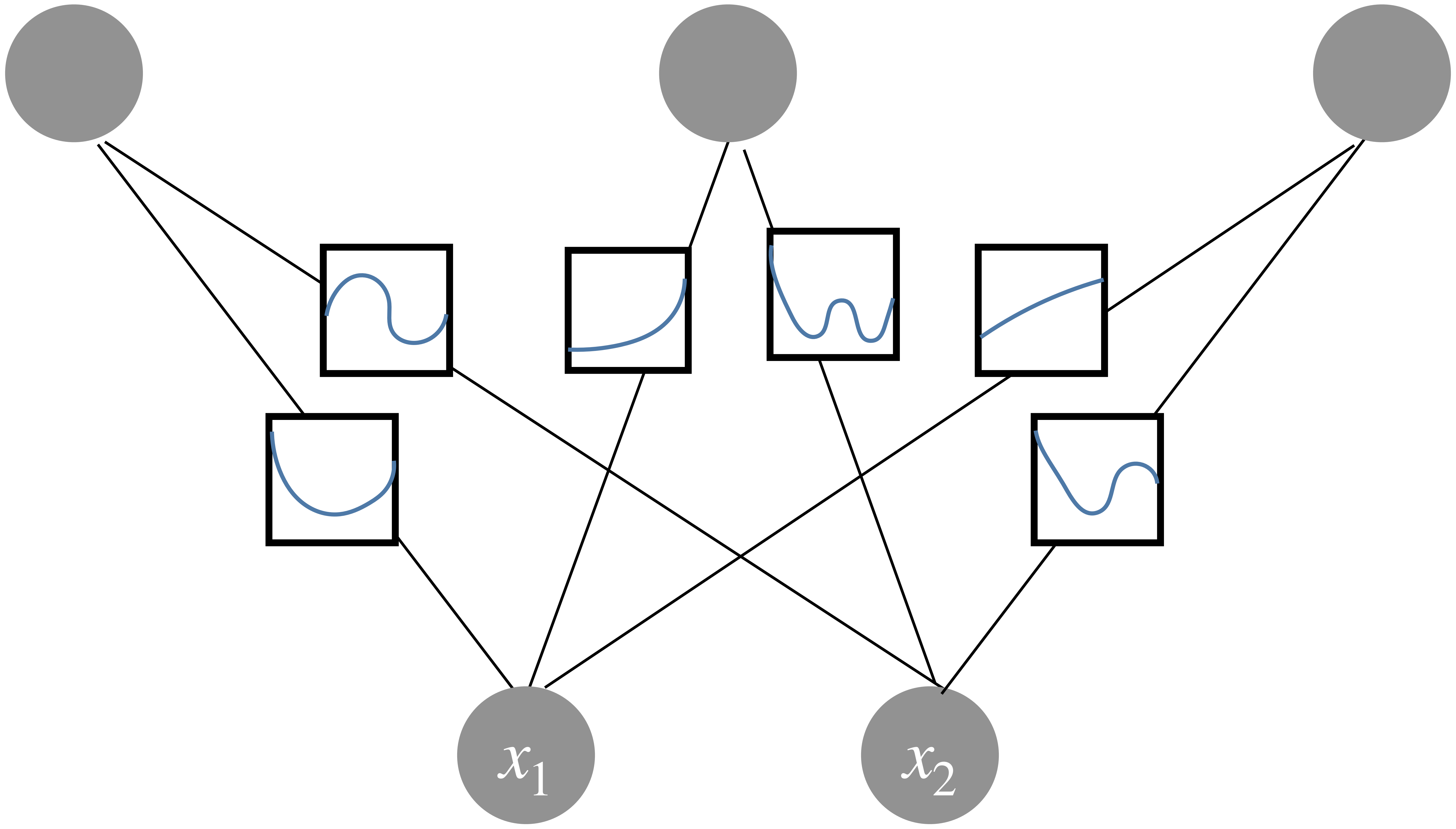
$$y = \left[\psi_1(z_1) + \psi_2(z_2) + \ldots + \psi_m(z_m)\right]$$

Generalize

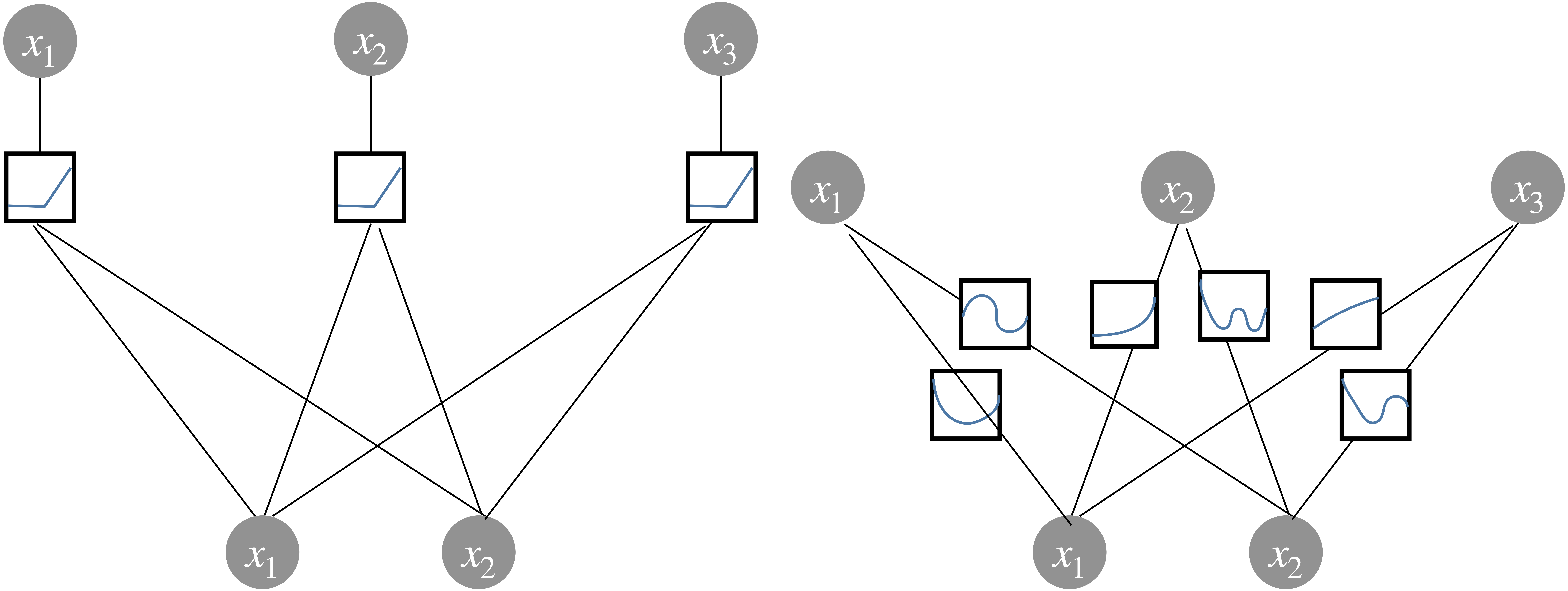
$$\underbrace{\begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{m1} & \phi_{m2} & \cdots & \phi_{mn} \end{bmatrix}}_{\Phi^{(l)}} \left(\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \right)$$

$$y = (\Phi^{(L)} \circ \ldots \circ \Phi^{(3)} \circ \Phi^{(2)} \circ \Phi^{(1)})x$$

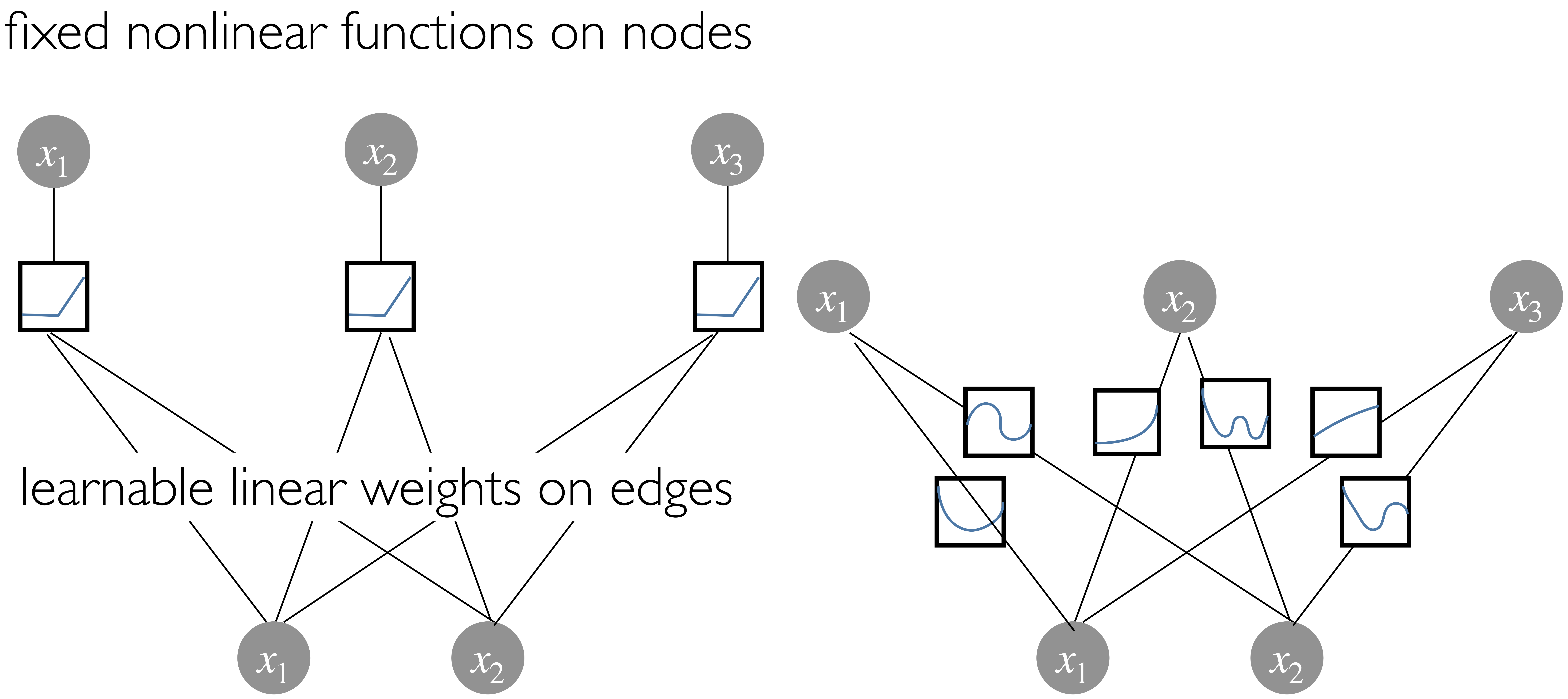
(Φ are functions, not numbers)

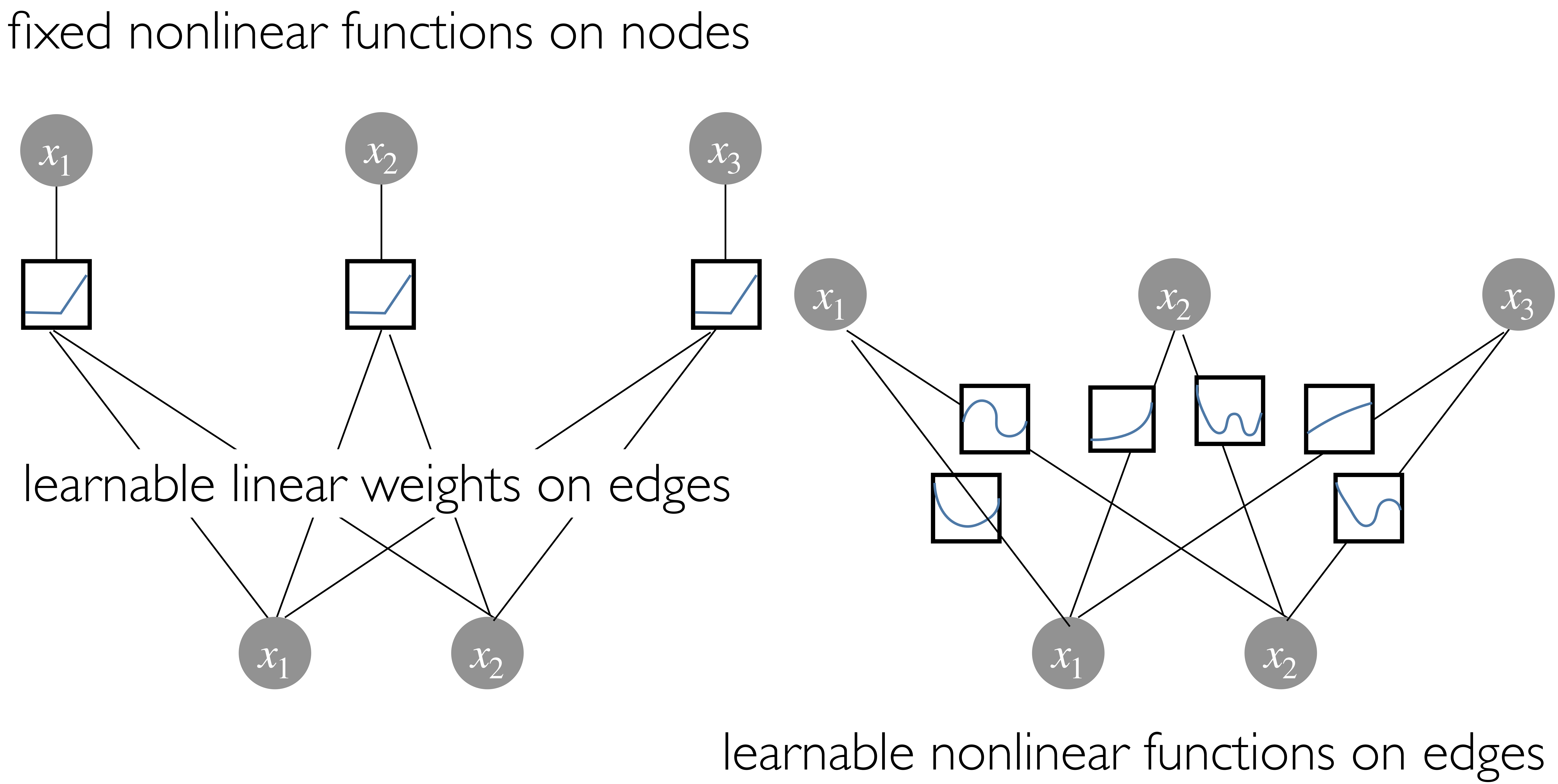


x_1
x_2

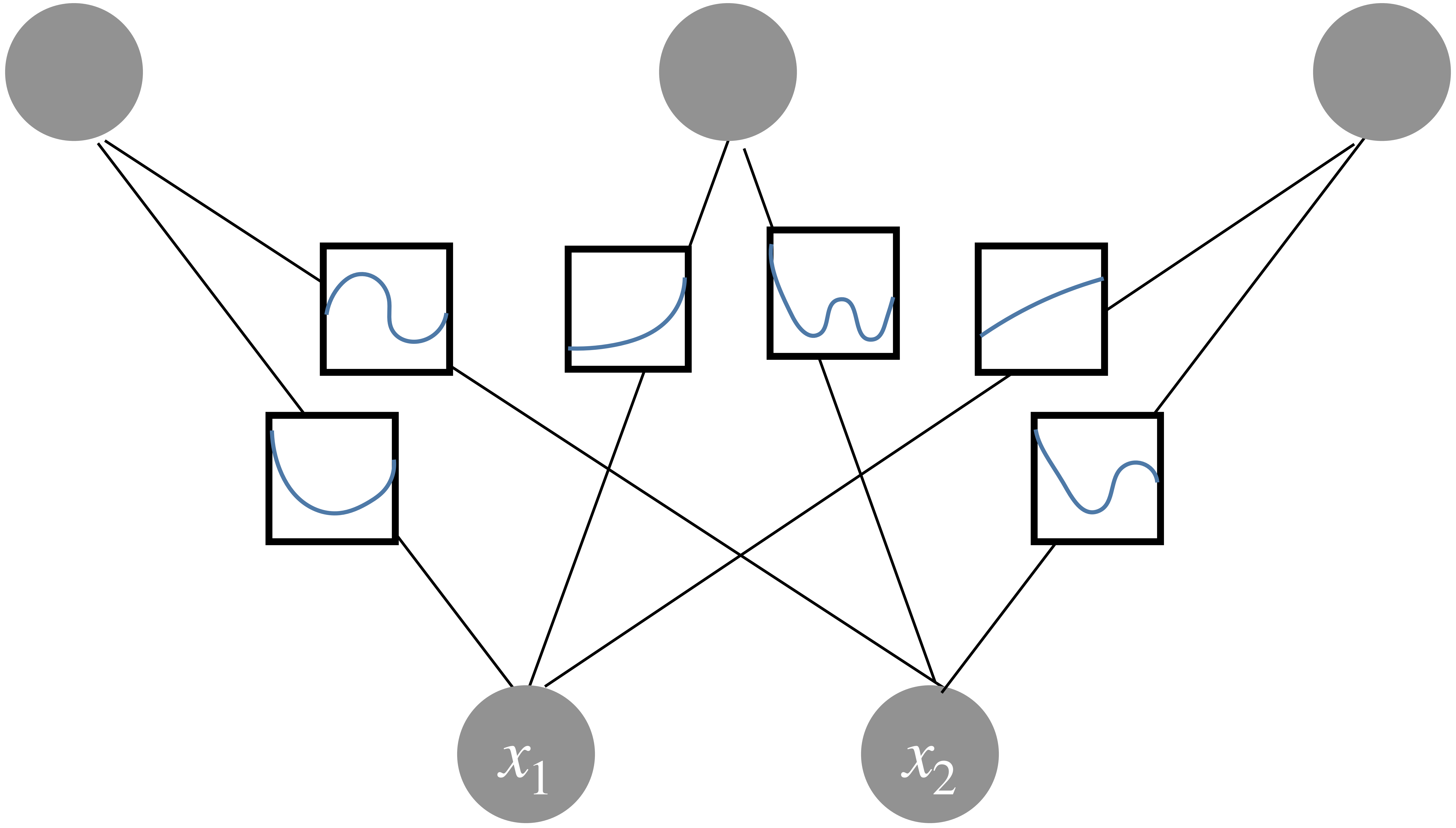


x_1
x_2
x_3
x_1
x_2
x_1
x_2
x_3
x_1
x_2

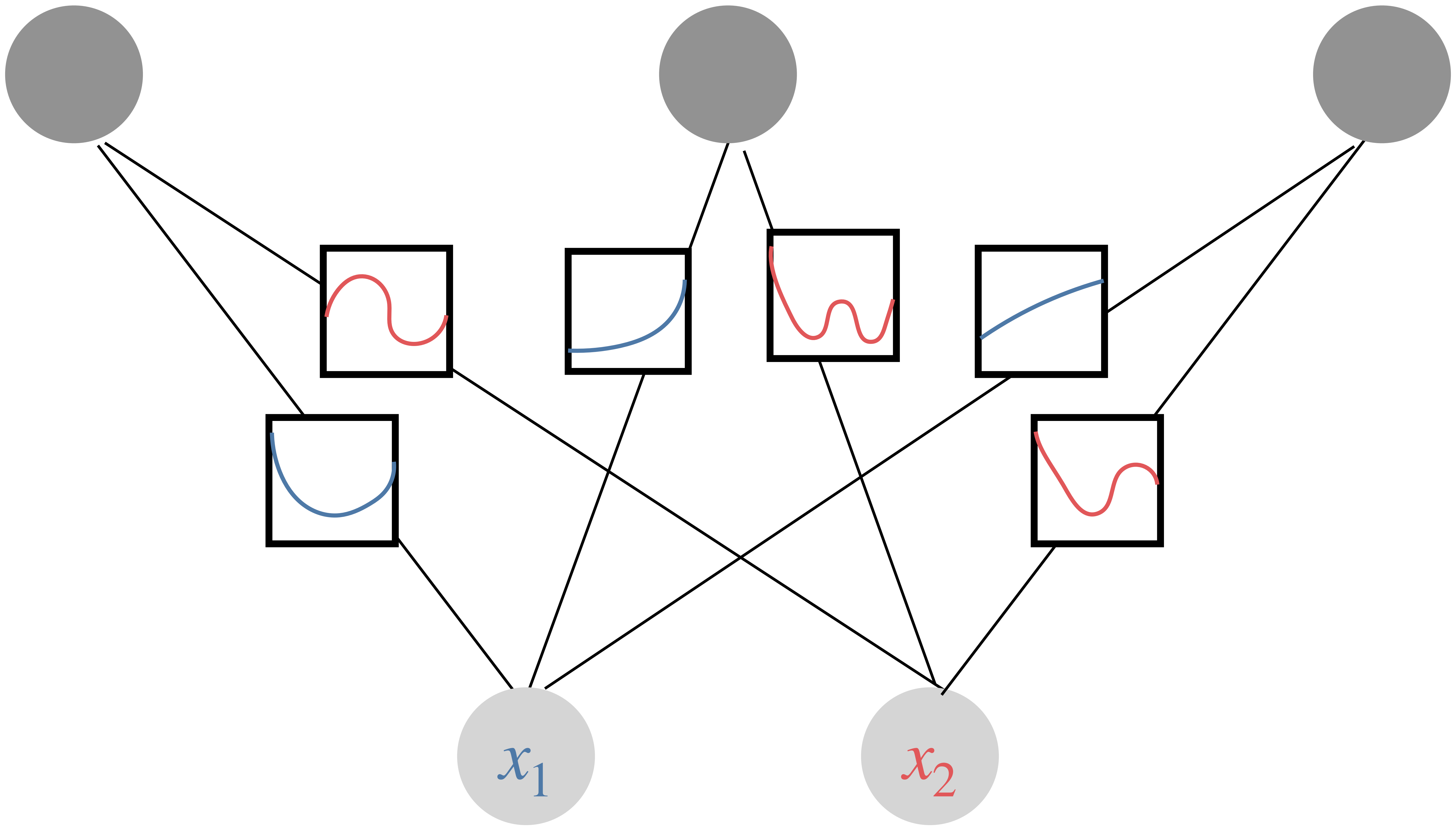

fixed nonlinear functions on nodes
learnable linear weights on edges
x_1
x_2
x_3
x_1
x_2
x_1
x_2
x_3
x_1
x_2



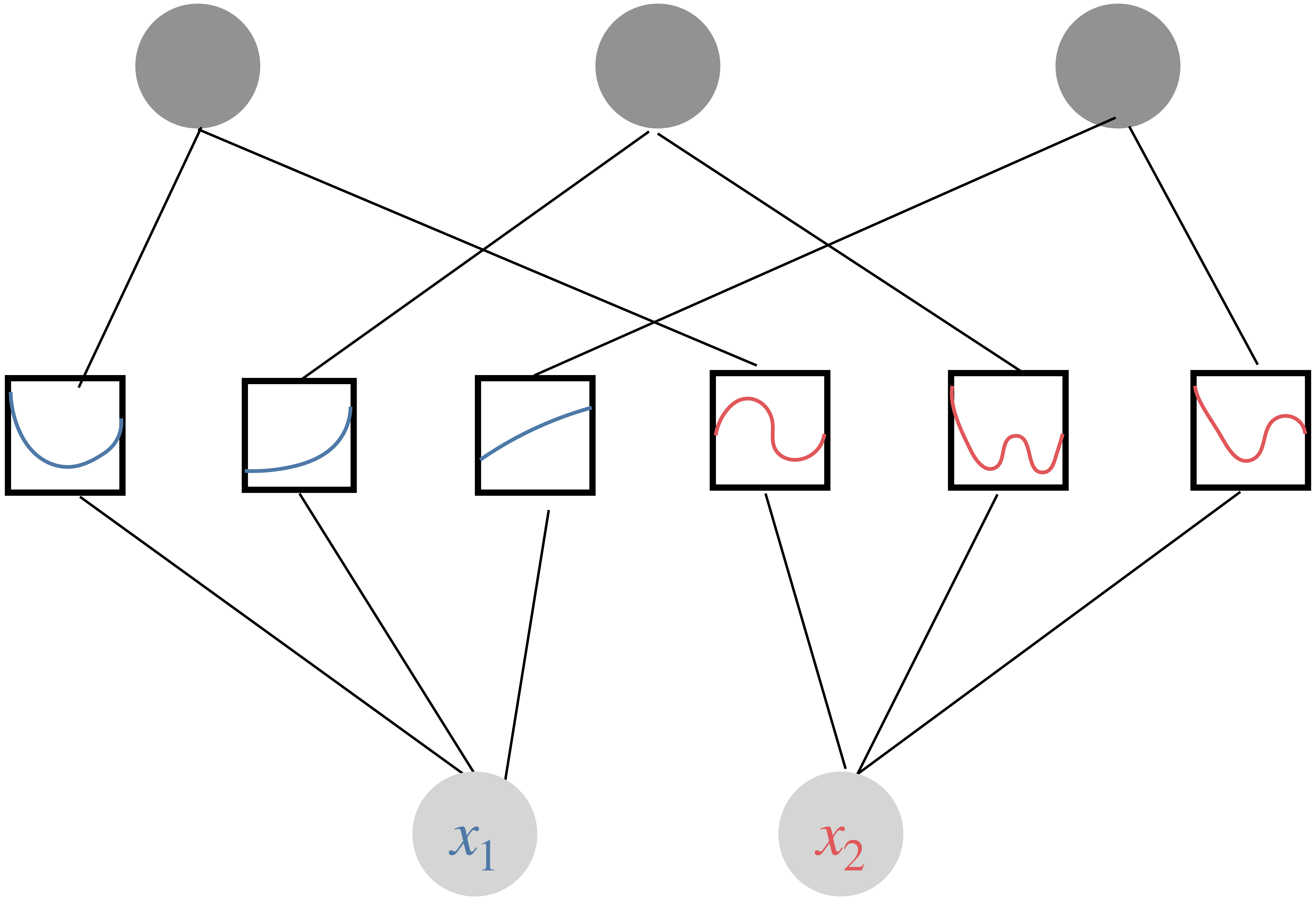
fixed nonlinear functions on nodes
x_1
x_2
x_3
learnable linear weights on edges
x_1
x_2
x_1
x_2
x_3
x_1
x_2
learnable nonlinear functions on edges



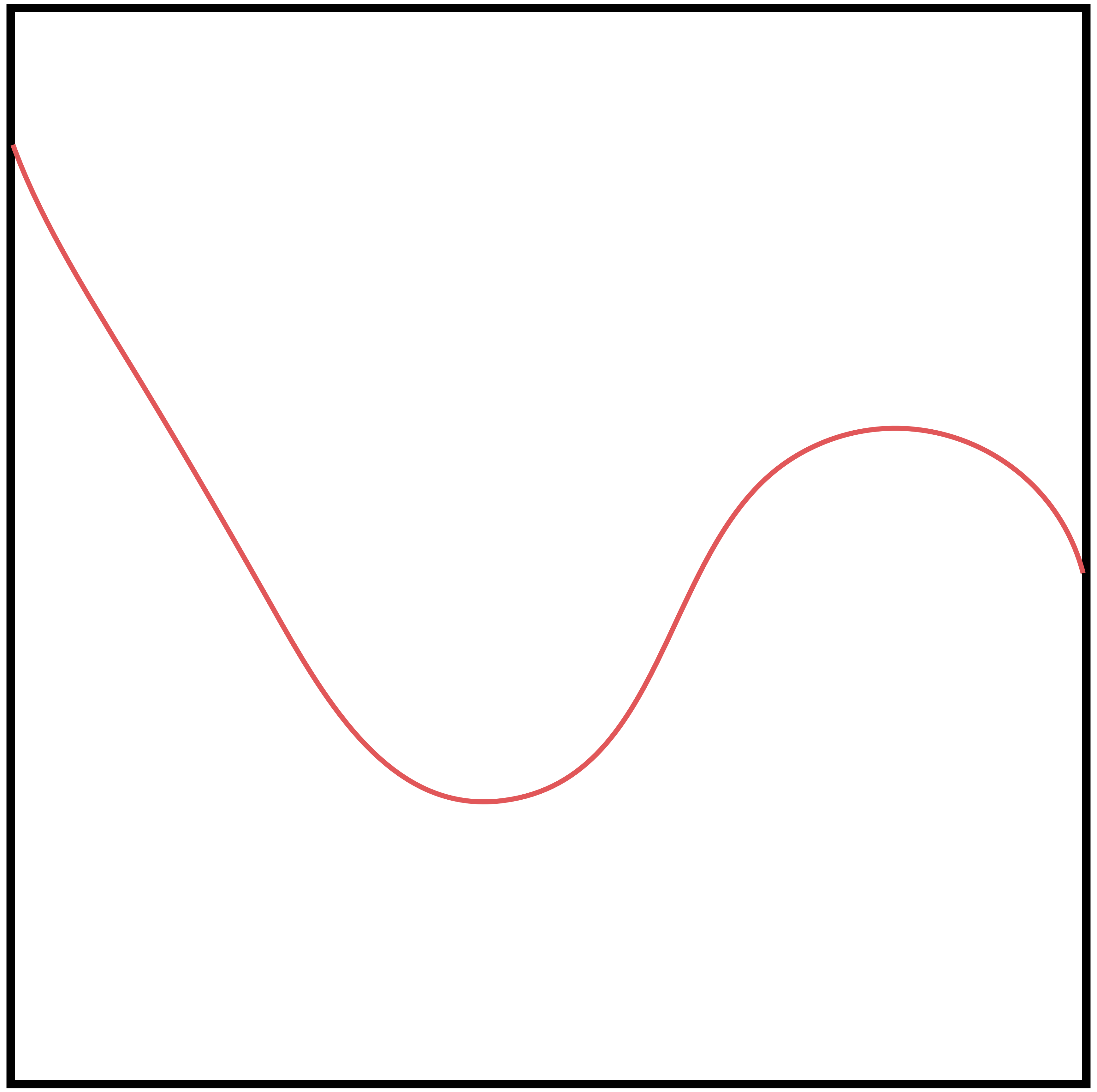
x_1
x_2



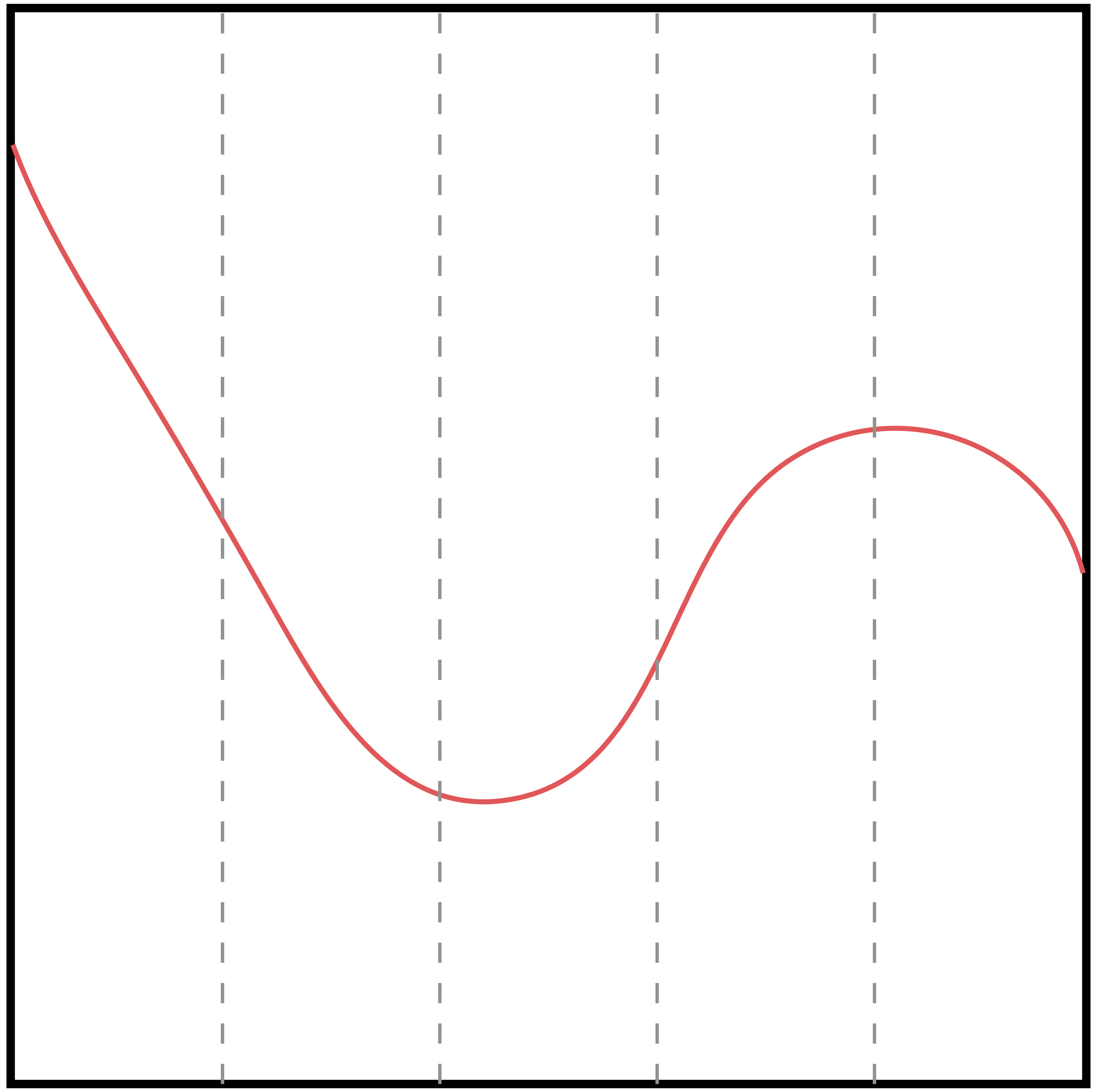
x_1
x_2


x_1
x_2

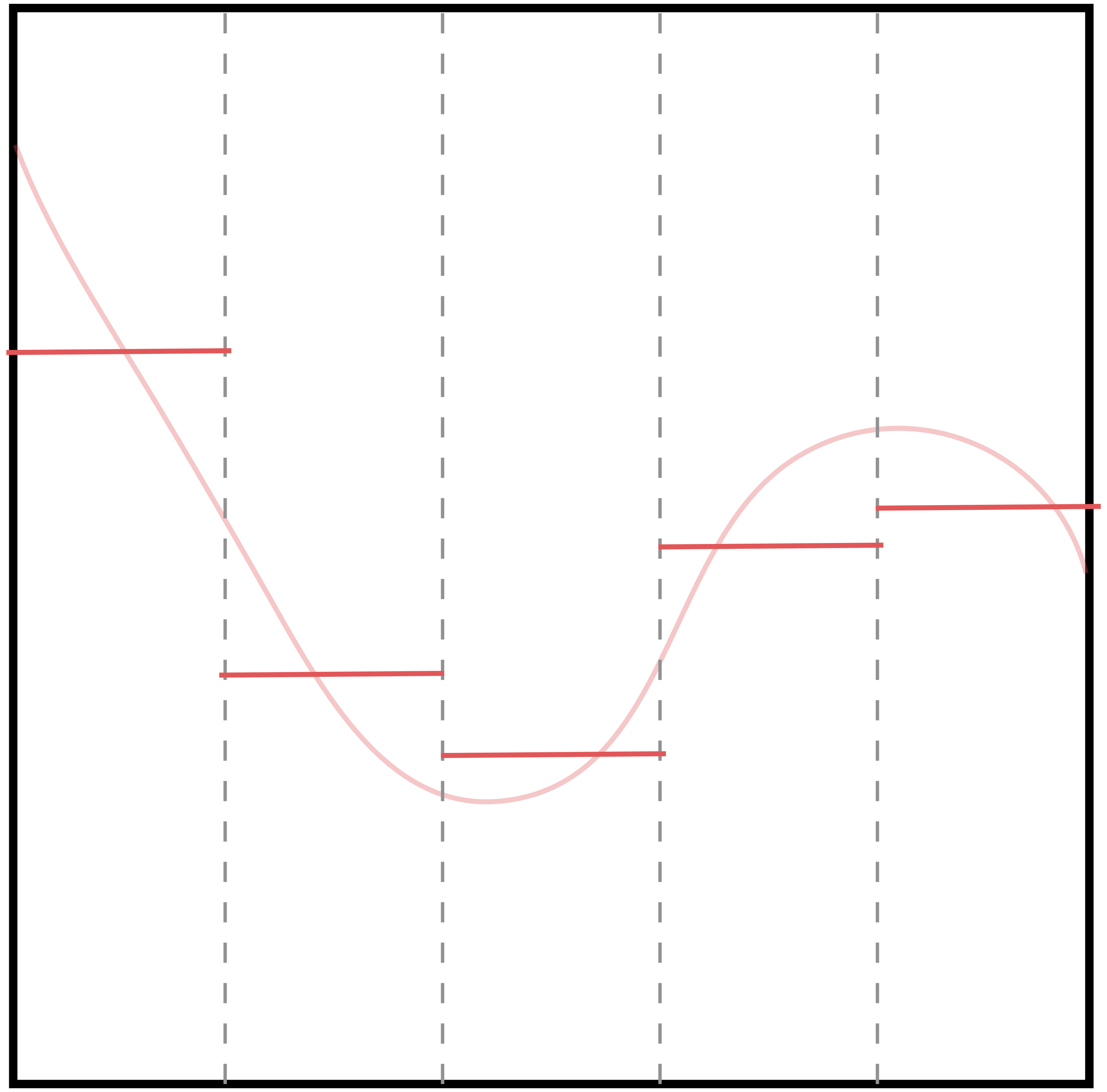
B-Splines



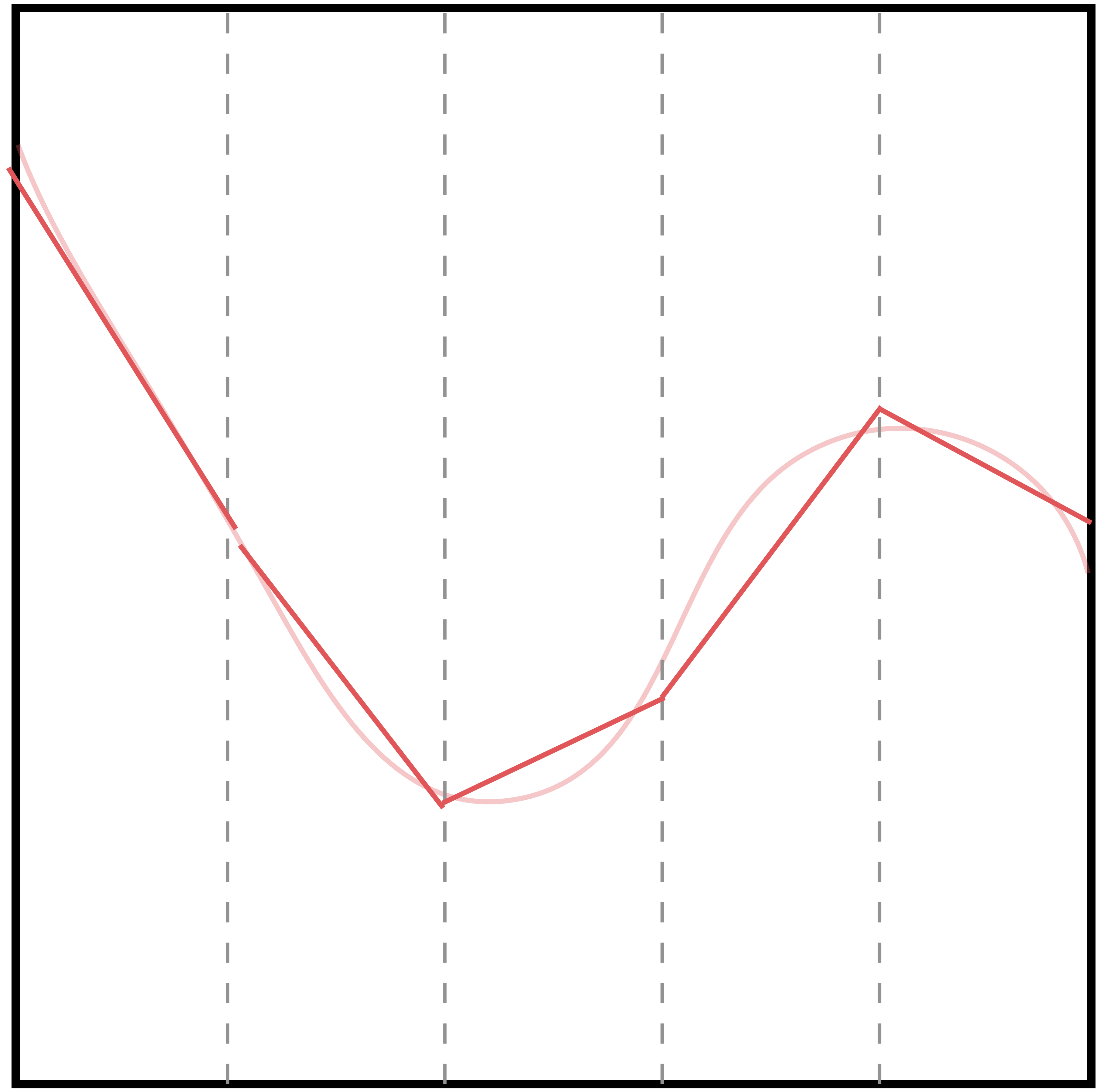
B-Splines



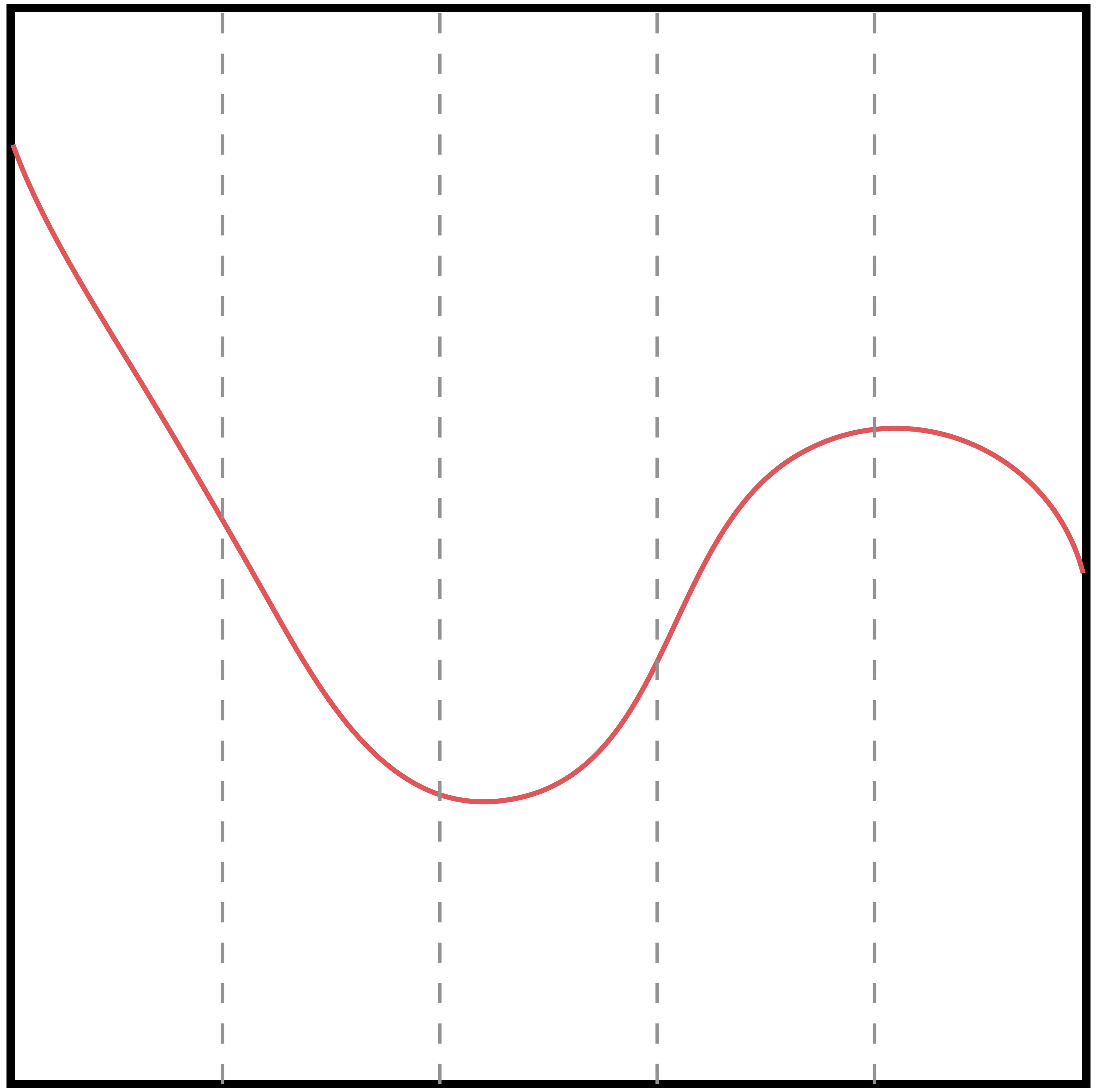
Constant B-Splines



Linear B-Splines

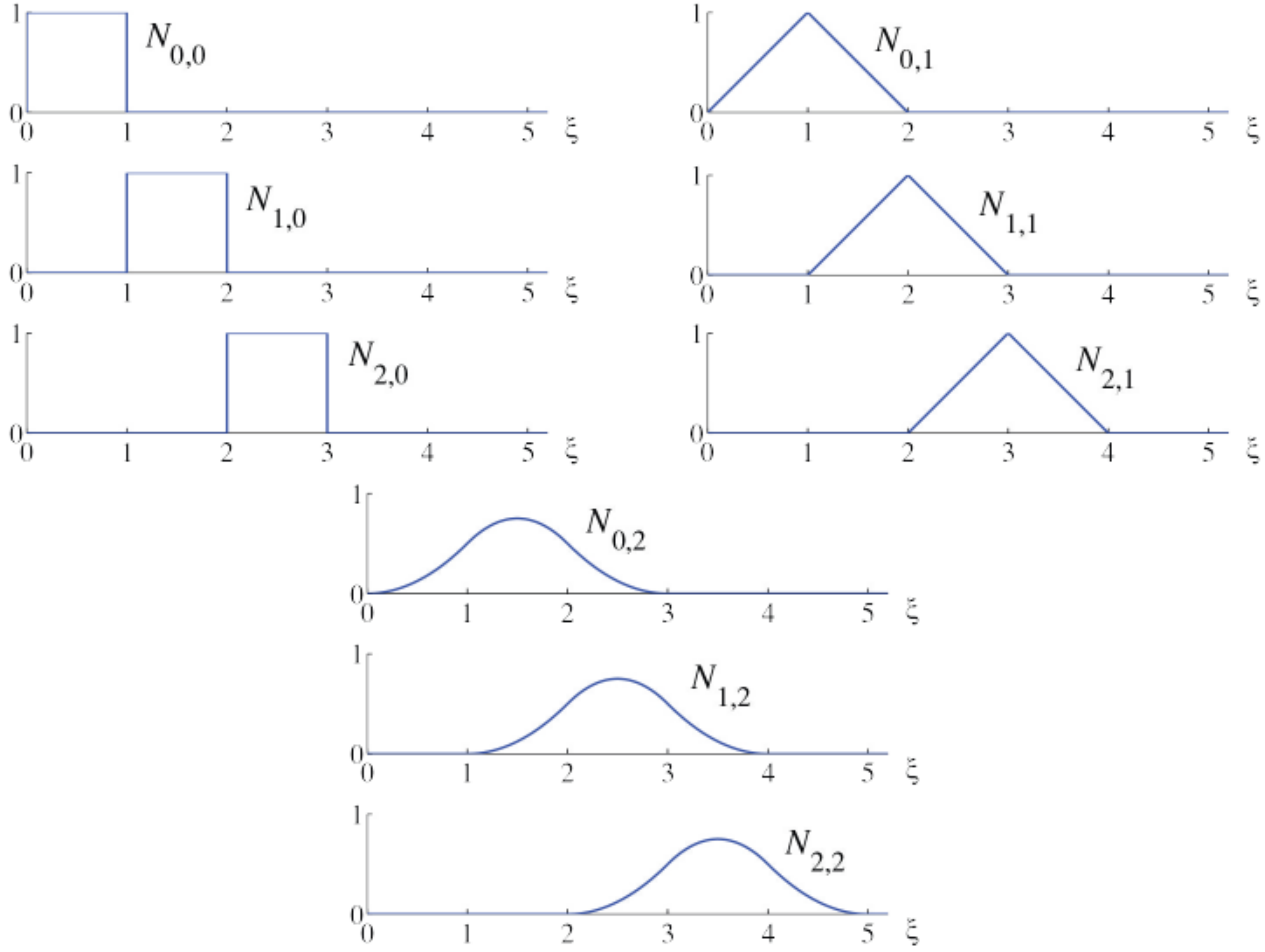


Cubic B-Splines



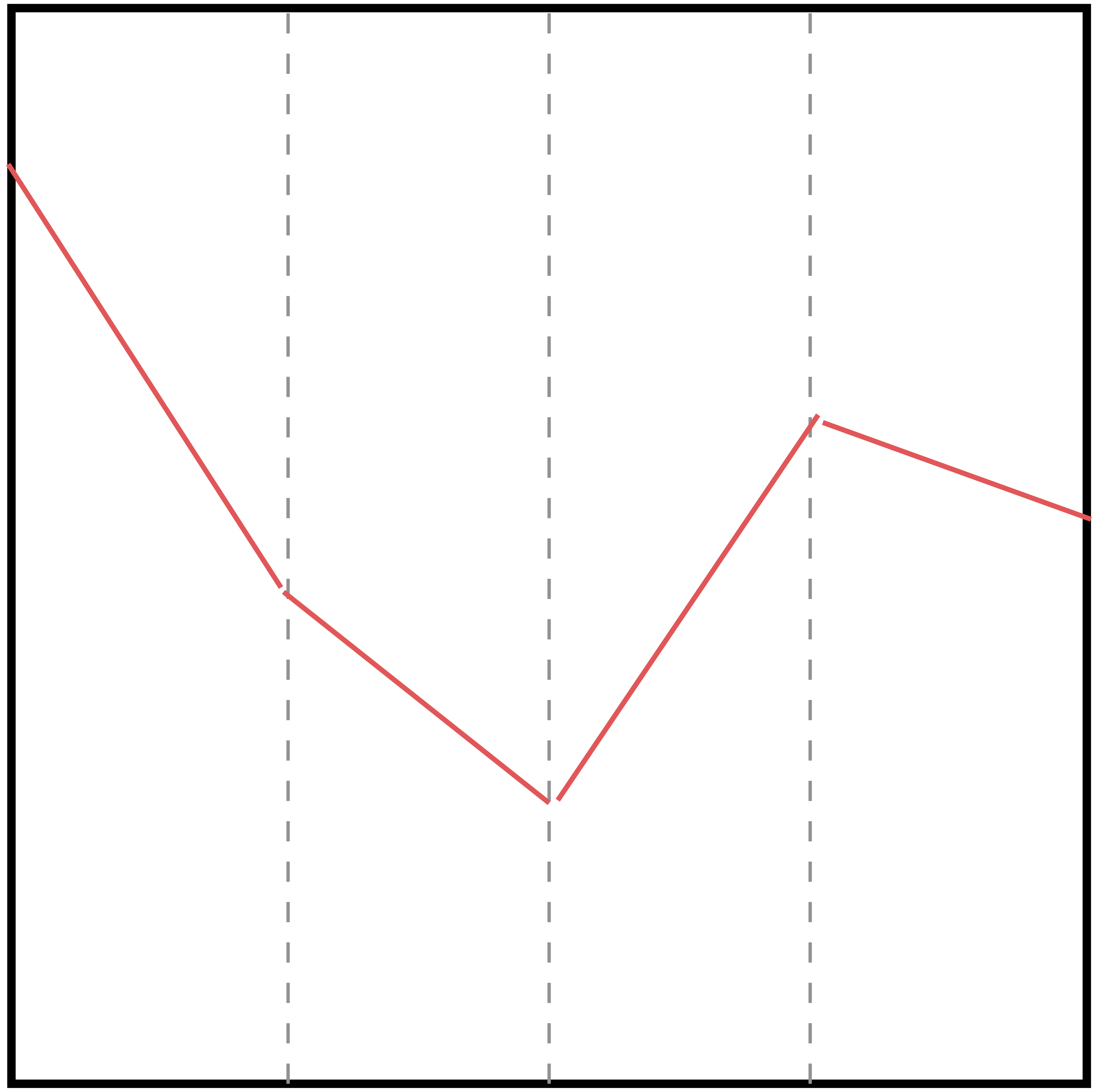
B-splines

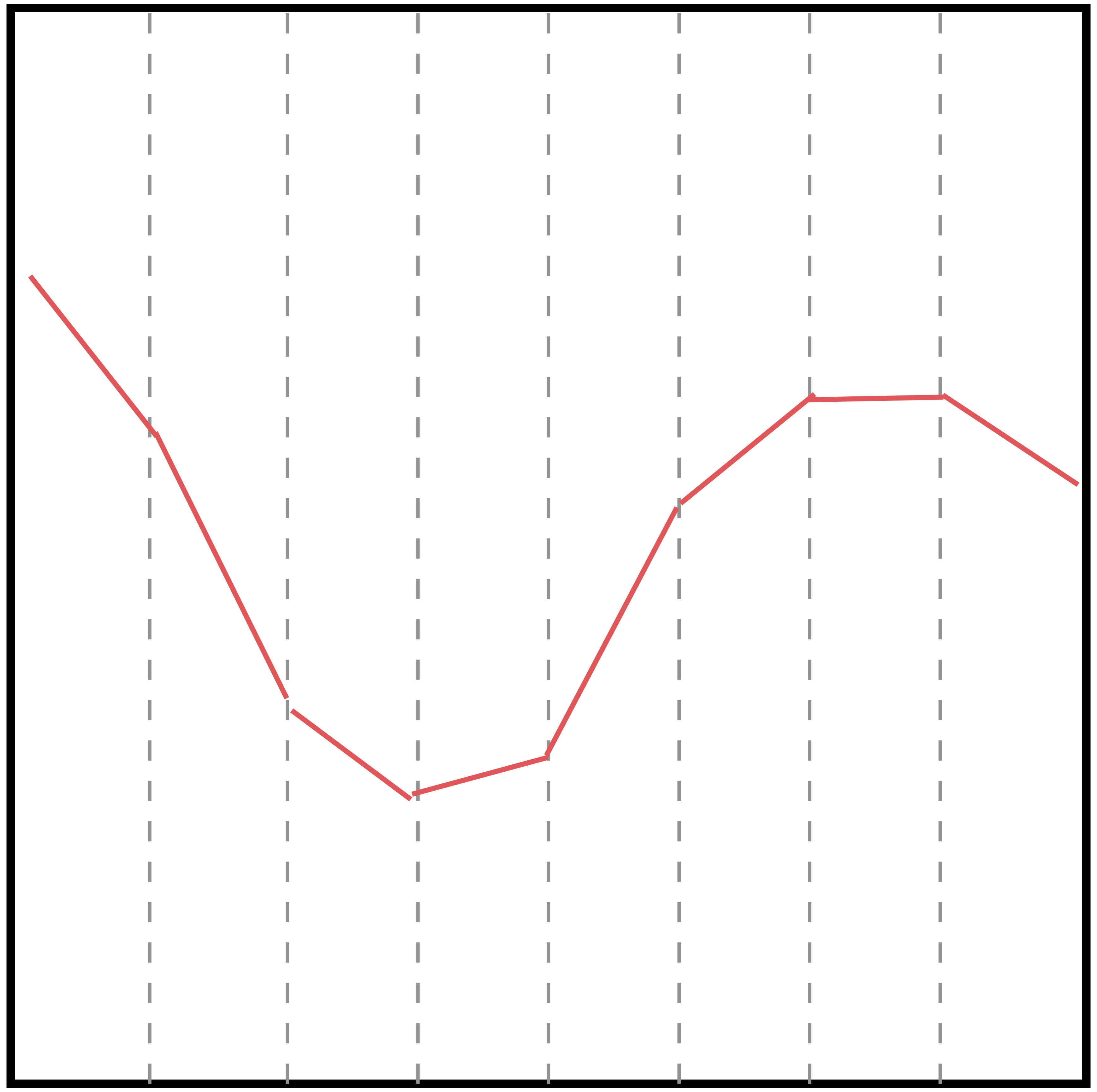
$$\text{spline}(x) = \sum_i c_i B_i(x)$$



Ruben van Nieuwpoort

Grid Extension



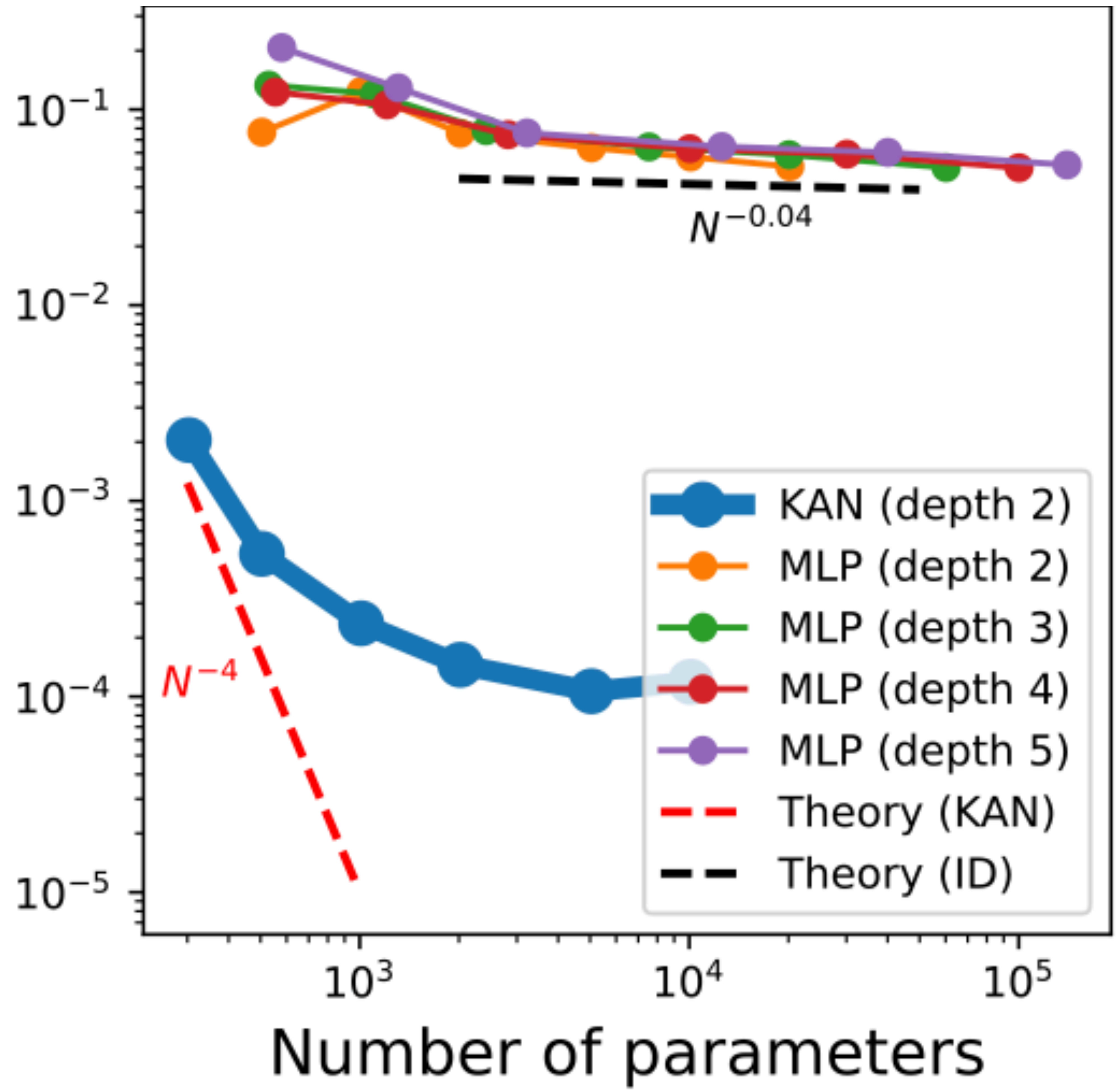


Example of benefits of KANs

$$f(x_1, \cdots, x_N) = \exp\left(\frac{1}{N}\sum_{i=1}^{N}\sin^2(x_i)\right),$$

$$f(x_1, \cdots, x_N) = \exp\left(\frac{1}{N}\sum_{i=1}^{N}\sin^2(x_i)\right),$$

RMSE



Encouraging Sparsity

L1 loss

$$|\Phi|_1 = \sum_i \sum_j |\phi_{i,j}|_1$$

$$|\phi|_1 = \frac{1}{n_s} \sum_{n_s} |\phi(x^{(i)})|$$

sum over all samples

Encouraging Sparsity

$$\ell_{\text{total}} = \ell_{\text{pred}} + \lambda \left(\mu_1 \sum_{l=0}^{L-1} |\mathbf{\Phi}_l|_1 + \mu_2 \sum_{l=0}^{L-1} S(\mathbf{\Phi}_l) \right),$$

L1 loss

$$|\Phi|_1 = \sum_i \sum_j |\phi_{i,j}|_1$$

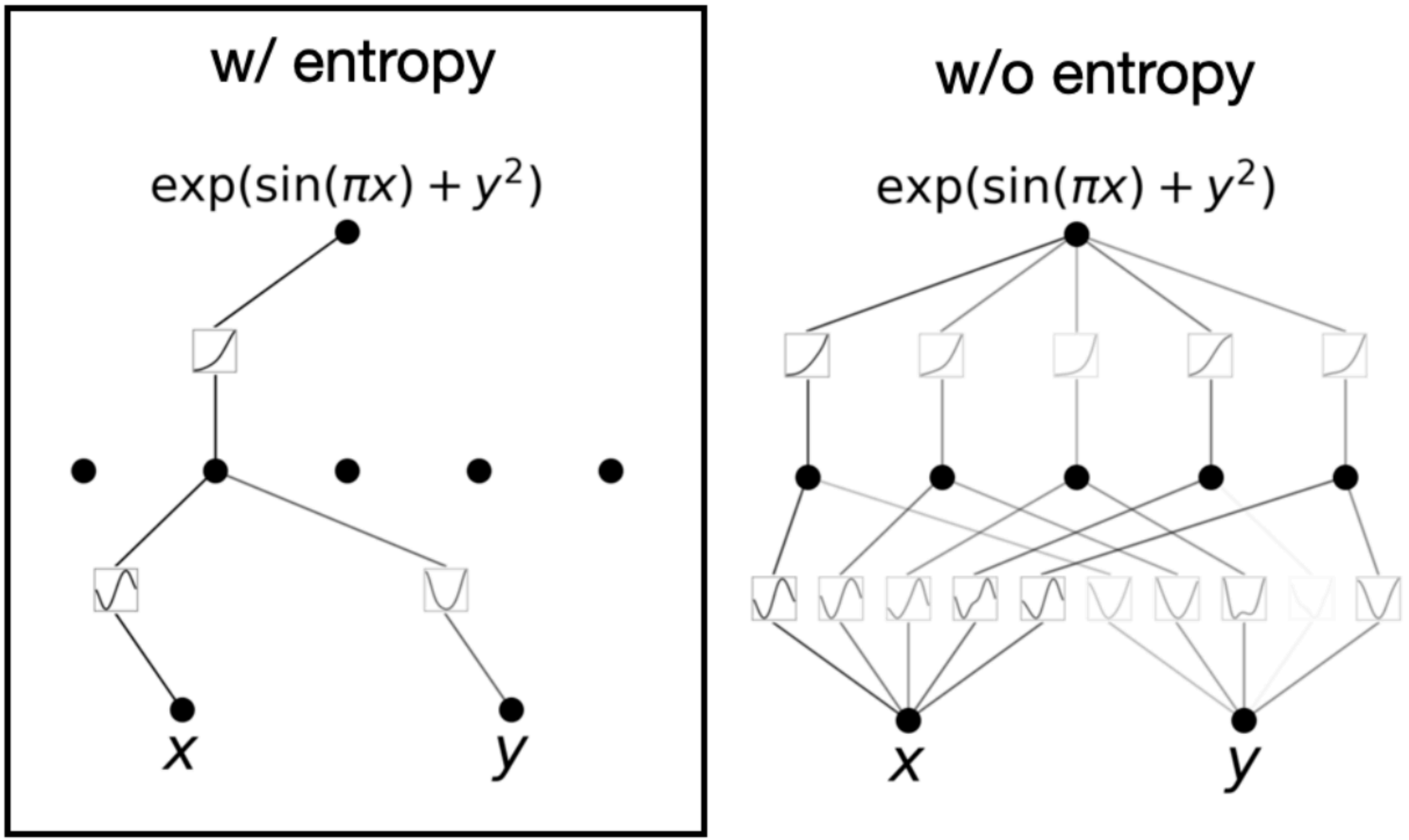
$$|\phi|_1 = \frac{1}{n_s} \sum_{n_s} |\phi(x^{(i)})|$$

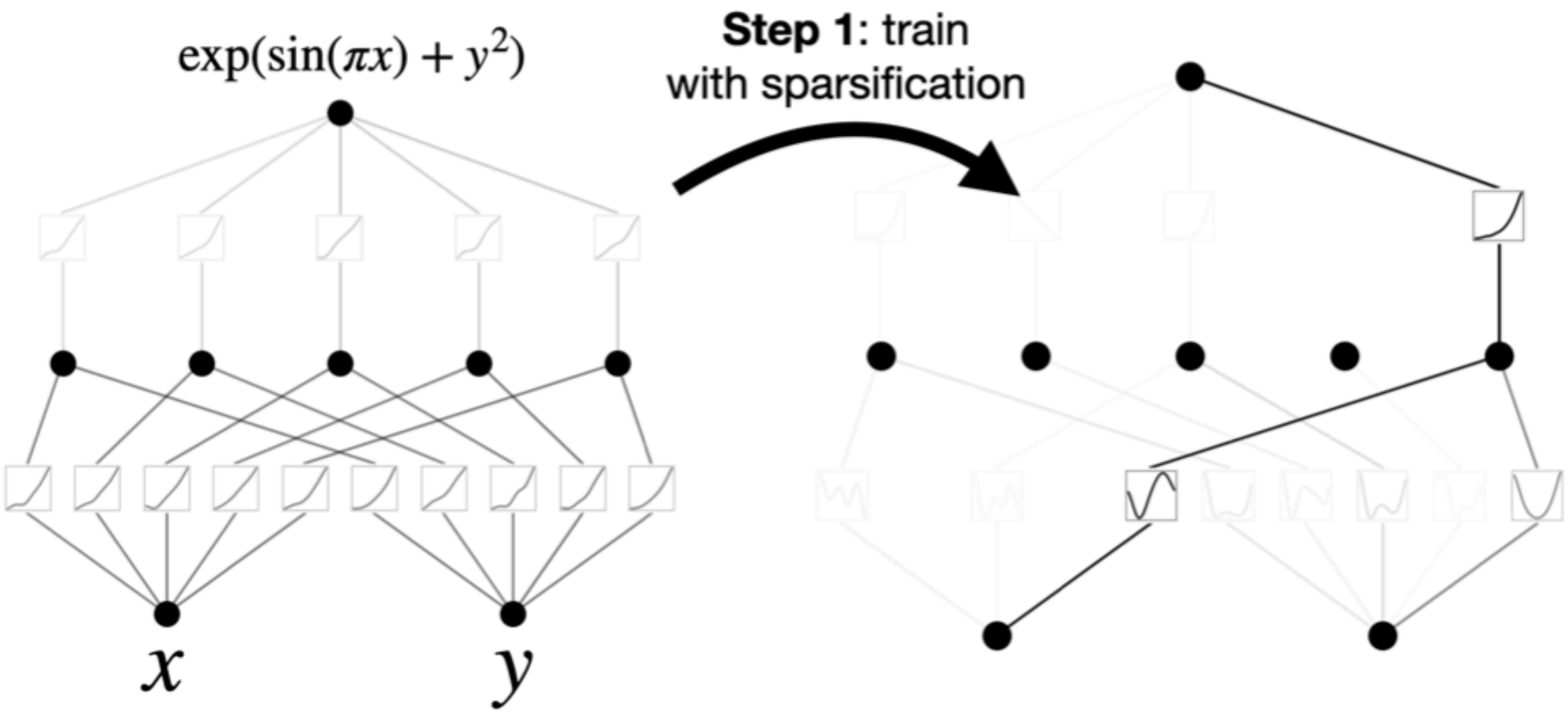
sum over all samples

Entropy

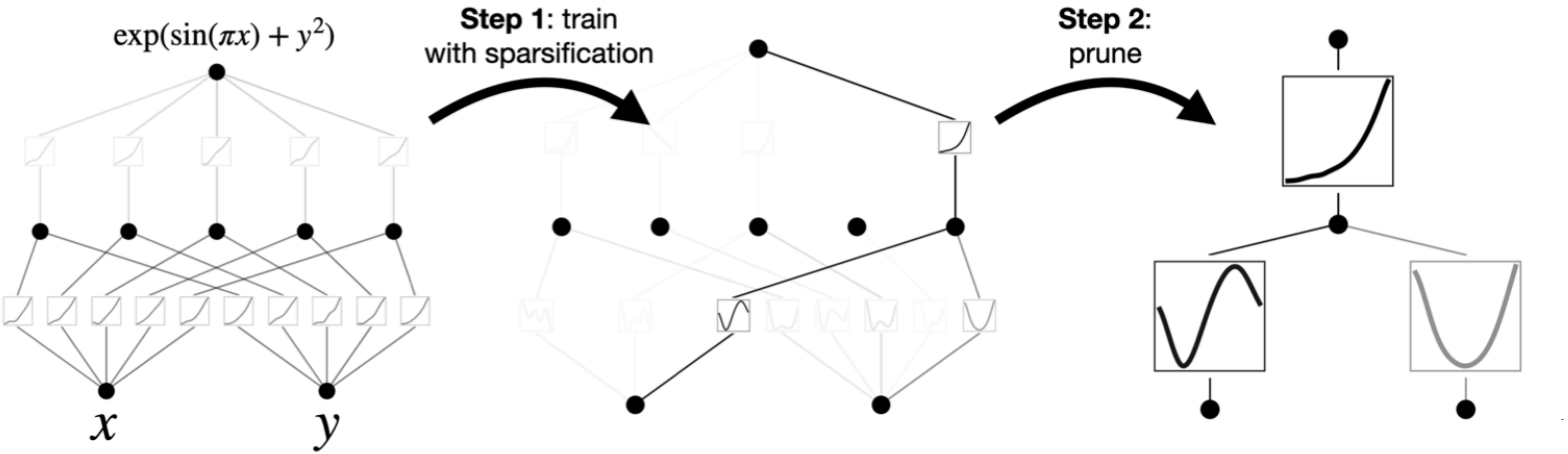
$$S(\mathbf{\Phi}) \equiv -\sum_{i=1}^{n_{\text{in}}} \sum_{j=1}^{n_{\text{out}}} \frac{|\phi_{i,j}|_1}{|\mathbf{\Phi}|_1} \log \left(\frac{|\phi_{i,j}|_1}{|\mathbf{\Phi}|_1} \right).$$

(a) Effect of entropy regularization

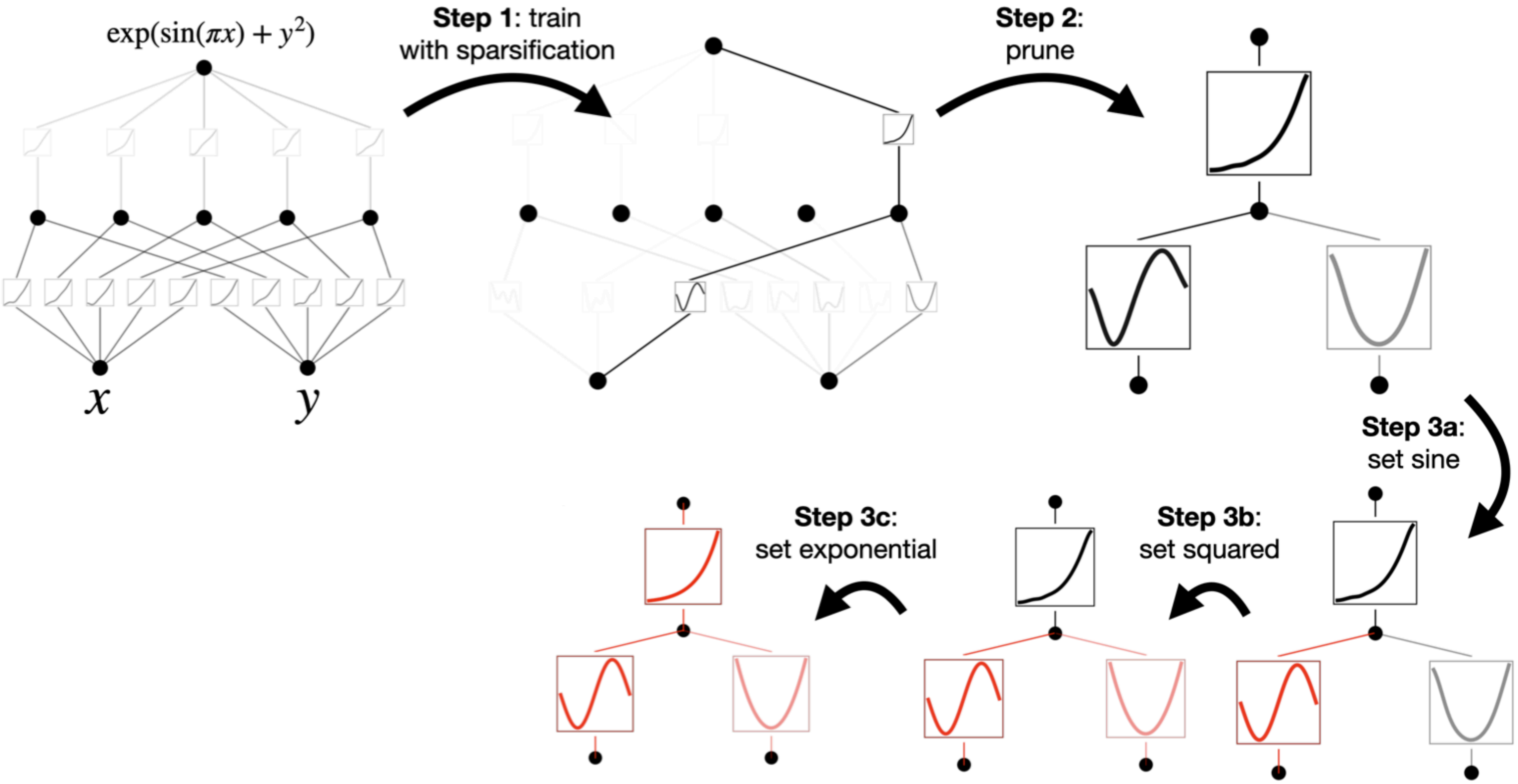




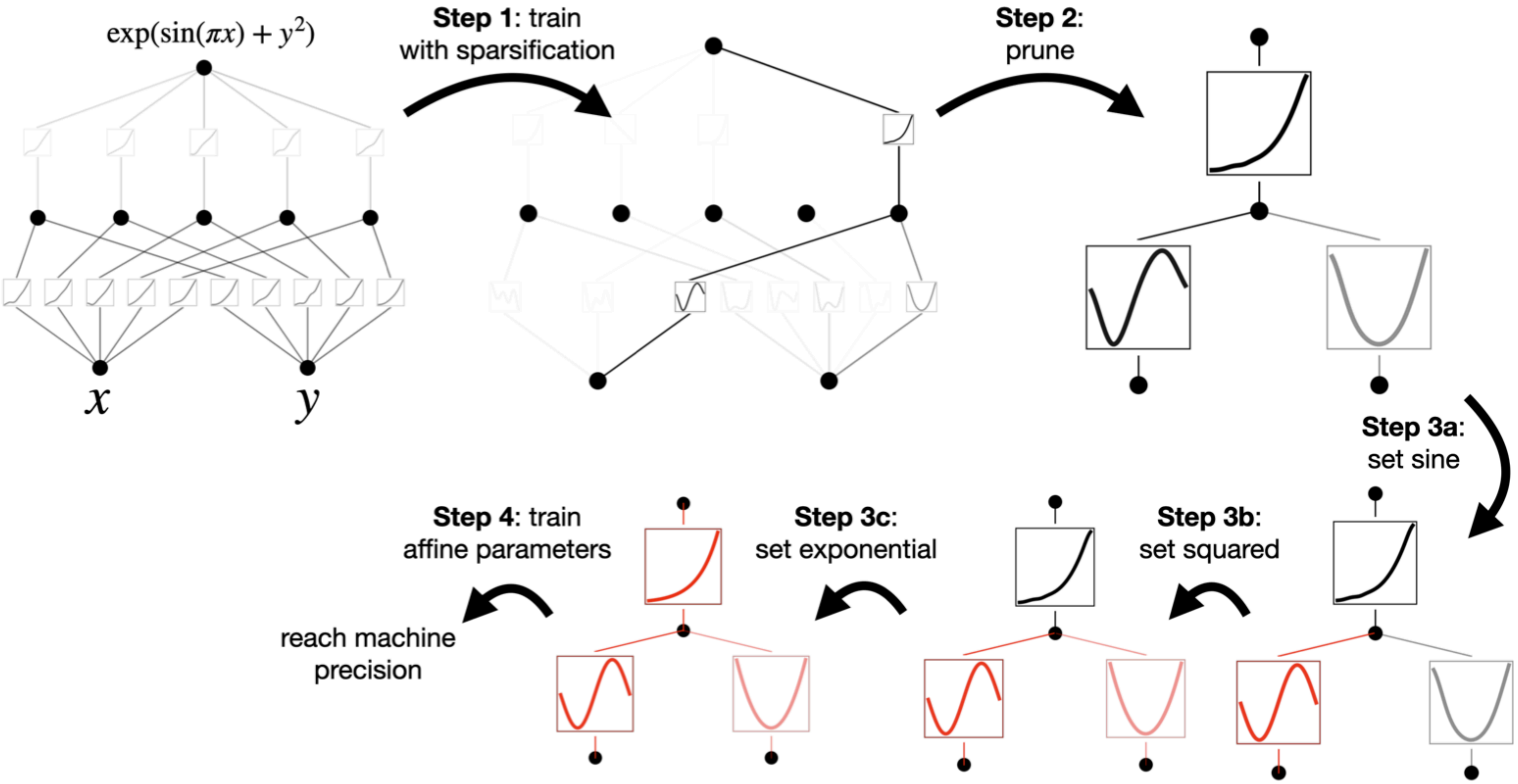
exp(sin(πx) + y²)
Step 1: train
with sparsification
x
y



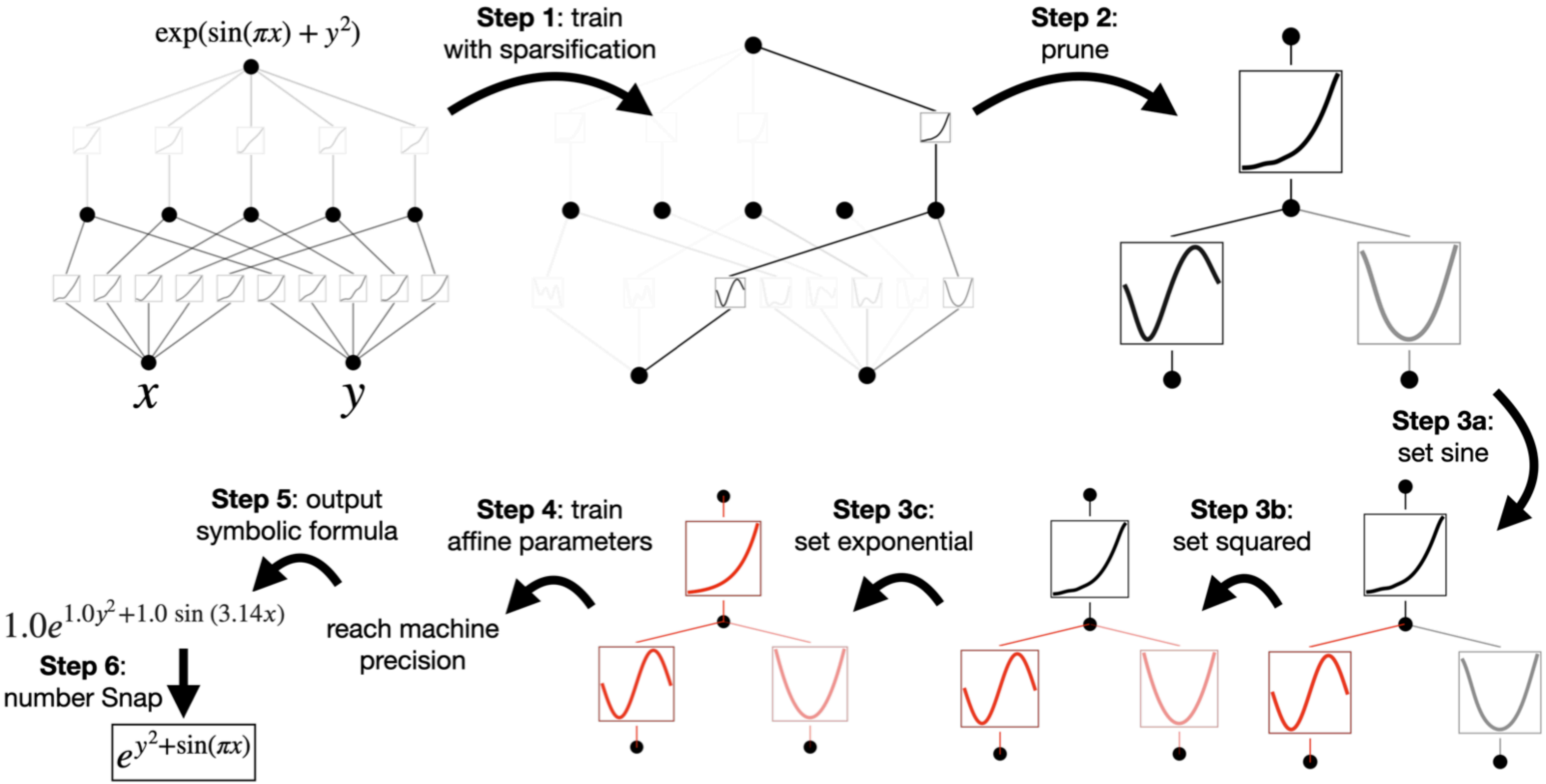
exp(sin(πx) + y²)
x
y
Step 1: train
with sparsification
Step 2:
prune



exp(sin(πx) + y²)
x
y
Step 1: train
with sparsification
Step 2:
prune
Step 3a:
set sine
Step 3b:
set squared
Step 3c:
set exponential

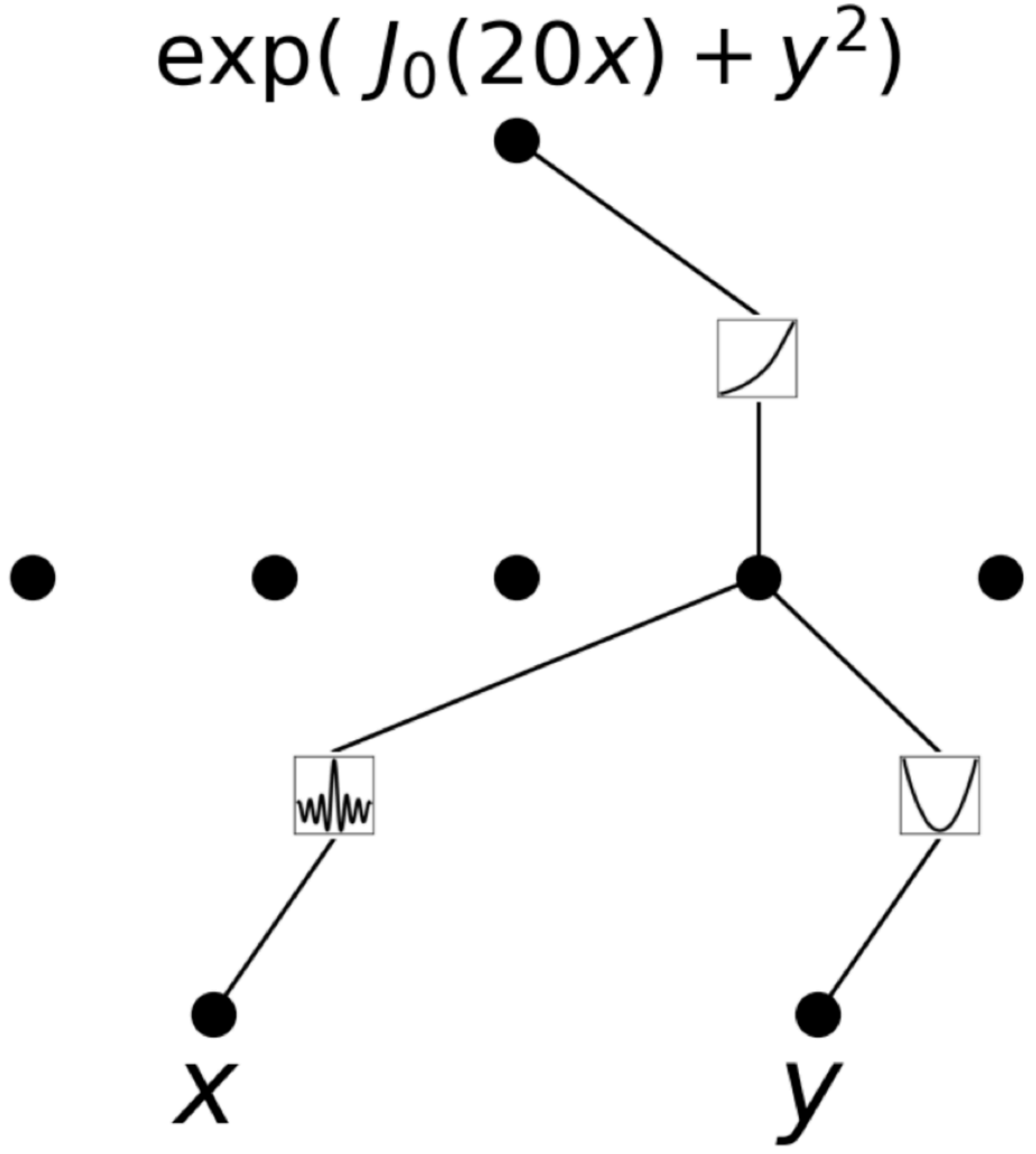


$\exp(\sin(\pi x) + y^2)$
x
y
Step 1: train
with sparsification
Step 2:
prune
Step 3a:
set sine
Step 3b:
set squared
Step 3c:
set exponential
Step 4: train
affine parameters
reach machine
precision

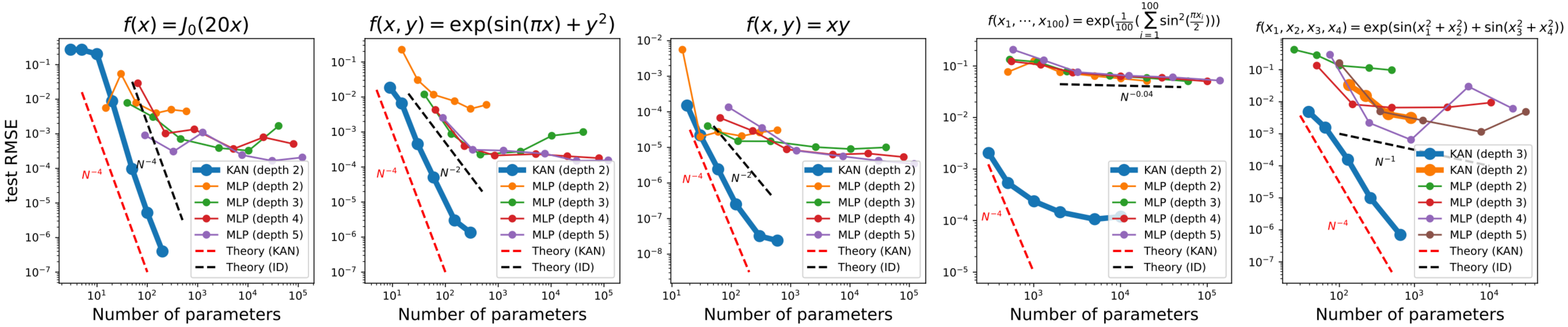


$\exp(\sin(\pi x) + y^2)$
x
y
Step 1: train
with sparsification
Step 2:
prune
Step 3a:
set sine
Step 3b:
set squared
Step 3c:
set exponential
Step 4: train
affine parameters
reach machine
precision
Step 5: output
symbolic formula
$1.0e^{1.0y^2+1.0\sin(3.14x)}$
Step 6:
number Snap
$e^{y^2+\sin(\pi x)}$

Advantages over Symbolic Regression



Accuracy compared to MLP



KAN compared to MLP

advantage: can facilitate interpretability, i.e., symbolic expressions

might achieve higher accuracy in some cases

trains slower

submitted Aug 2024

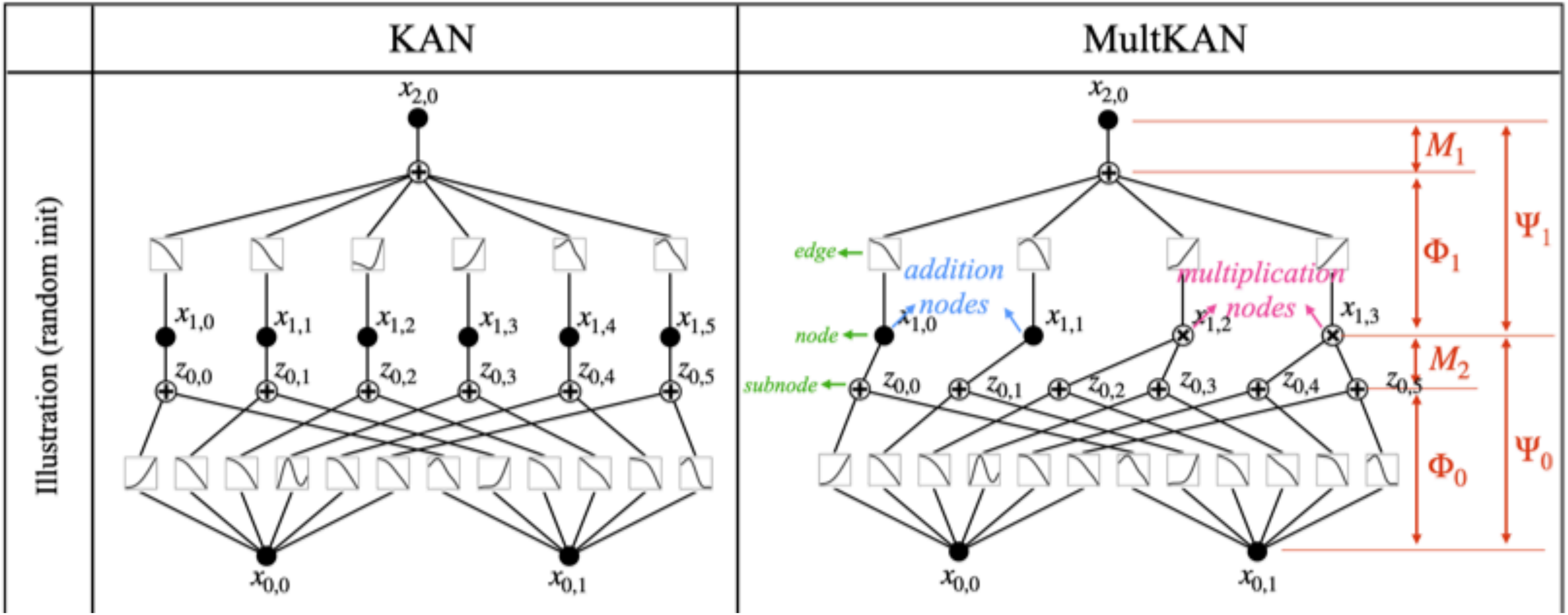
KAN 2.0: Kolmogorov-Arnold Networks Meet Science

Ziming Liu[1,4*] **Pingchuan Ma**[1,3] **Yixuan Wang**[2] **Wojciech Matusik**[1,3] **Max Tegmark**[1,4]
[1] Massachusetts Institute of Technology
[2] California Institute of Technology
[3] Computer Science and Artificial Intelligence Laboratory (CSAIL), MIT
[4] The NSF Institute for Artificial Intelligence and Fundamental Interactions

Learning other symbolic functions

$$2xy = (x+y)^2 - (x^2+y^2)$$

$$x/y = \exp(\log x - \log y)$$



$$\Psi^{(l)} = M^{(l)} \circ \Phi^{(l)}$$