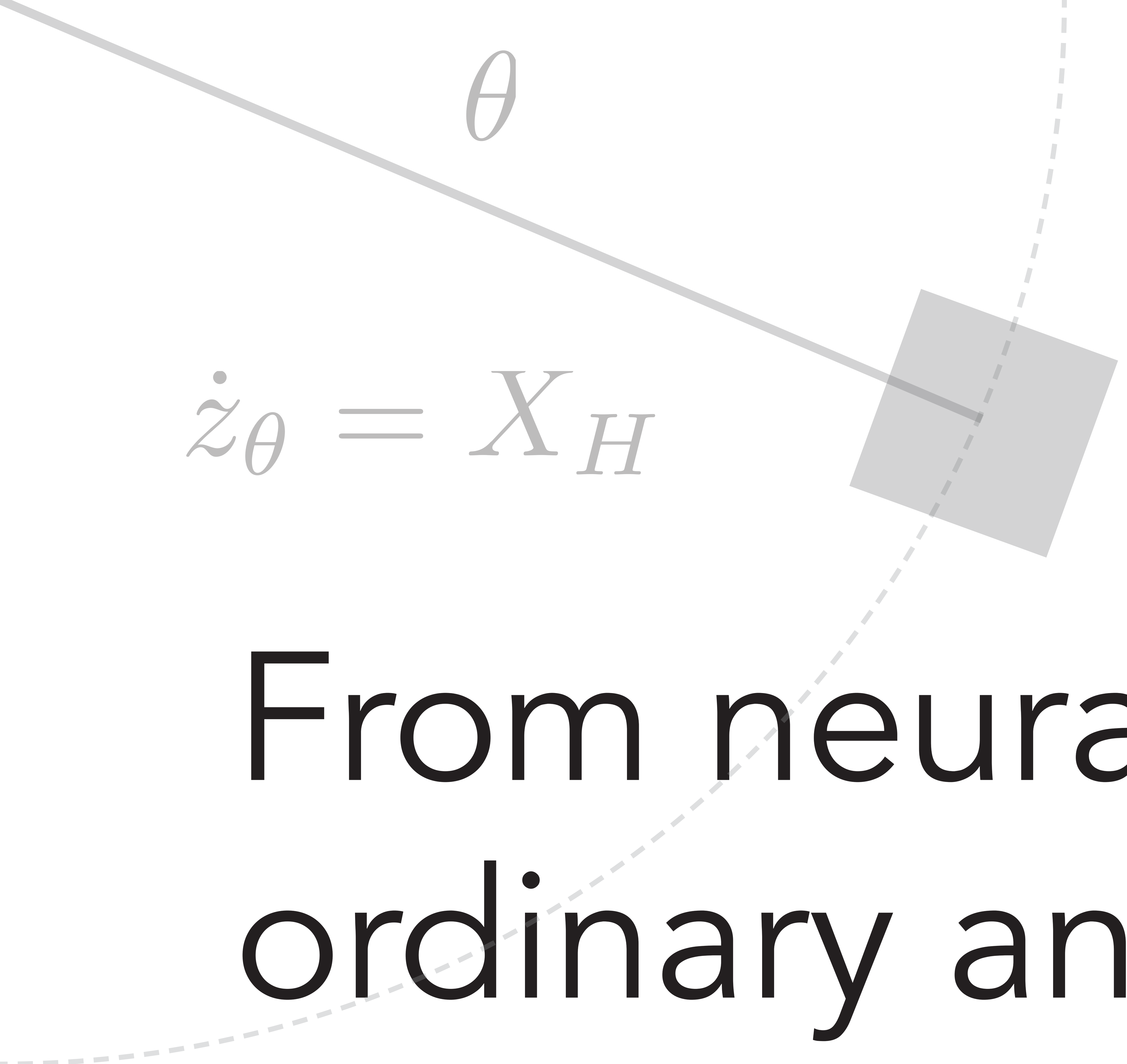
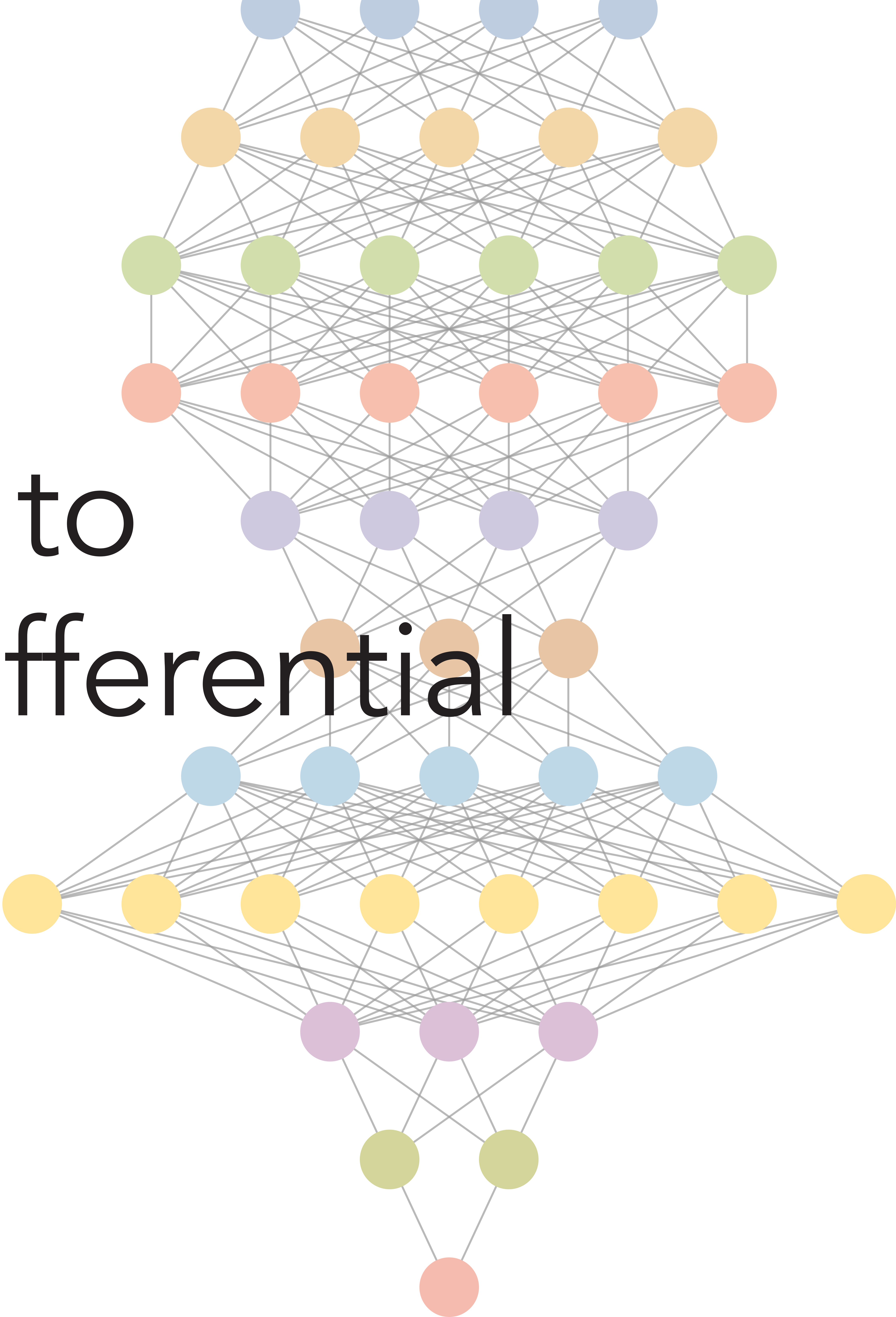


From neural networks to ordinary and partial differential equations, and back



Thomas Richter and Christian Lessig
Otto-von-Guericke-Universität Magdeburg

Neural networks and differential equations

- Use (applied) mathematics to improve understanding of and computations with neural networks
- Use neural network to improve simulation of differential equations (and in general the solution of problems in numerical analysis)

What is a neural network?

What is a neural network?

- Nonlinear mapping of the form

$$\mathcal{N} = L_J \circ L_{J-1} \circ \cdots \circ L_2 \circ L_1$$

What is a neural network?

- Nonlinear mapping of the form

$$\mathcal{N} = L_J \circ L_{J-1} \circ \cdots \circ L_2 \circ L_1$$

with layers

$$L_j \, : \, h_{j+1} = \sigma\big(W_j h_j + b_j\big)$$

What is a neural network?

- Nonlinear mapping of the form

$$\mathcal{N} = L_J \circ L_{J-1} \circ \cdots \circ L_2 \circ L_1$$

with layers

$$L_j \,:\, h_{j+1} = \sigma\big(W_j h_j + b_j\big) \qquad \begin{array}{l} W_j \in \mathbb{R}^{m \times n} \\ b_j \in \mathbb{R}^m \end{array}$$

What is a neural network?

- Nonlinear mapping of the form

$$\mathcal{N} = L_J \circ L_{J-1} \circ \cdots \circ L_2 \circ L_1$$

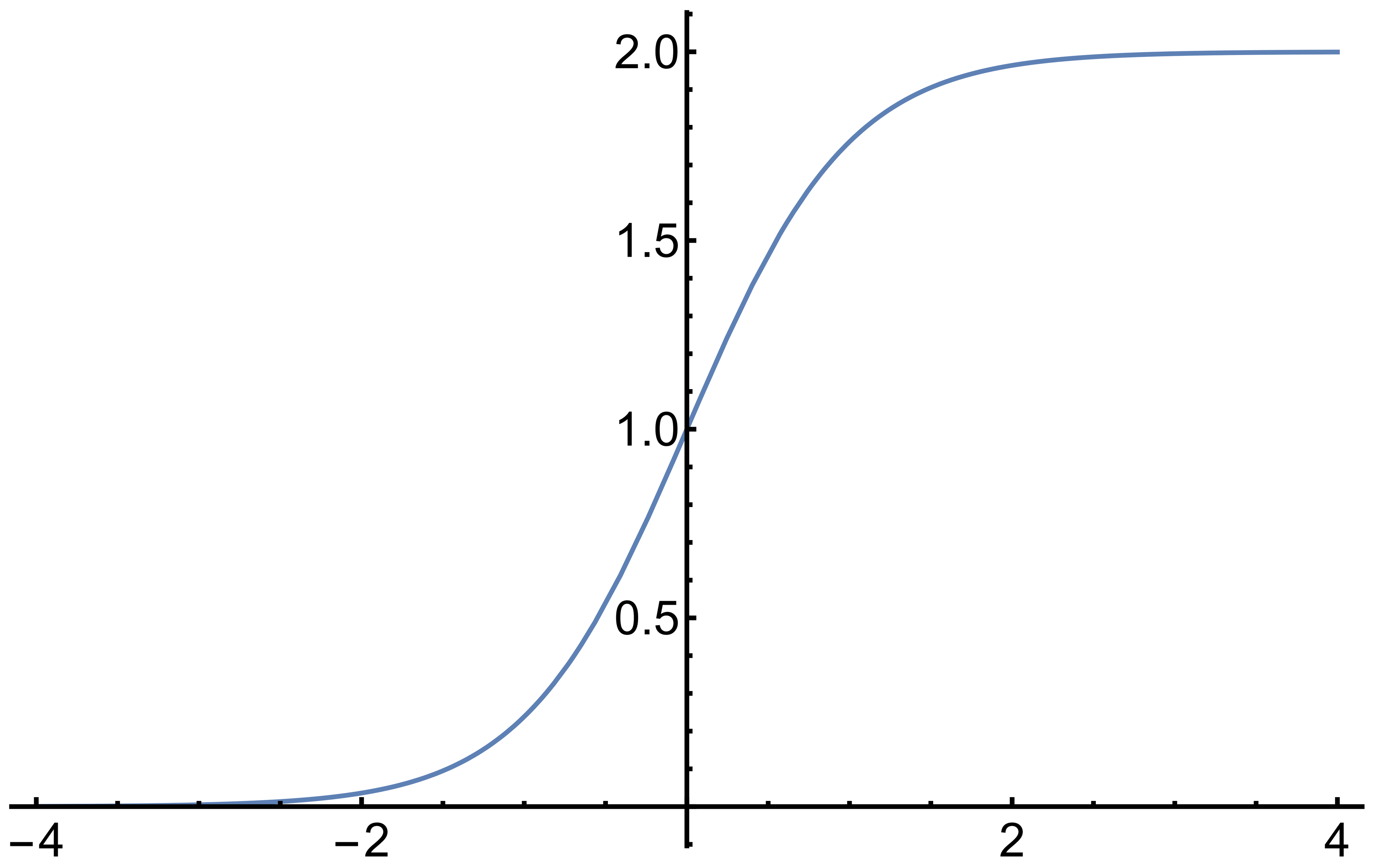
with layers

$$L_j \, : \, h_{j+1} = \sigma\big(W_j h_j + b_j\big) \qquad W_j \in \mathbb{R}^{m \times n}$$
$$b_j \in \mathbb{R}^m$$

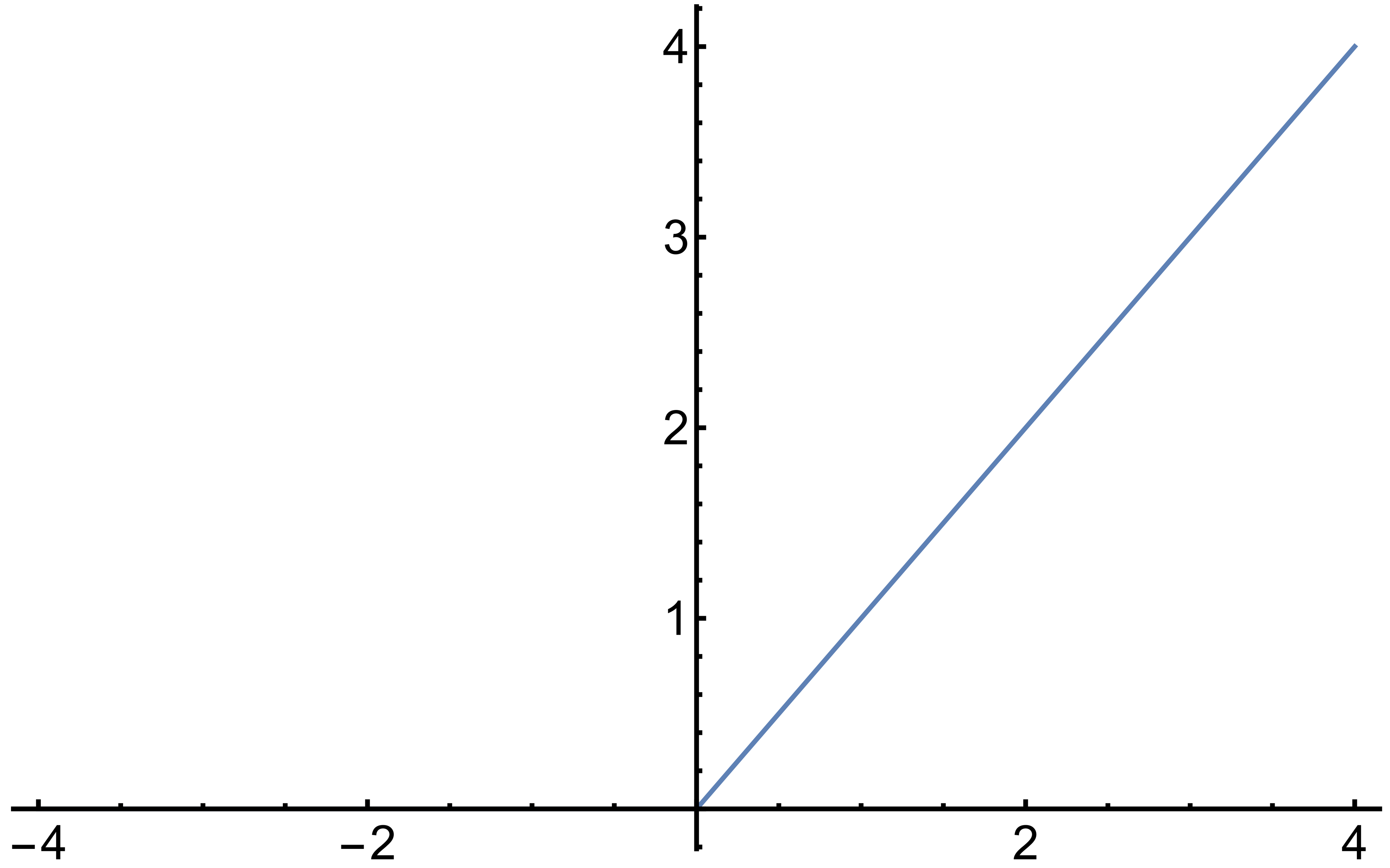
Element-wise nonlinearity,
e.g. sigmoid

What is a neural network?

- Nonlinearity



tanh



ReLU

What is a neural network?

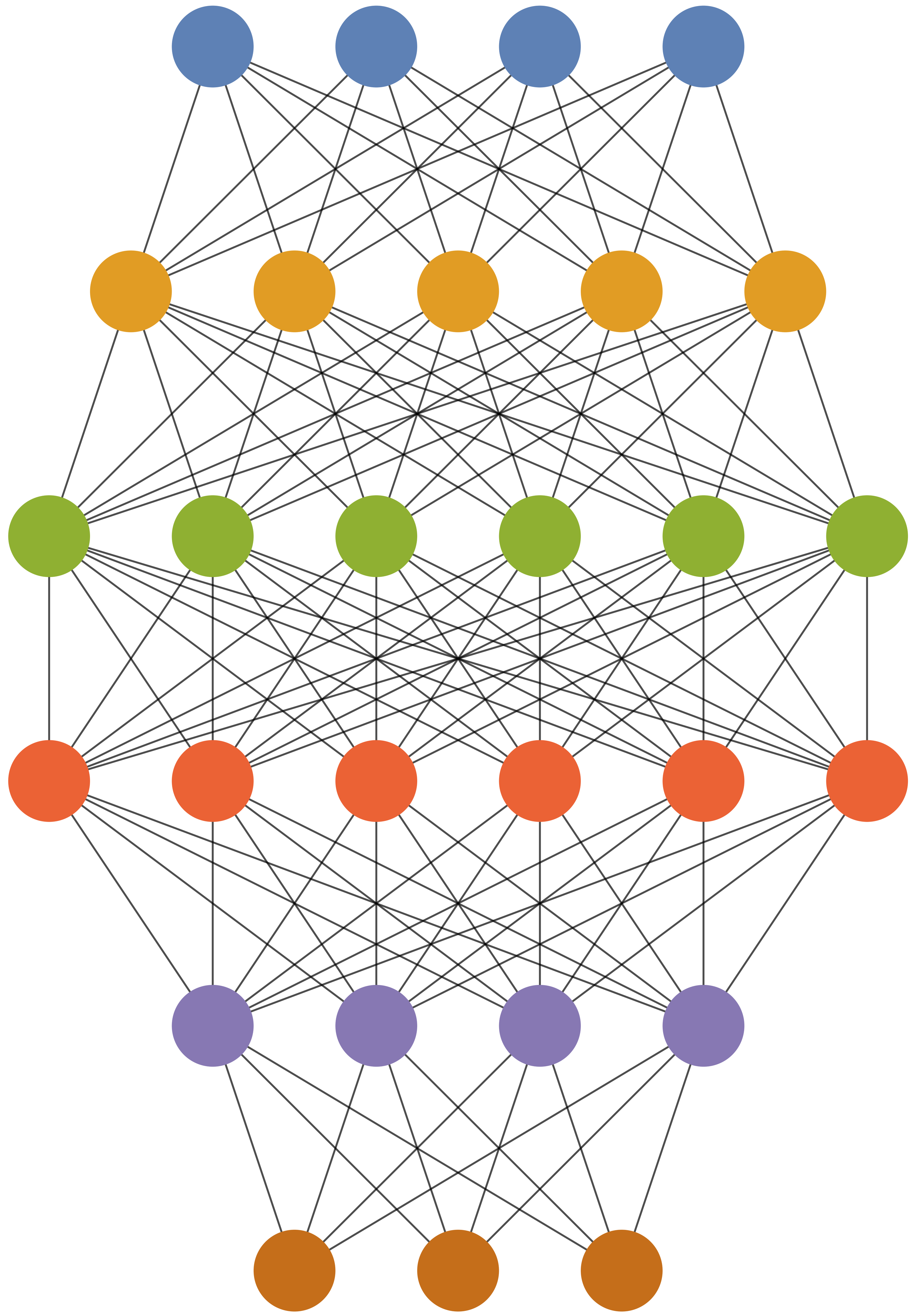
- Nonlinear mapping of the form

$$\mathcal{N} = L_J \circ L_{J-1} \circ \cdots \circ L_2 \circ L_1$$

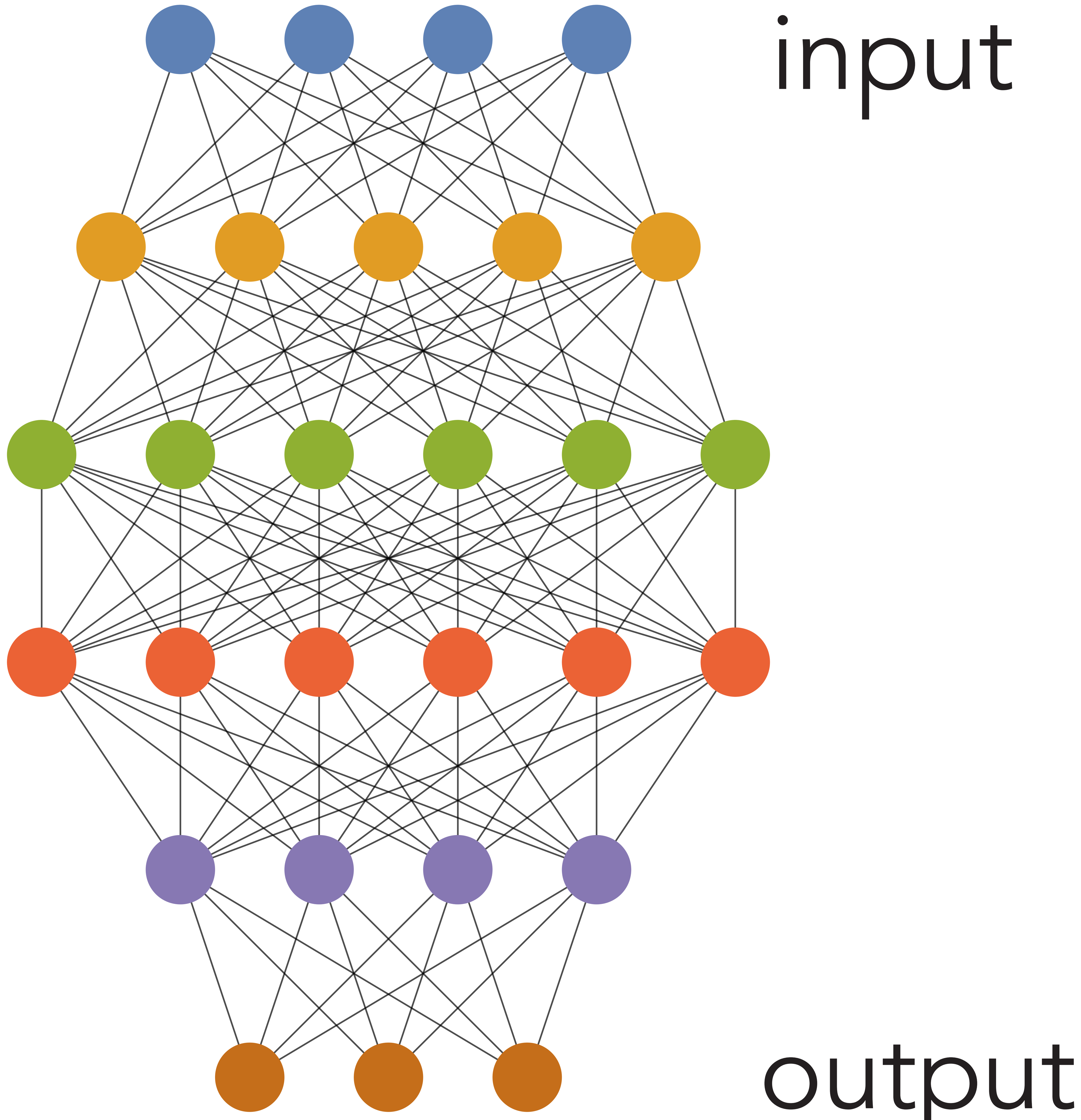
with layers

$$L_j \,:\, h_{j+1} = \sigma\big(W_j h_j + b_j\big)$$

What is a neural network?

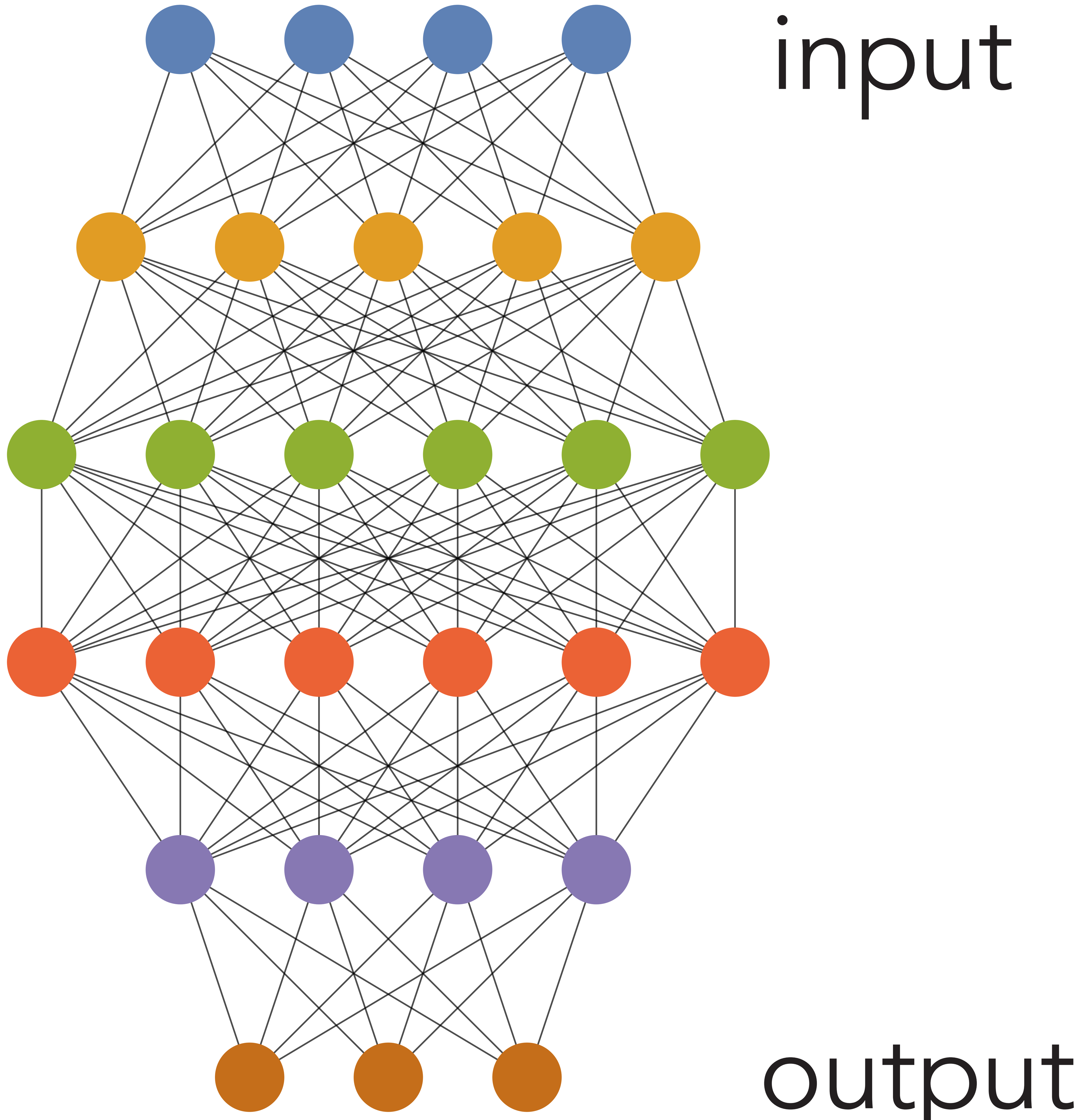


What is a neural network?



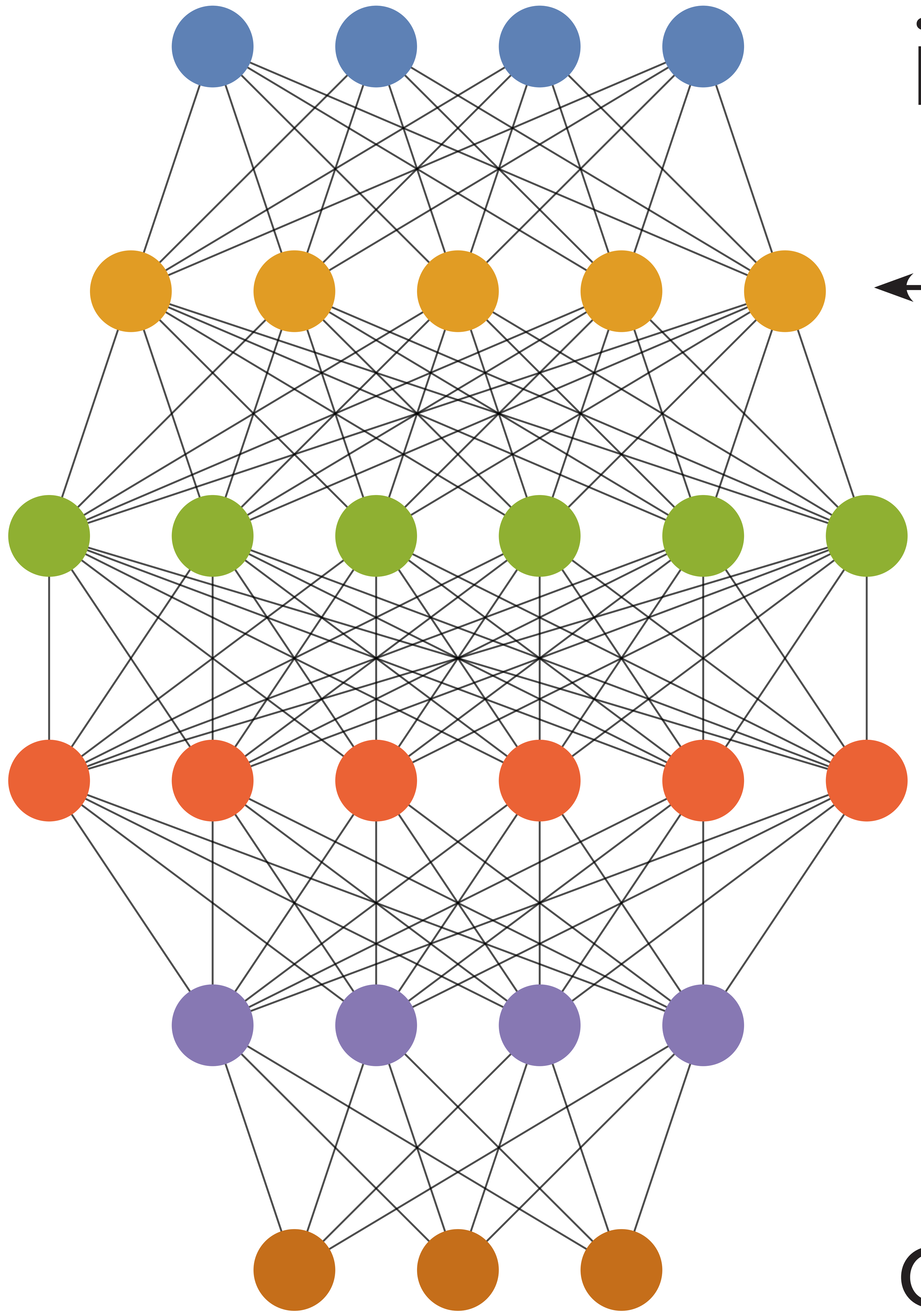
What is a neural network?

$$h_{j+1} = \sigma\left(W_j h_j + b_j\right)$$



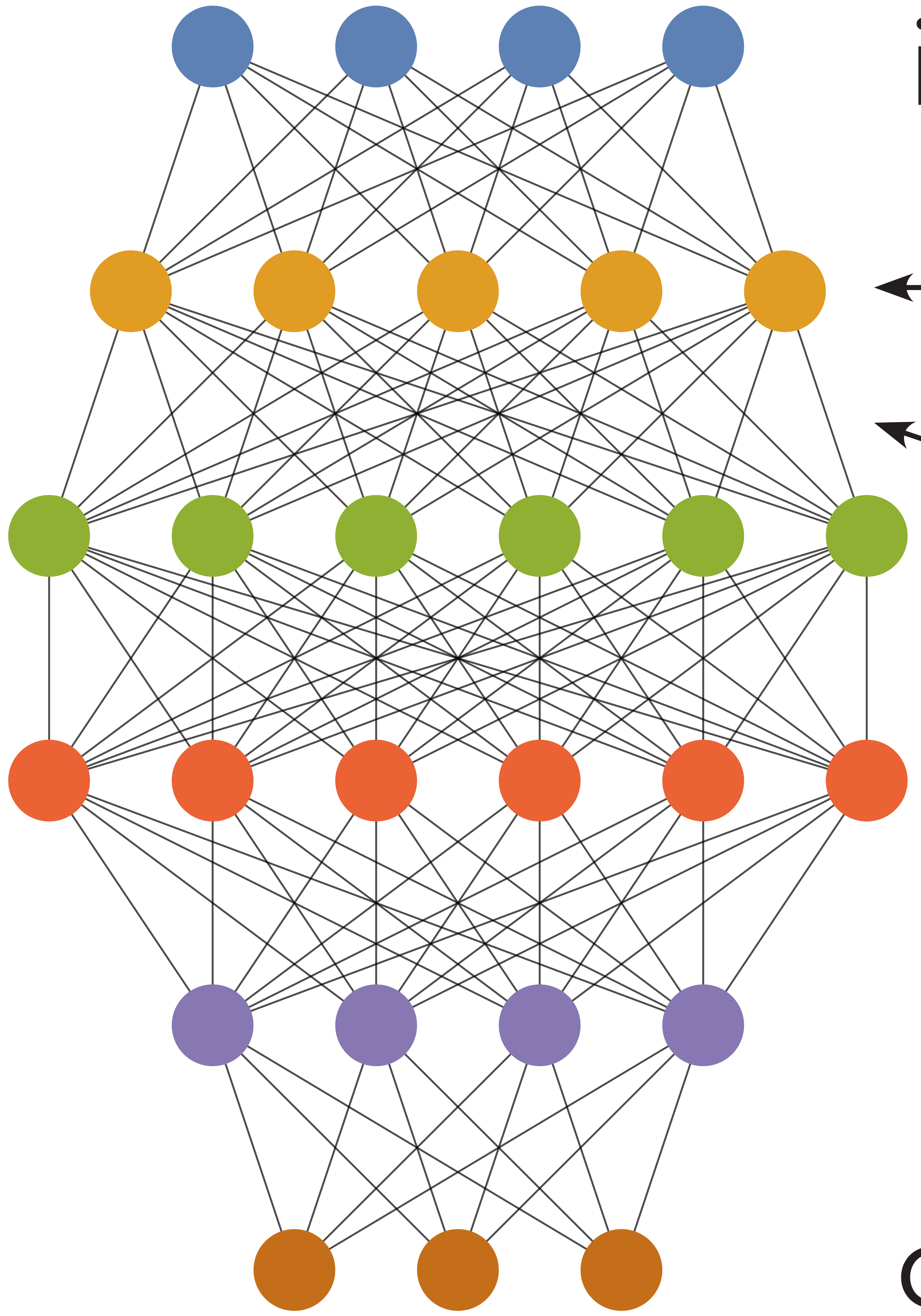
What is a neural network?

$$h_{j+1} = \sigma\left(W_j h_j + b_j\right)$$



What is a neural network?

$$h_{j+1} = \sigma\left(W_j h_j + b_j\right)$$



What is a neural network?

- Nonlinear mapping of the form

$$\mathcal{N} = L_J \circ L_{J-1} \circ \cdots \circ L_2 \circ L_1$$

with layers

$$L_j \, : \, h_{j+1} = \sigma\left(\boxed{W_j} h_j + \boxed{b_j}\right)$$

Parameters that are learned / fitted to data

What is a neural network?

- Training of neural network using data $D = \{x_i, y_i\}_{i=1}^N$

$$\underset{\theta \in \mathrm{R}^\tau}{\operatorname{argmin}} \, \mathcal{L}_\theta(D; \theta)$$

What is a neural network?

- Training of neural network using data $D = \{x_i, y_i\}_{i=1}^{N}$

$$\underset{\theta \in \mathrm{R}^{\tau}}{\operatorname{argmin}} \mathcal{L}_{\theta}(D; \theta)$$

Weight matrices and bias vectors

What is a neural network?

- Training of neural network using data $D = \{x_i, y_i\}_{i=1}^{N}$

$$\underset{\theta \in \mathrm{R}^\tau}{\operatorname{argmin}} \mathcal{L}_\theta(D; \theta)$$

with loss function

$$\mathcal{L}_\theta(D; \theta) = \sum_{i=1}^{N} d\big(\mathcal{N}_\theta(x_i)\,,\; y_i\big)$$

What is a neural network?

- Training of neural network using data $D = \{x_i, y_i\}_{i=1}^N$

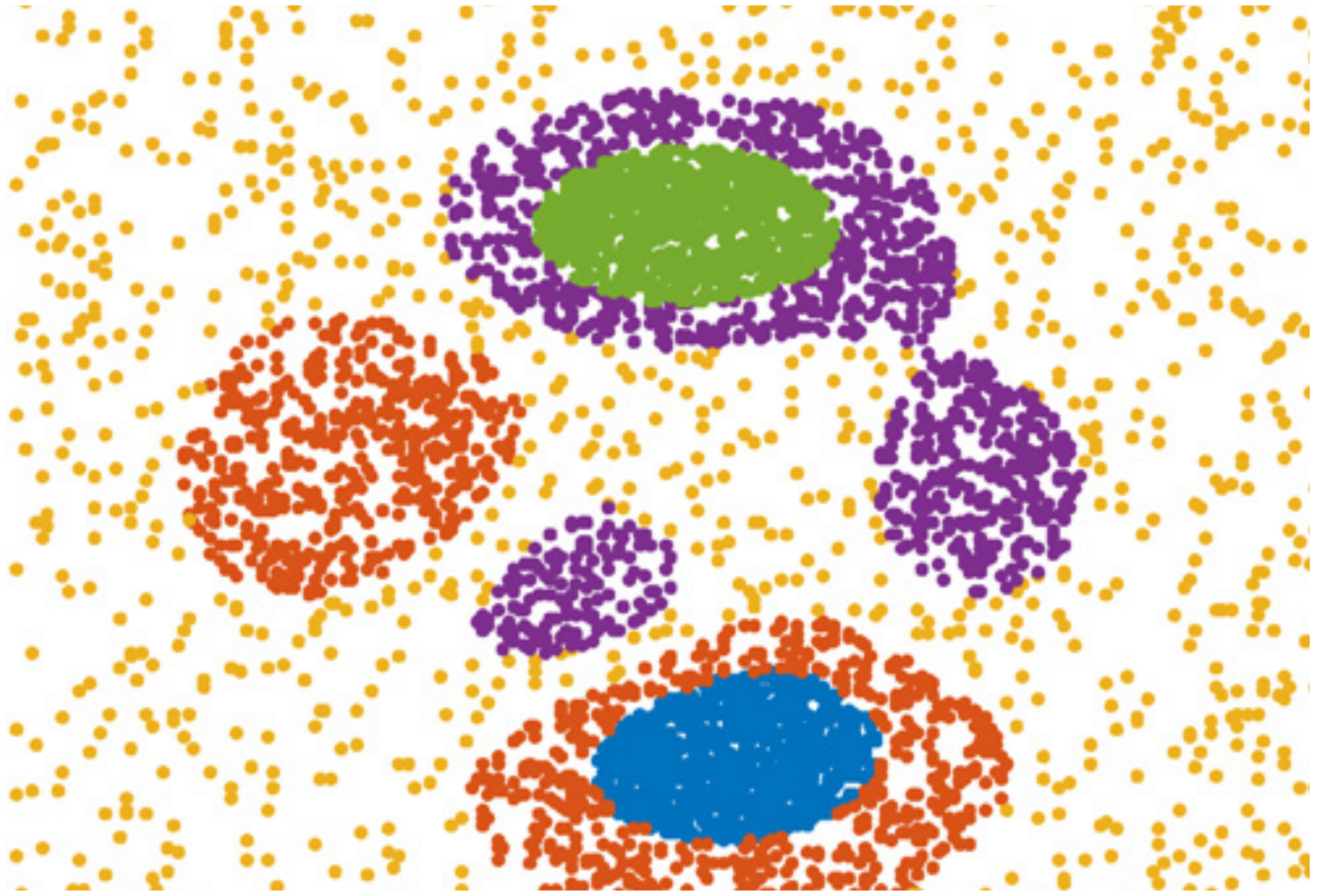
$$\underset{\theta \in \mathrm{R}^\tau}{\operatorname{argmin}} \mathcal{L}_\theta(D; \theta)$$

Objective:

> Good performance on *unseen* data from the same distribution as D.

What is a neural network?

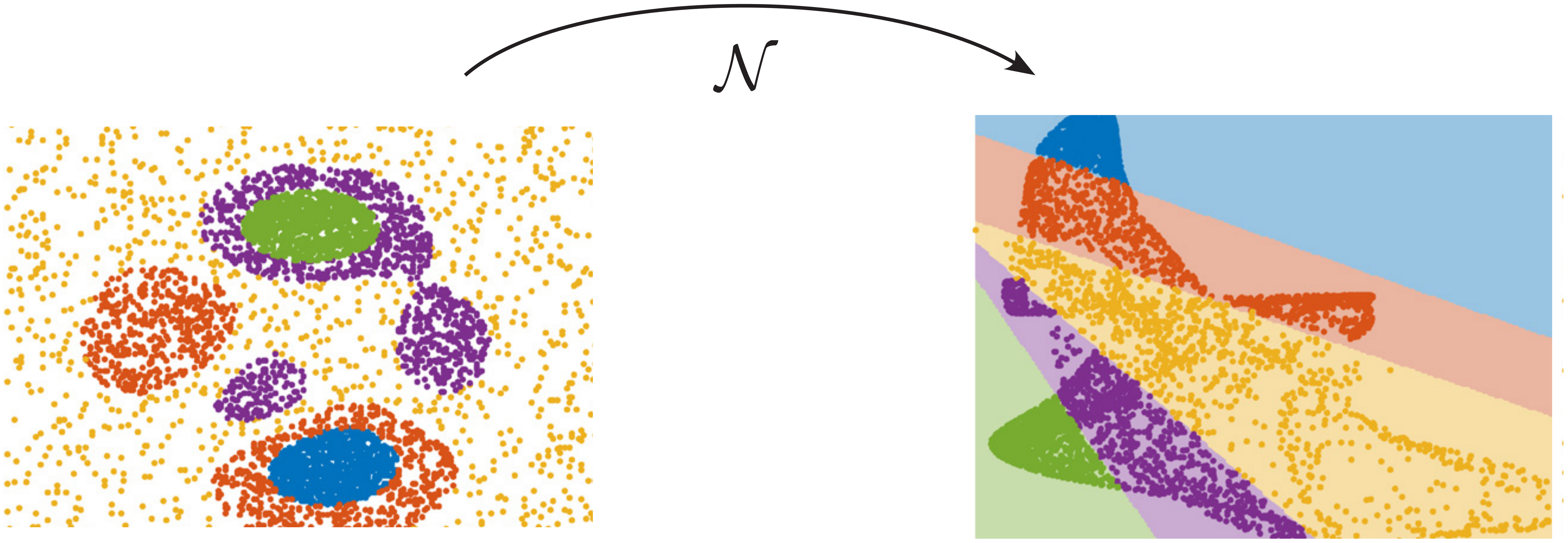
- Classification:



Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.

What is a neural network?

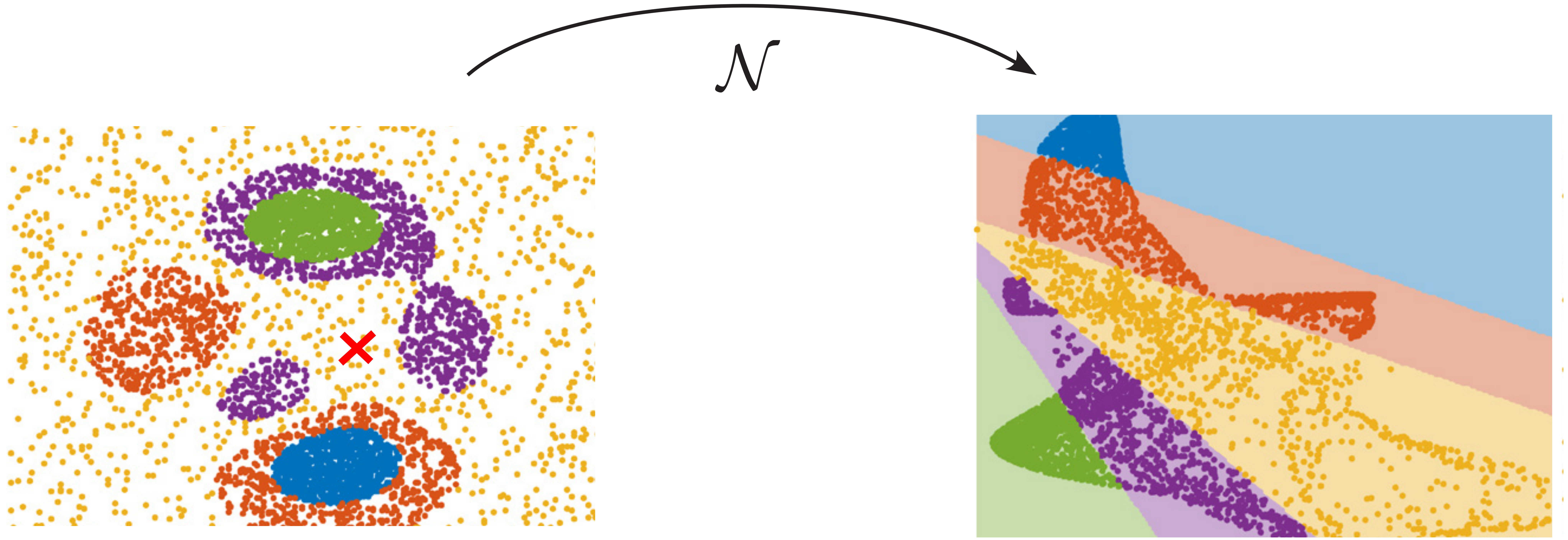
- Classification:



Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.

What is a neural network?

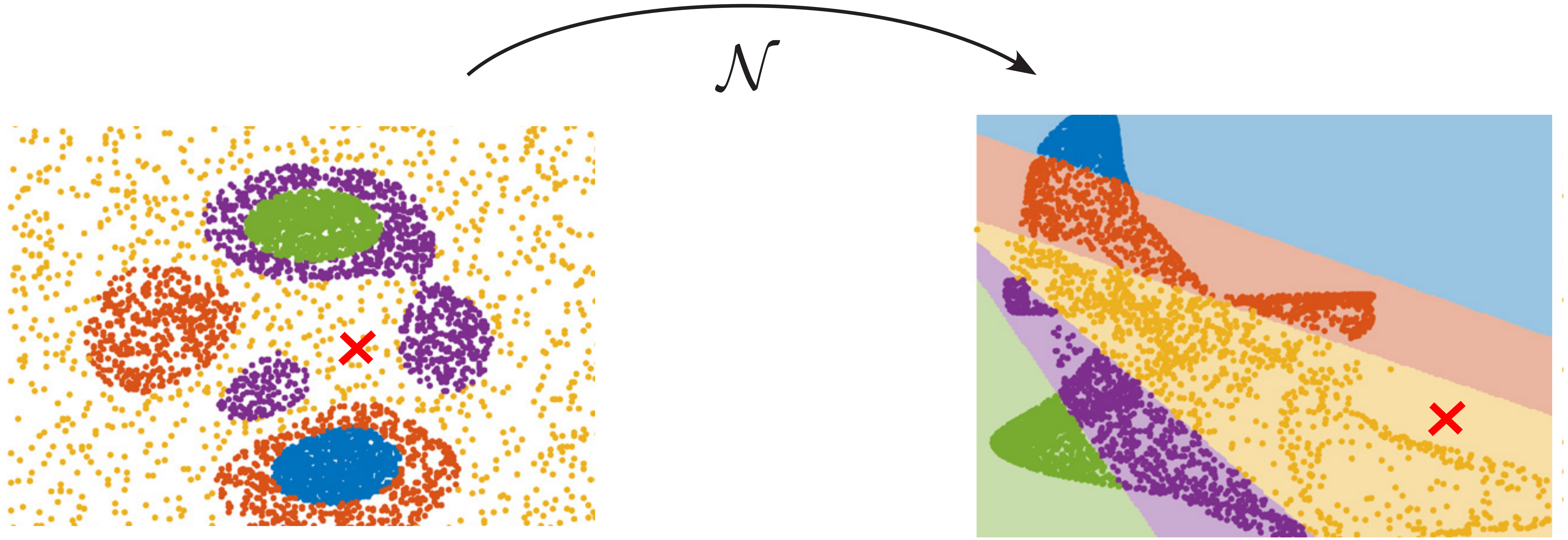
- Classification:



Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.

What is a neural network?

- Classification:



Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.

What is a neural network?

- Neural net is nonlinear map
  - Between vector spaces / manifolds

What is a neural network?

- Neural net is nonlinear map
  - Between vector spaces / manifolds
  - Consists of simple building blocks with simple nonlinearity

What is a neural network?

- Neural net is nonlinear map
  - Between vector spaces / manifolds
  - Consists of simple building blocks with simple nonlinearity
  - Linear superposition (as in bases, frames) is replaced by nonlinear composition

Neural nets as ordinary differential equations

Neural nets as ordinary differential equations

- Residual neural network

$$L_j \; : \; h_{j+1} = h_j + \sigma\big(W_j h_j + b_j\big)$$

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

Neural nets as ordinary differential equations

- Residual neural network

$$L_j \ : \ h_{j+1} = h_j + \alpha \, \sigma\big(W_j h_j + b_j\big)$$

Neural nets as ordinary differential equations

- Residual neural network

$$L_j \;:\; h_{j+1} = h_j + \boxed{\alpha}\,\sigma\big(W_j h_j + b_j\big)$$

Neural nets as ordinary differential equations

- Residual neural network

$$L_j \ : \ h_{j+1} = h_j + \alpha \, \sigma\big(W_j h_j + b_j\big)$$

With $\alpha = \Delta t$

Neural nets as ordinary differential equations

- Residual neural network

$$L_j \; : \; h_{j+1} = h_j + \alpha \, \sigma\big(W_j h_j + b_j\big)$$

With $\alpha = \Delta t$ a layer is an explicit Euler step of the nonlinear ordinary differential equation

$$\dot{h}(t) = \sigma\big(W(t) h(t) + b(t)\big)$$

Neural nets as ordinary differential equations

- Residual neural network

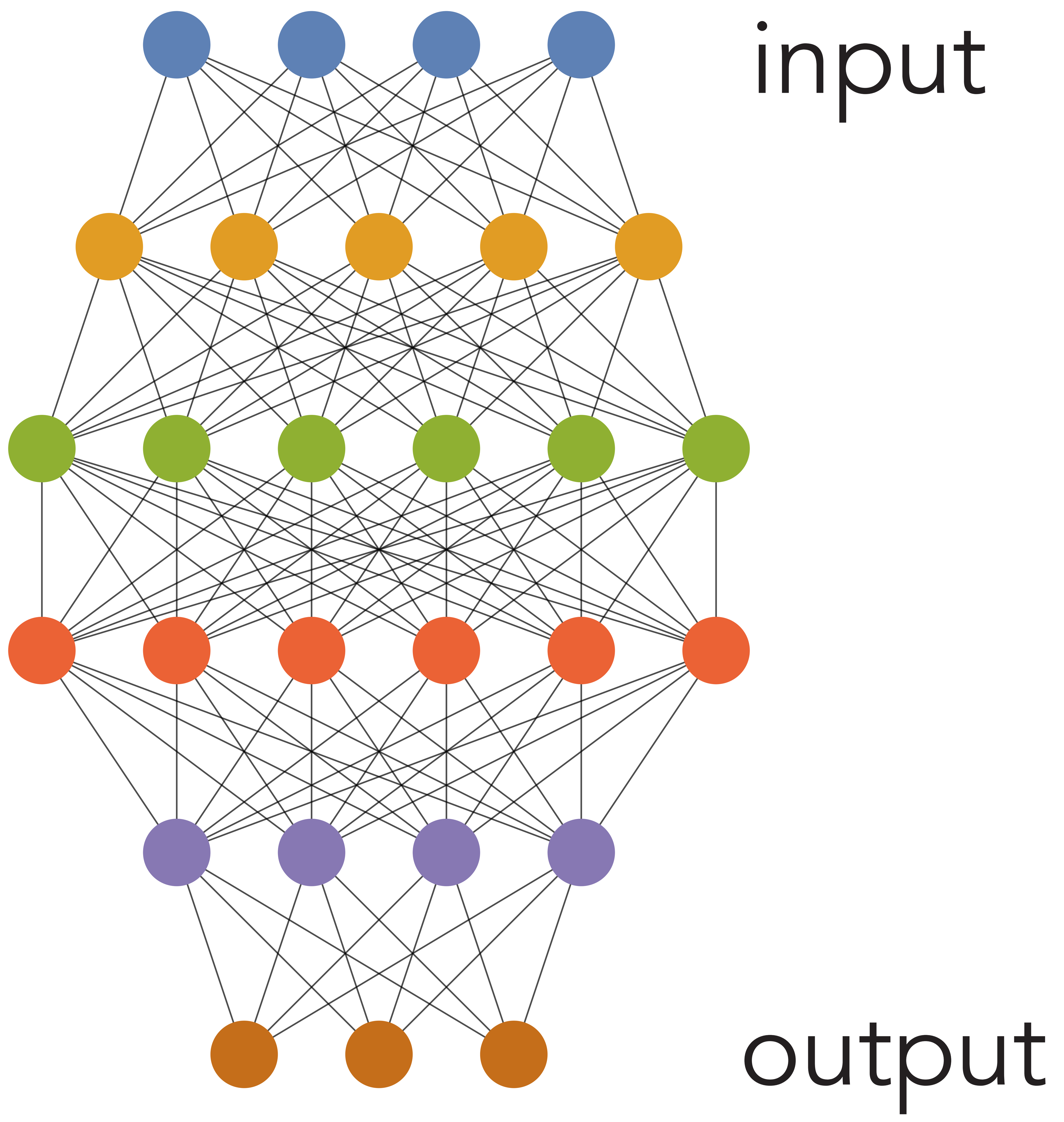
$$L_j \,:\, h_{j+1} = h_j + \alpha\, \sigma\big(W_j h_j + b_j\big)$$

With $\alpha = \Delta t$ a layer is an explicit Euler step of the nonlinear ordinary differential equation

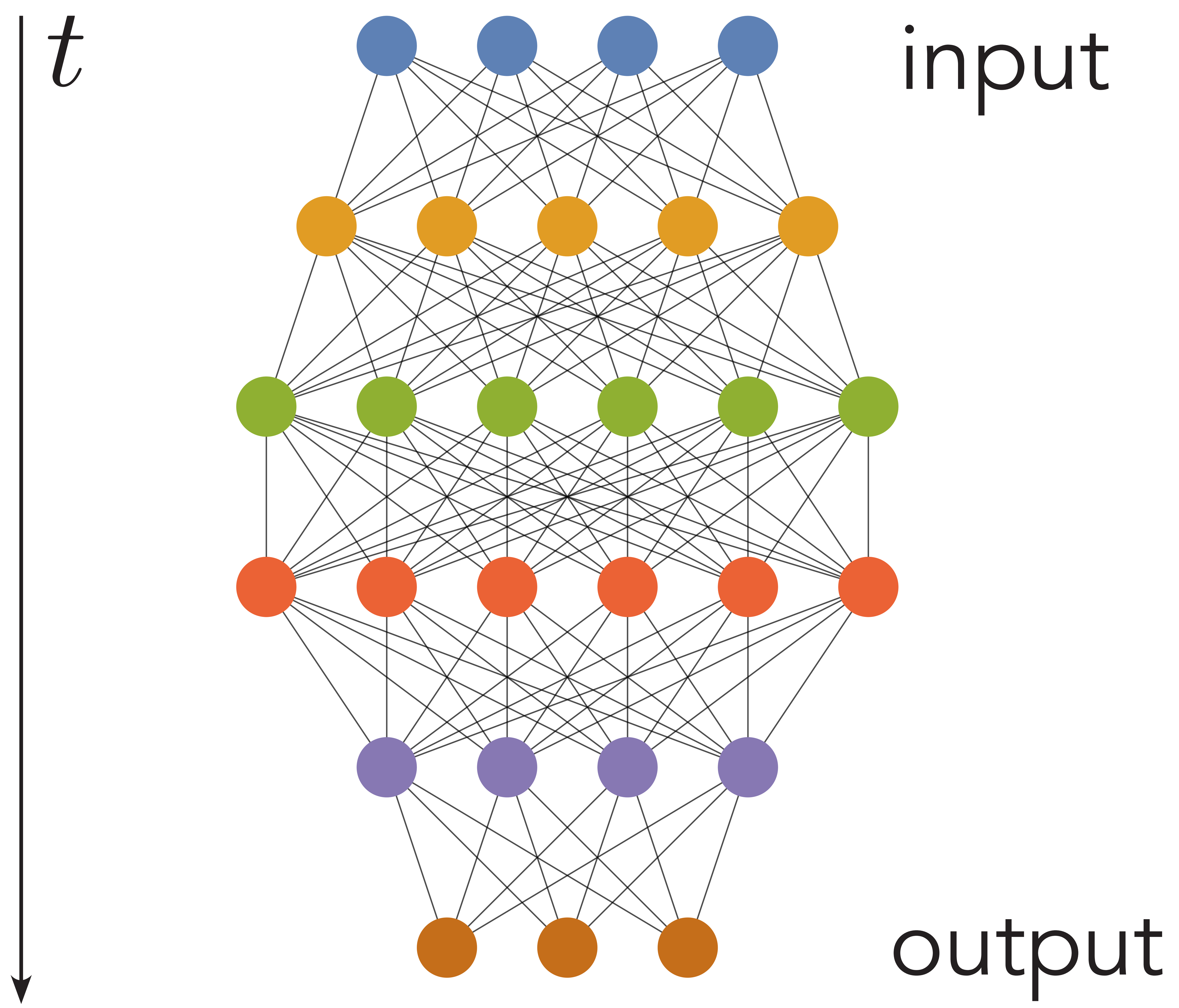
$$\dot{h}(t) = \sigma\big(W(t)h(t) + b(t)\big)$$

W. E. A proposal on machine learning via dynamical systems. Communications in Mathematics and Statistics, 5(1):1–11, 2017.
R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. NIPS 2018, jun 2018.
E. Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.

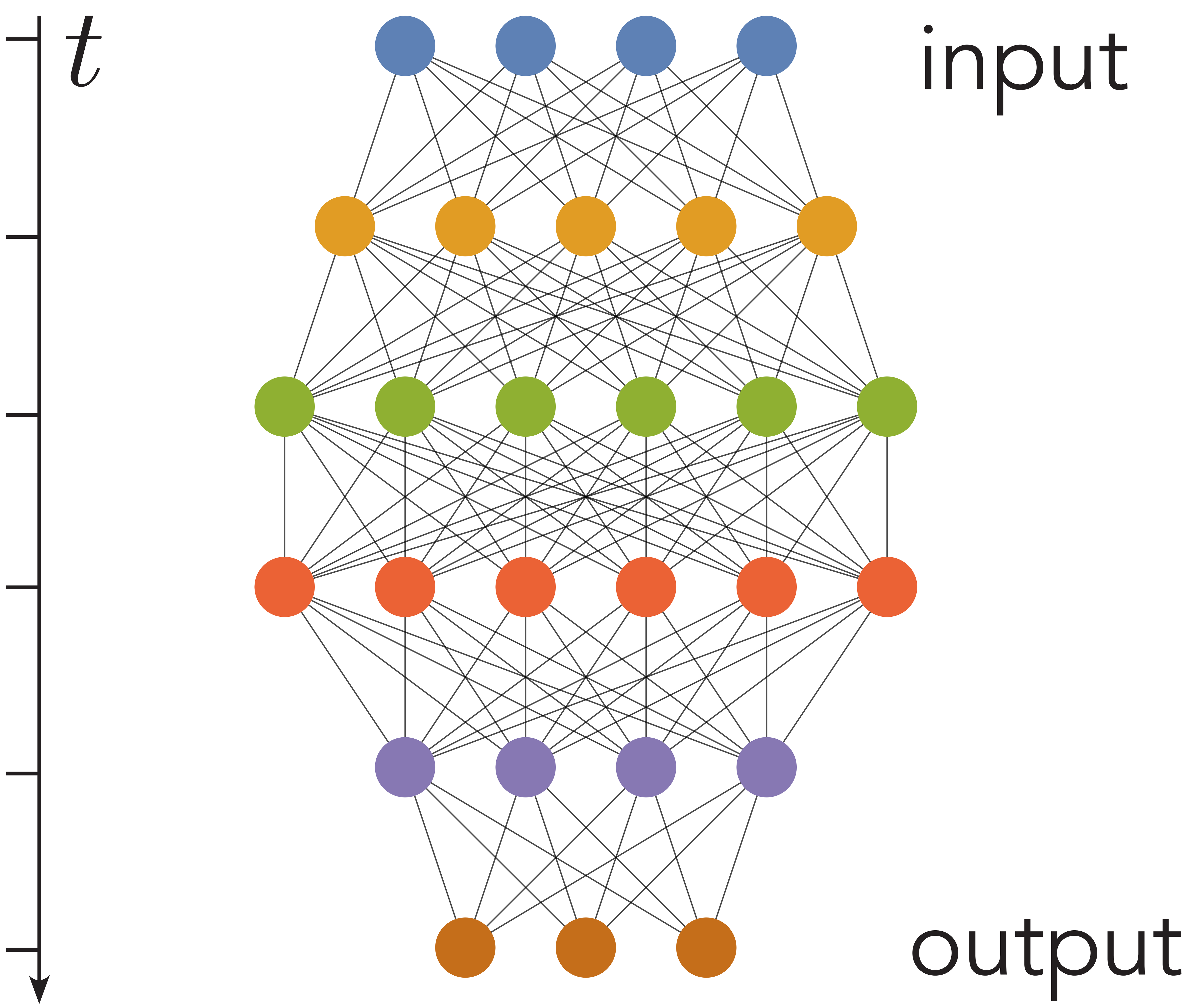
Neural nets as ordinary differential equations



Neural nets as ordinary differential equations

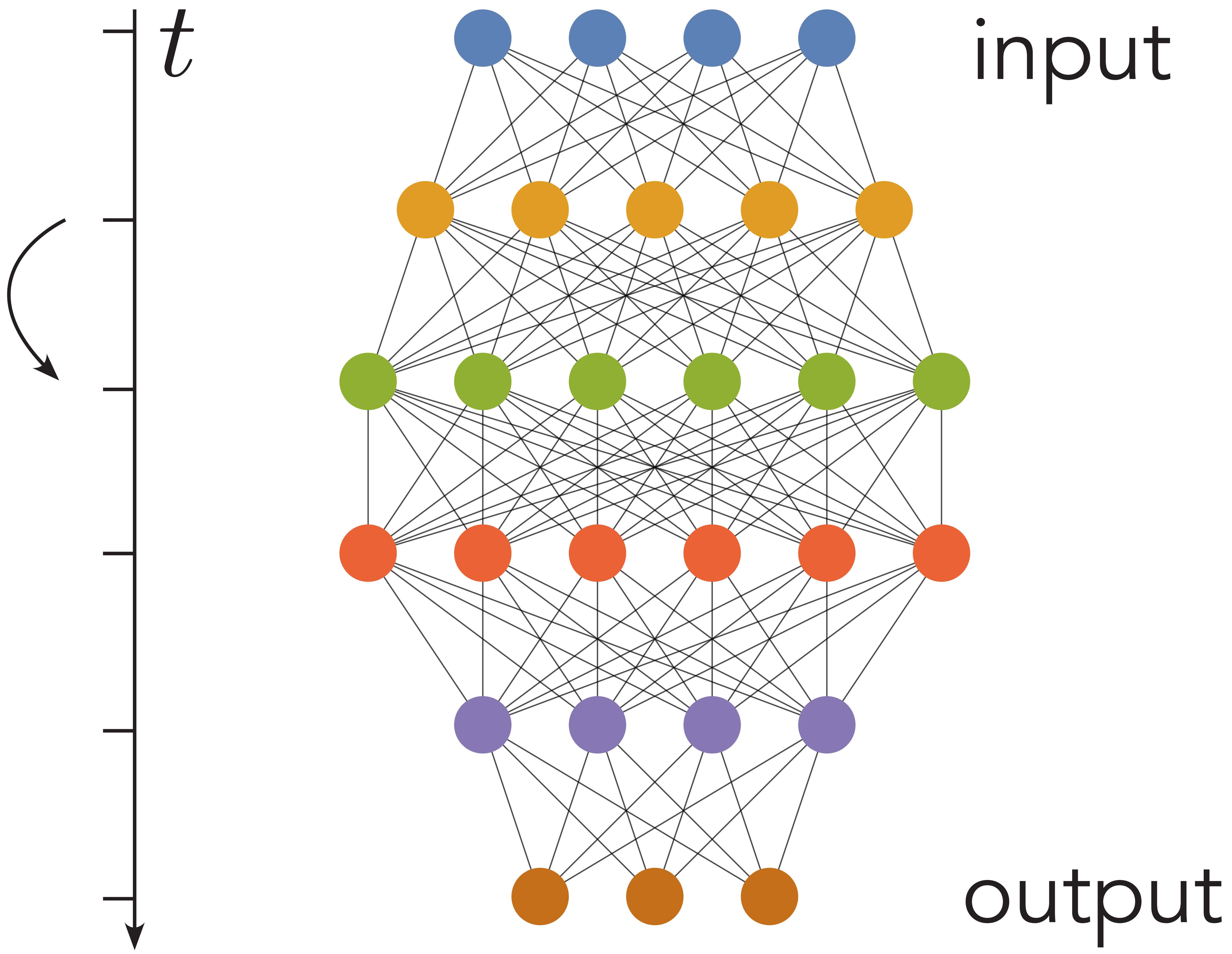


Neural nets as ordinary differential equations



Neural nets as ordinary differential equations

$$h_j + \Delta t \, \sigma\big(W_j h_j + b_j\big)$$



Neural nets as ordinary differential equations

- Higher order / implicit time integration schemes (e.g. [Chen et al. 2018])
- Stability (e.g. [Haber and Ruthotto 2017])
- Gradient computation (e.g. [E 2017], [Chang et al. 2018])
- Optimization / parameter estimation (e.g. [Li et al. 2017])

W. E. A proposal on machine learning via dynamical systems. Communications in Mathematics and Statistics, 5(1):1–11, 2017.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. NIPS 2018, jun 2018.

E. Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.

Q. Li, L. Chen, C. Tai, and E. Weinan. Maximum principle based algorithms for deep learning. J. Mach. Learn. Res., 18(1):5998–6026, Jan. 2017.

B. Chang, L. Meng, et al.. Reversible architectures for arbitrarily deep residual neural networks. In AAAI Conference on Artificial Intelligence, 2018.

Neural nets as ordinary differential equations

- Stability:

$$\big\|y_\theta(T) - \tilde{y}_\theta(T)\big\| \le M\big\|y_\theta(0) - \tilde{y}_\theta(0)\big\|$$

Neural nets as ordinary differential equations

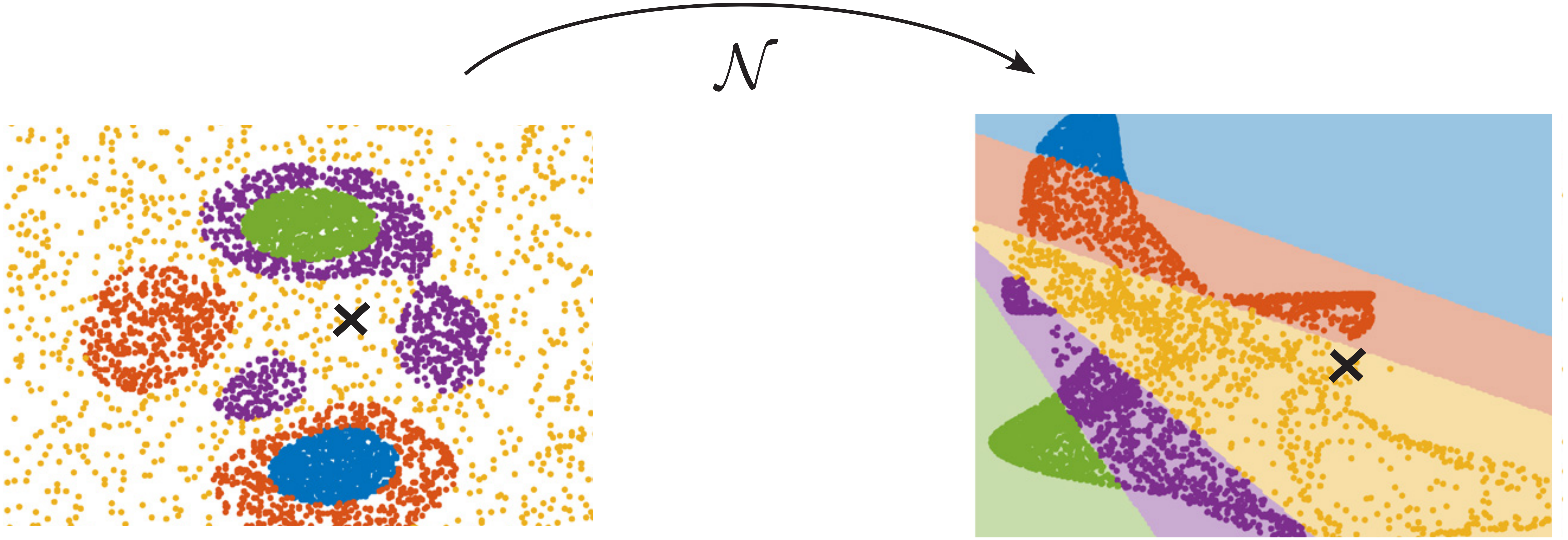
- Stability:

$$\left\| y_\theta(T) - \tilde{y}_\theta(T) \right\| \leq M \left\| y_\theta(0) - \tilde{y}_\theta(0) \right\|$$

Objective:

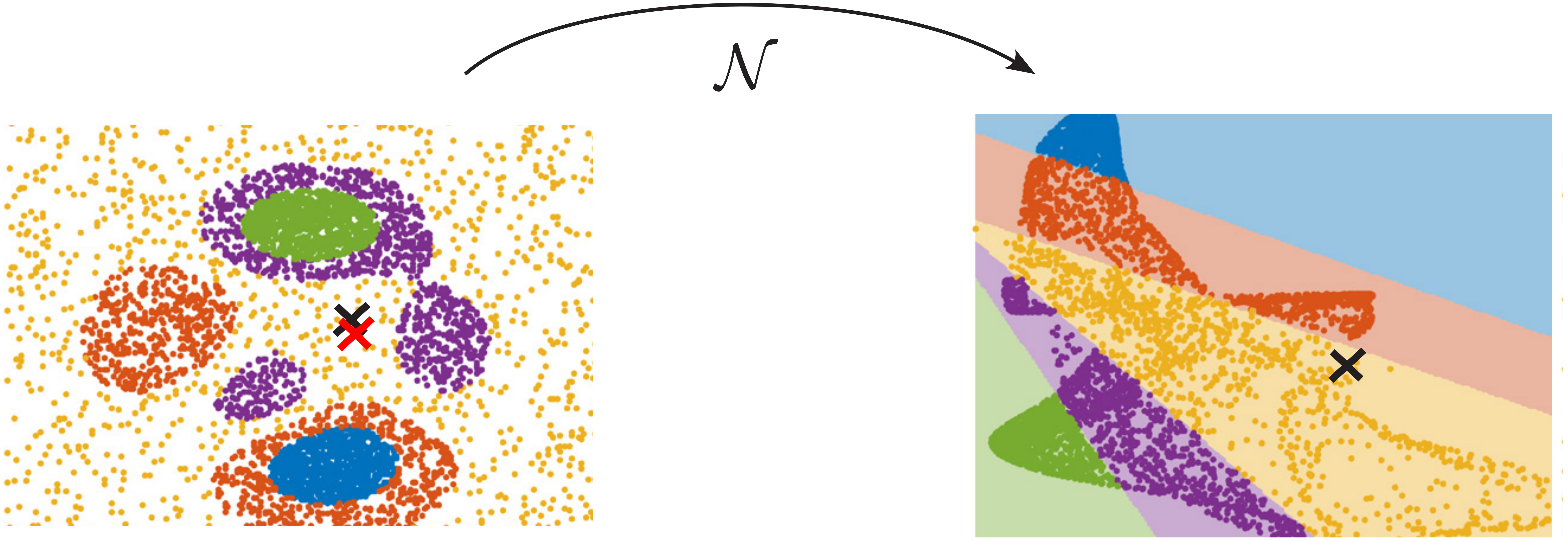
$$\boxed{\text{Good performance on unseen data from the same distribution as } D\,.}$$

Neural nets as ordinary differential equations



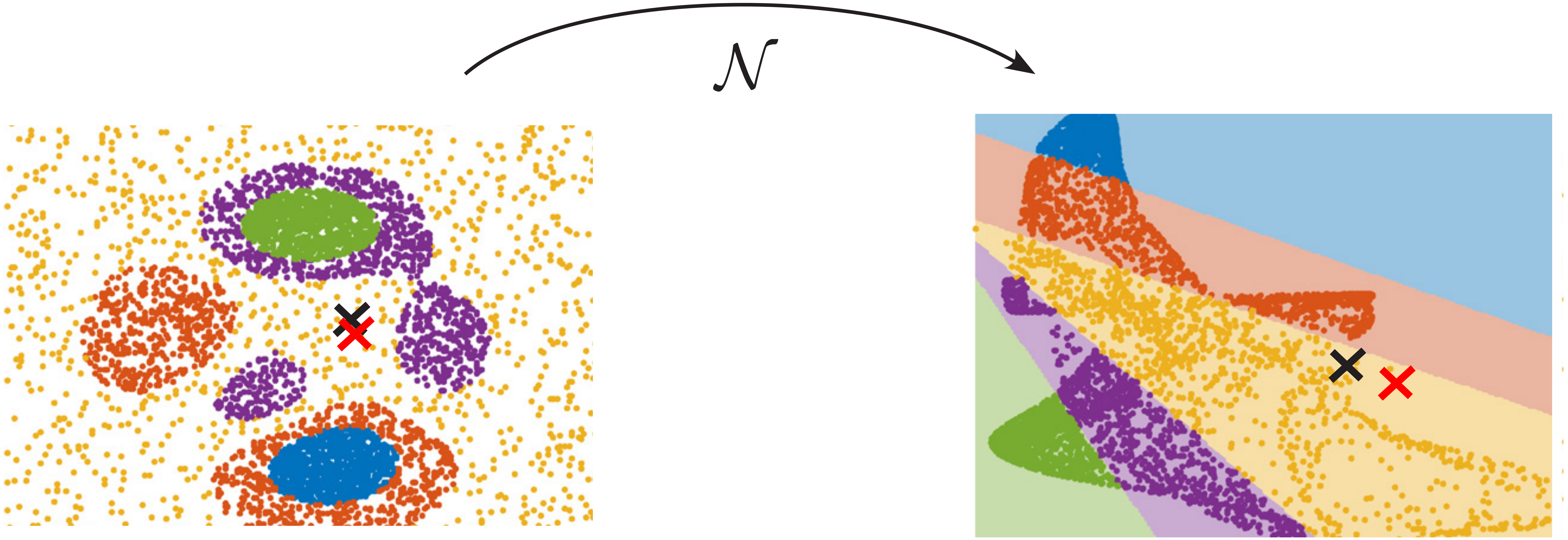
Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.

Neural nets as ordinary differential equations



Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.

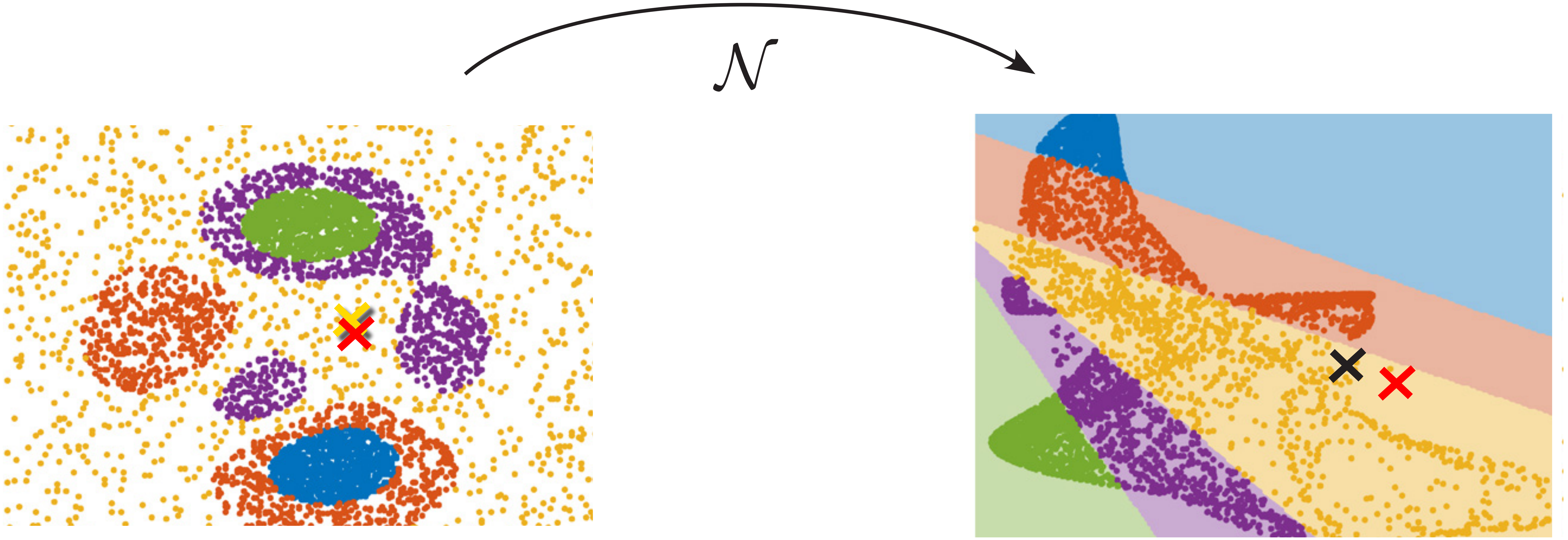
Neural nets as ordinary differential equations



Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.

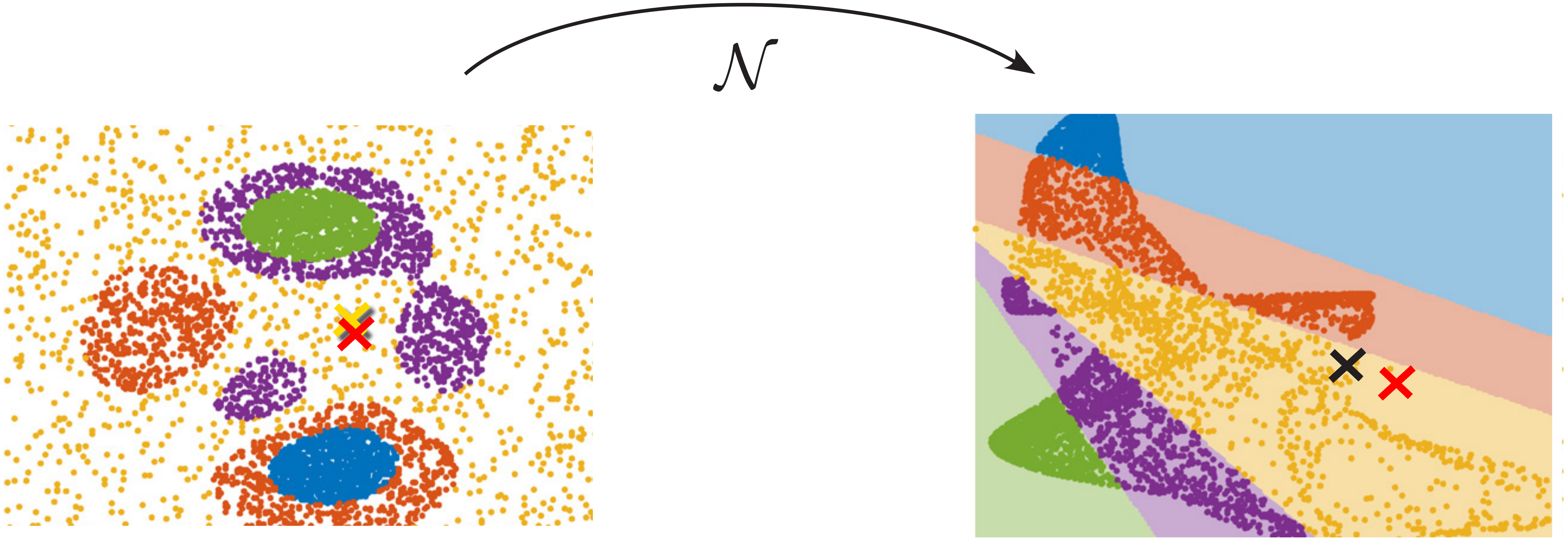
Neural nets as ordinary differential equations

Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.



We want good performance on perturbations of the training data!

Neural nets as ordinary differential equations



Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.

We want good performance on $\boxed{\text{perturbations}}$ of the training data!

Neural nets as ordinary differential equations

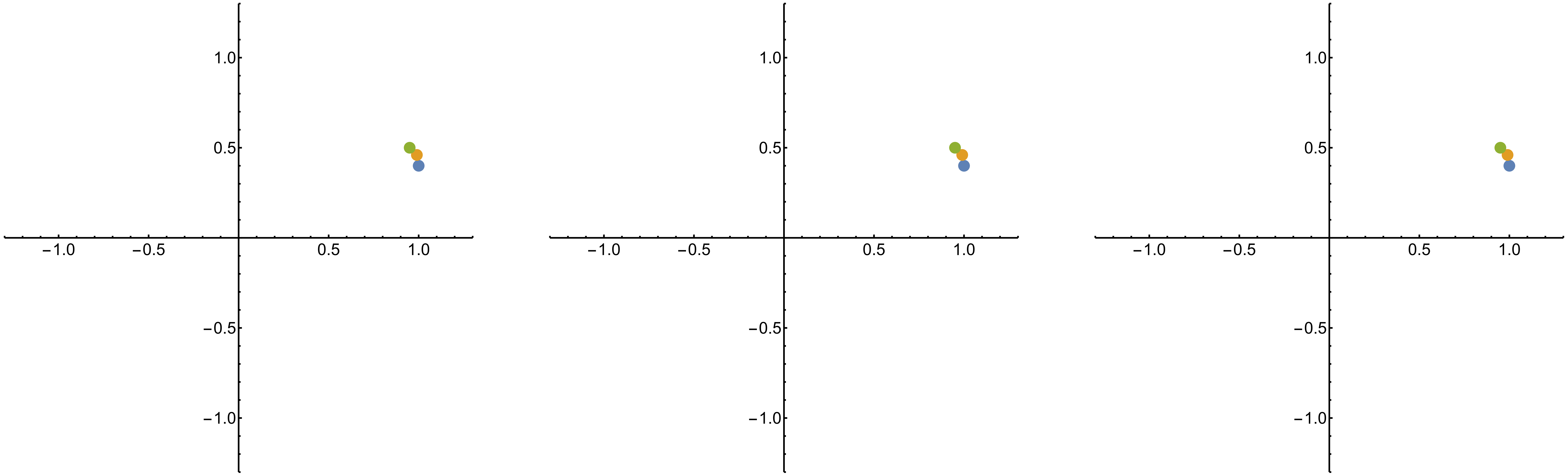
- Simple example with 20 layers given by:

$$L_j \,:\, h_{j+1} = h_j + \alpha\,\sigma\big(W_j h_j + b_j\big)$$

$$h_j \in \mathbb{R}^2 \qquad\qquad \sigma = \tanh$$

$$\alpha = 0.1 \qquad\qquad b_j = (0,0)^T$$

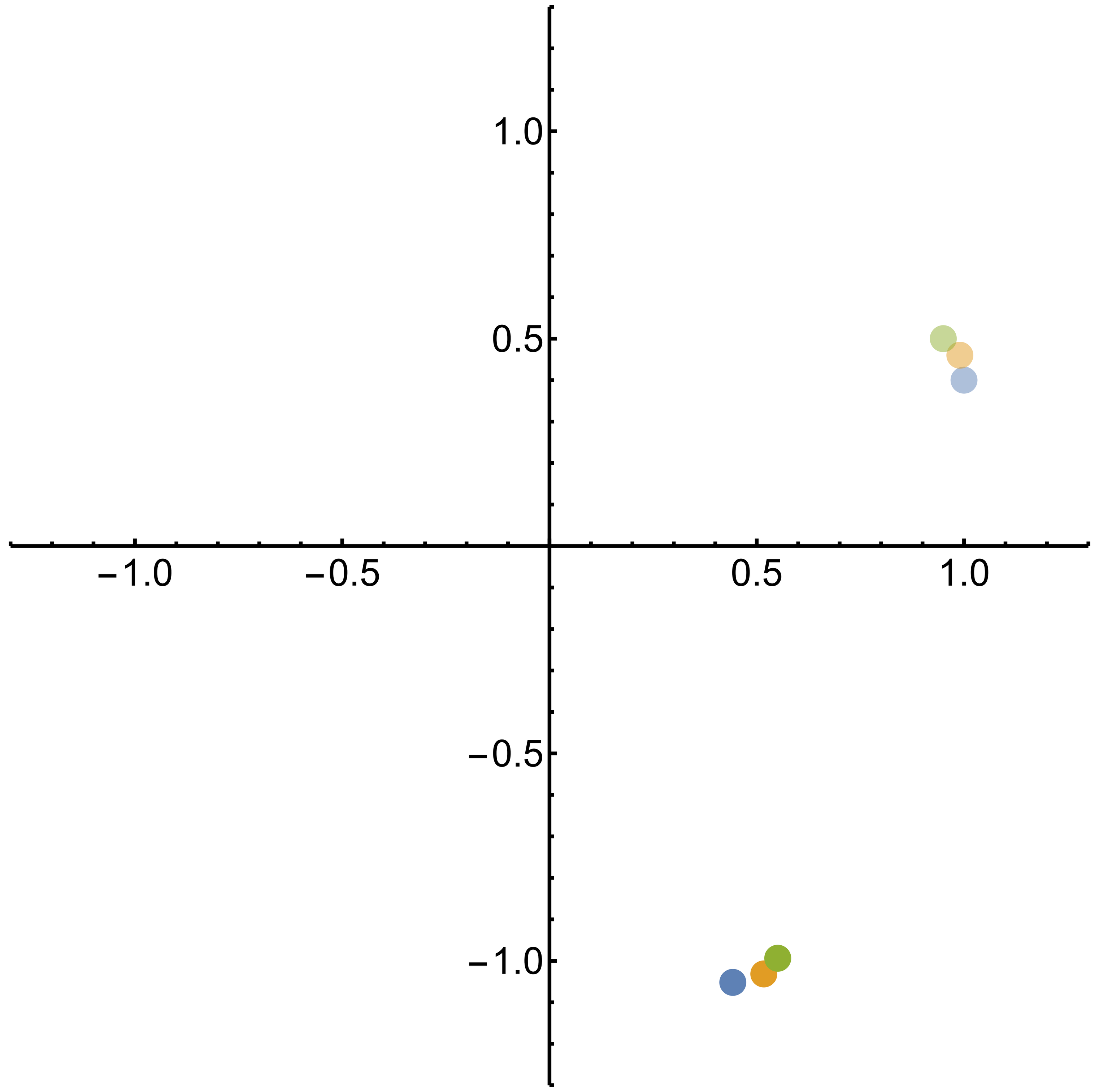
$$W_j = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \qquad W_j = \begin{pmatrix} -2 & 0 \\ 2 & -2 \end{pmatrix} \qquad W_j = \begin{pmatrix} 2 & -2 \\ 0 & 2 \end{pmatrix}$$

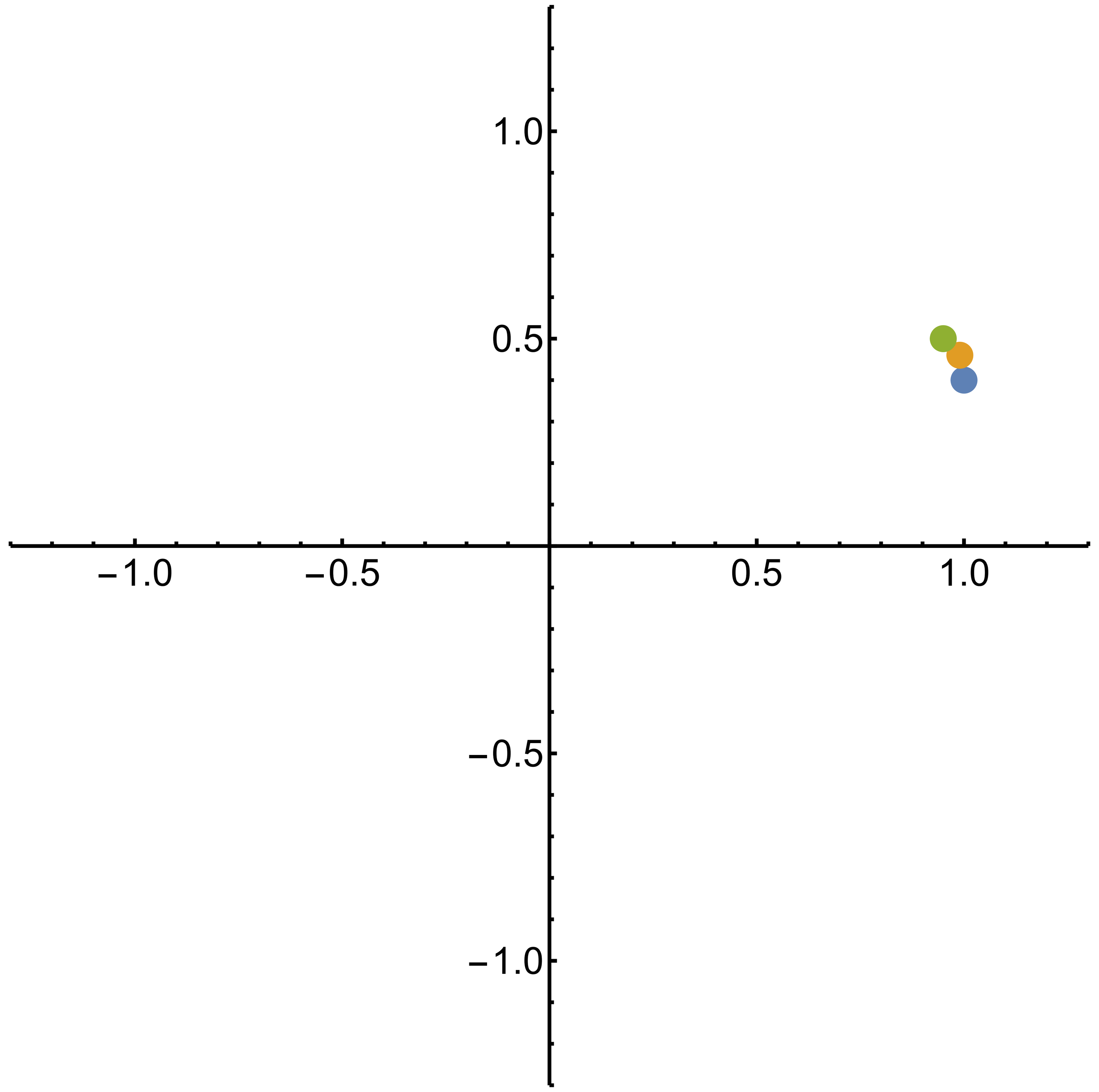


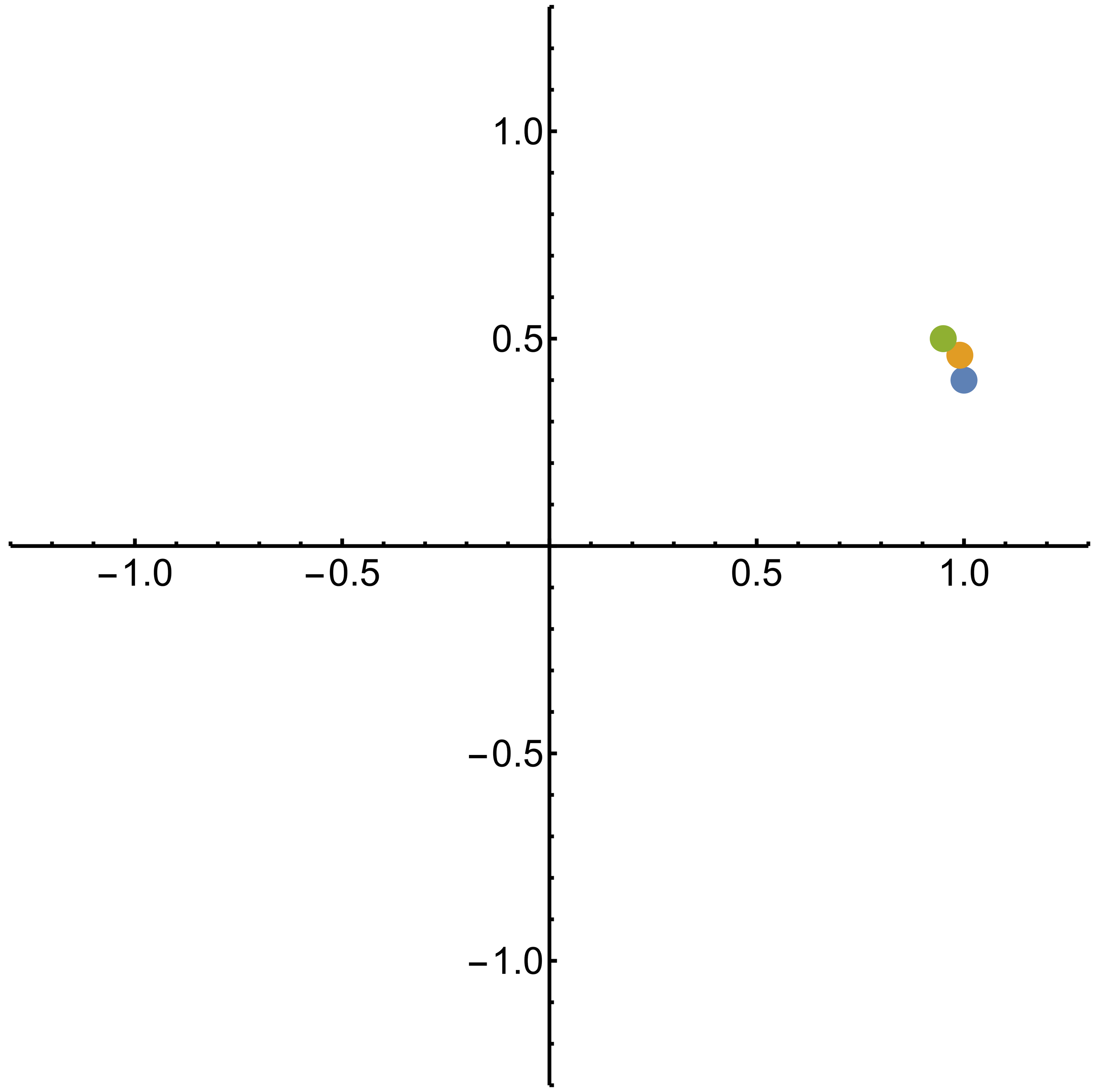
$$W_j = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

$$W_j = \begin{pmatrix} -2 & 0 \\ 2 & -2 \end{pmatrix}$$

$$W_j = \begin{pmatrix} 2 & -2 \\ 0 & 2 \end{pmatrix}$$



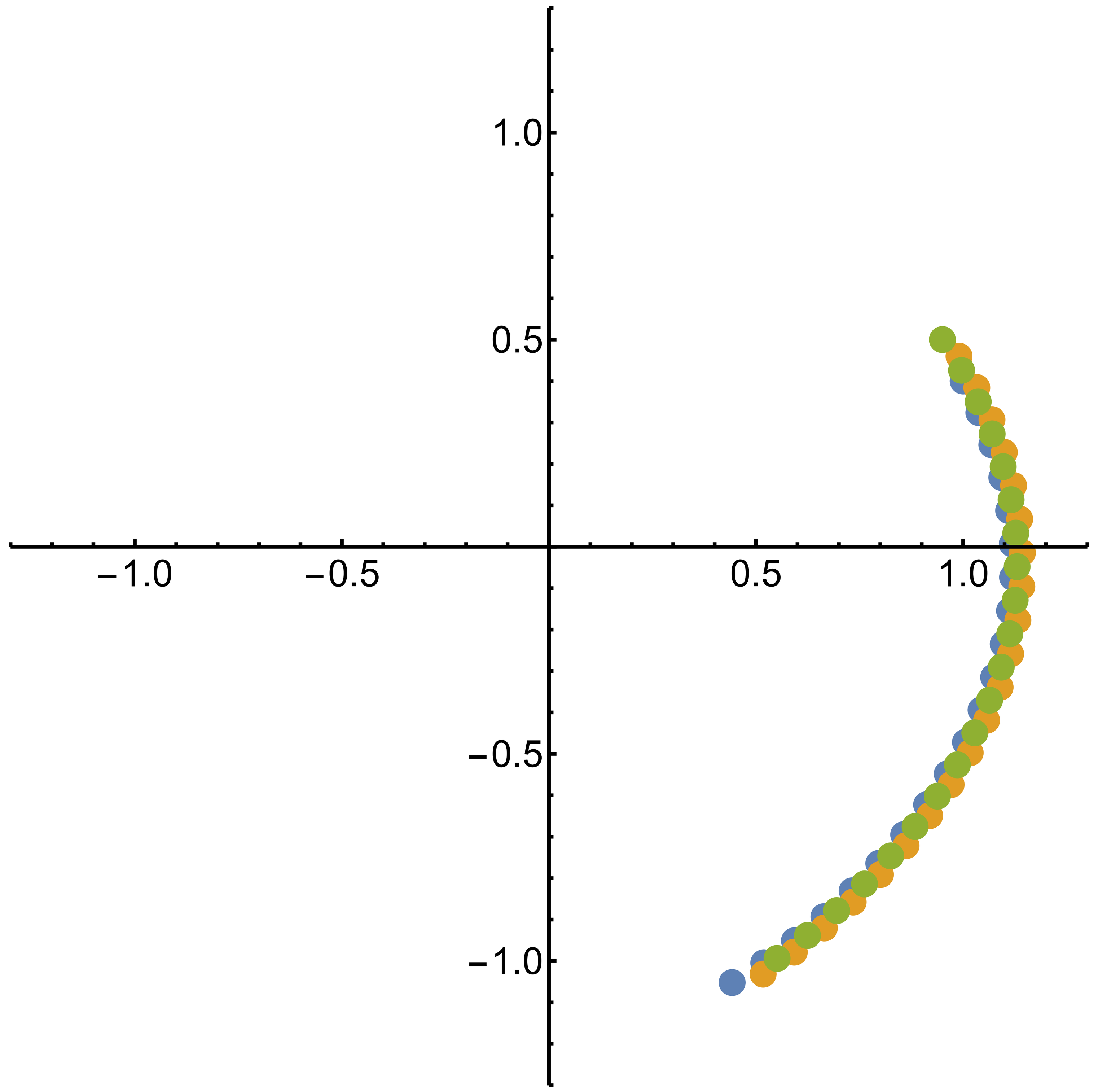


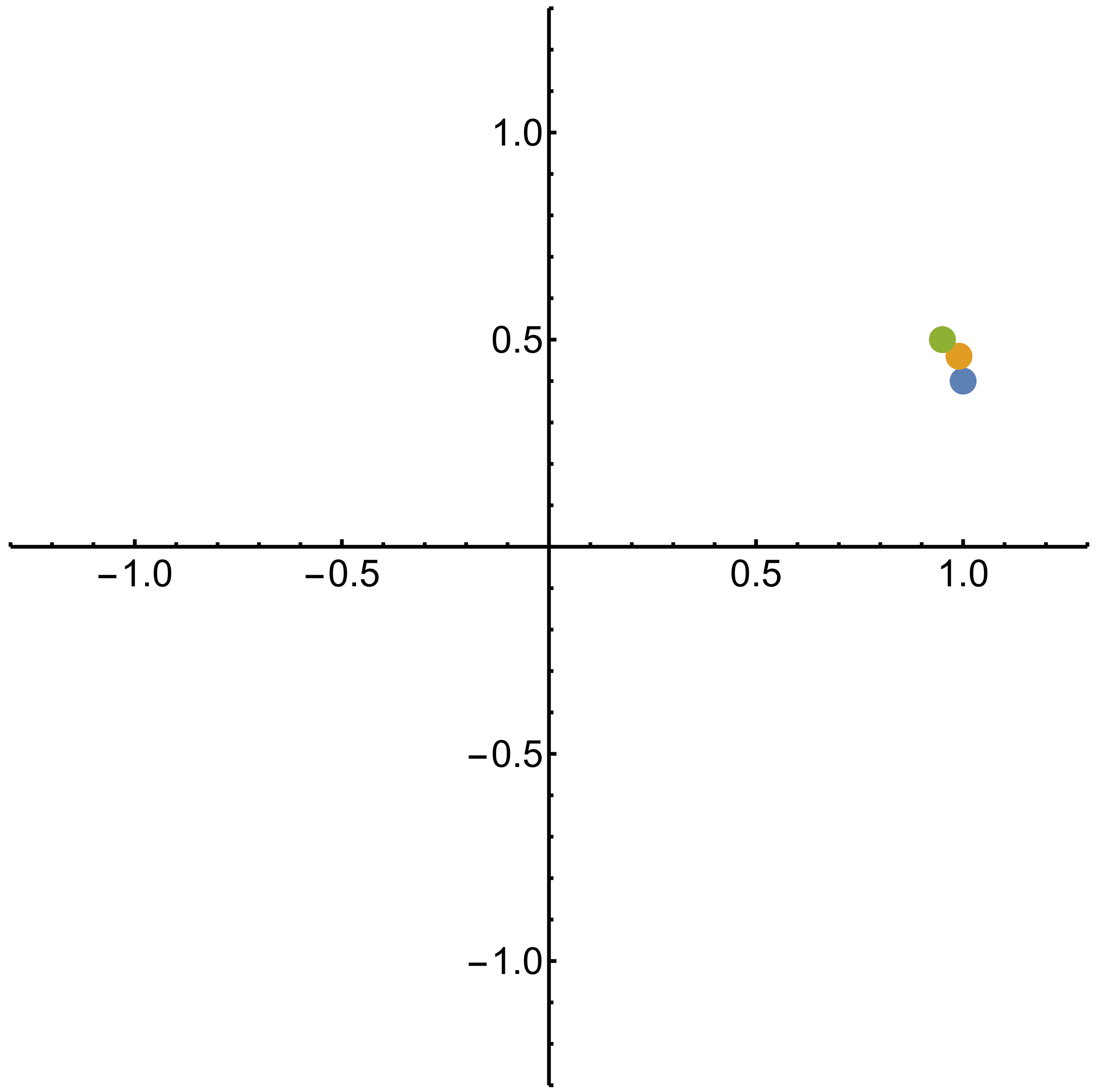


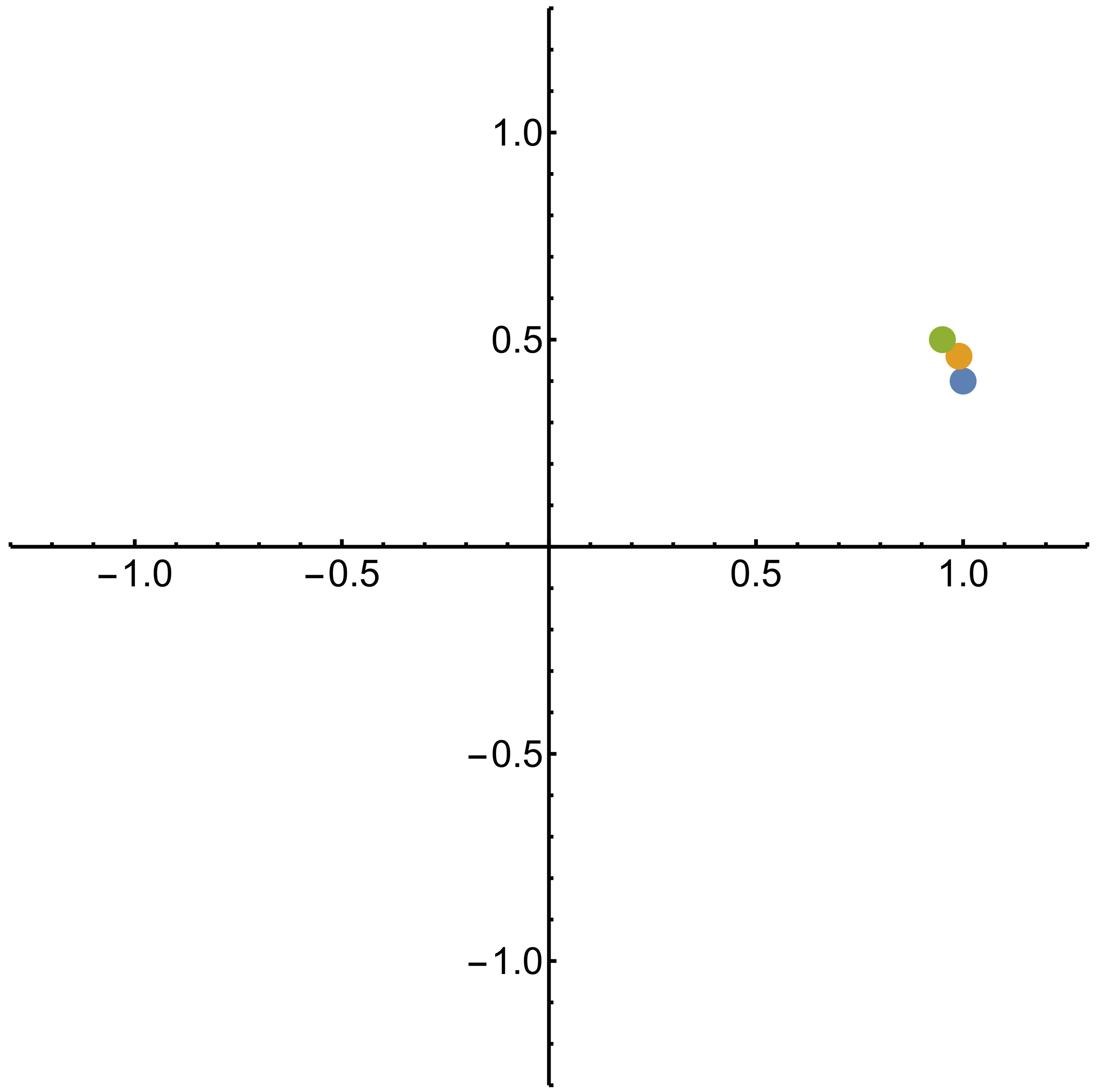
$$W_j = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

$$W_j = \begin{pmatrix} -2 & 0 \\ 2 & -2 \end{pmatrix}$$

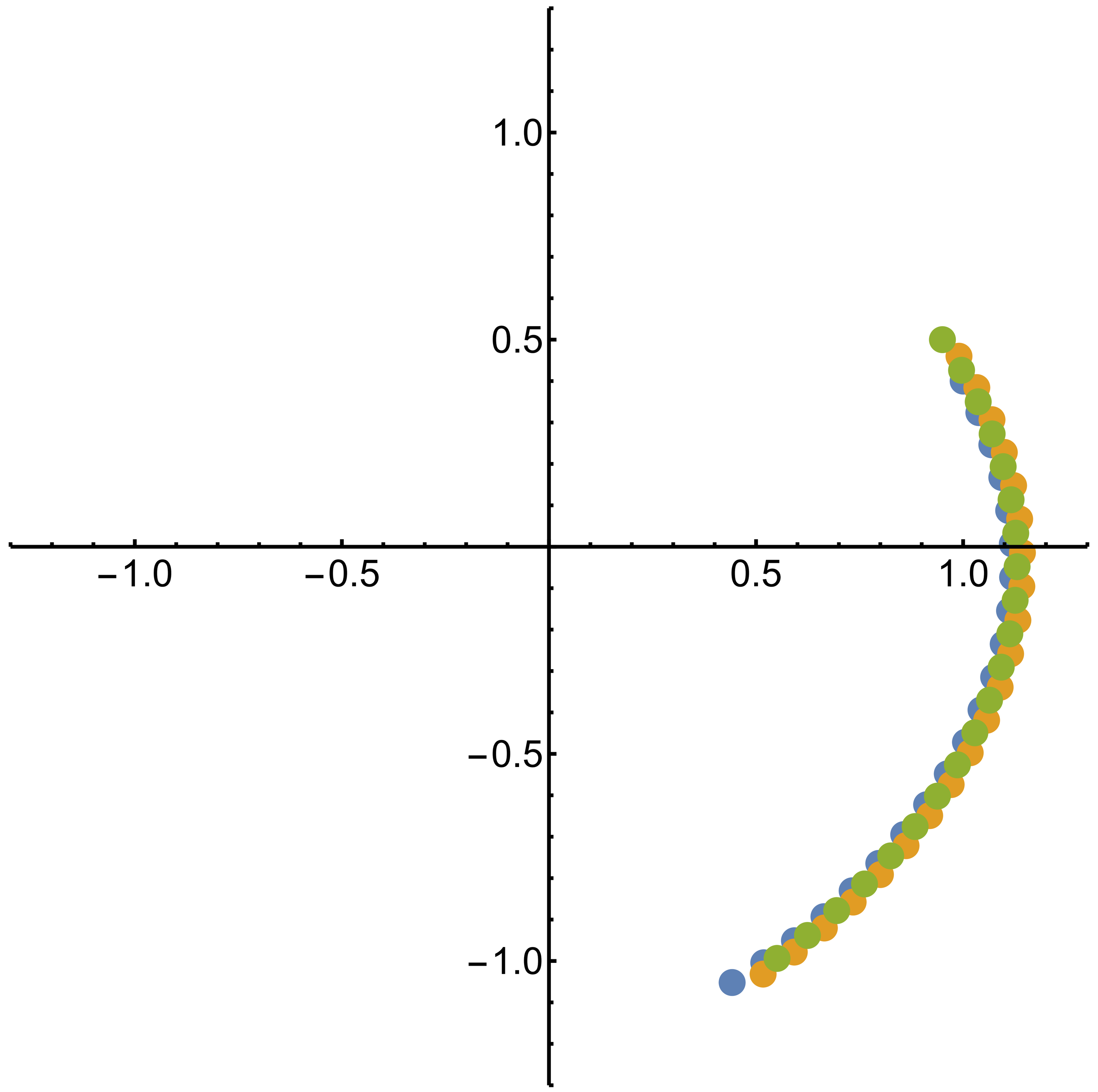
$$W_j = \begin{pmatrix} 2 & -2 \\ 0 & 2 \end{pmatrix}$$



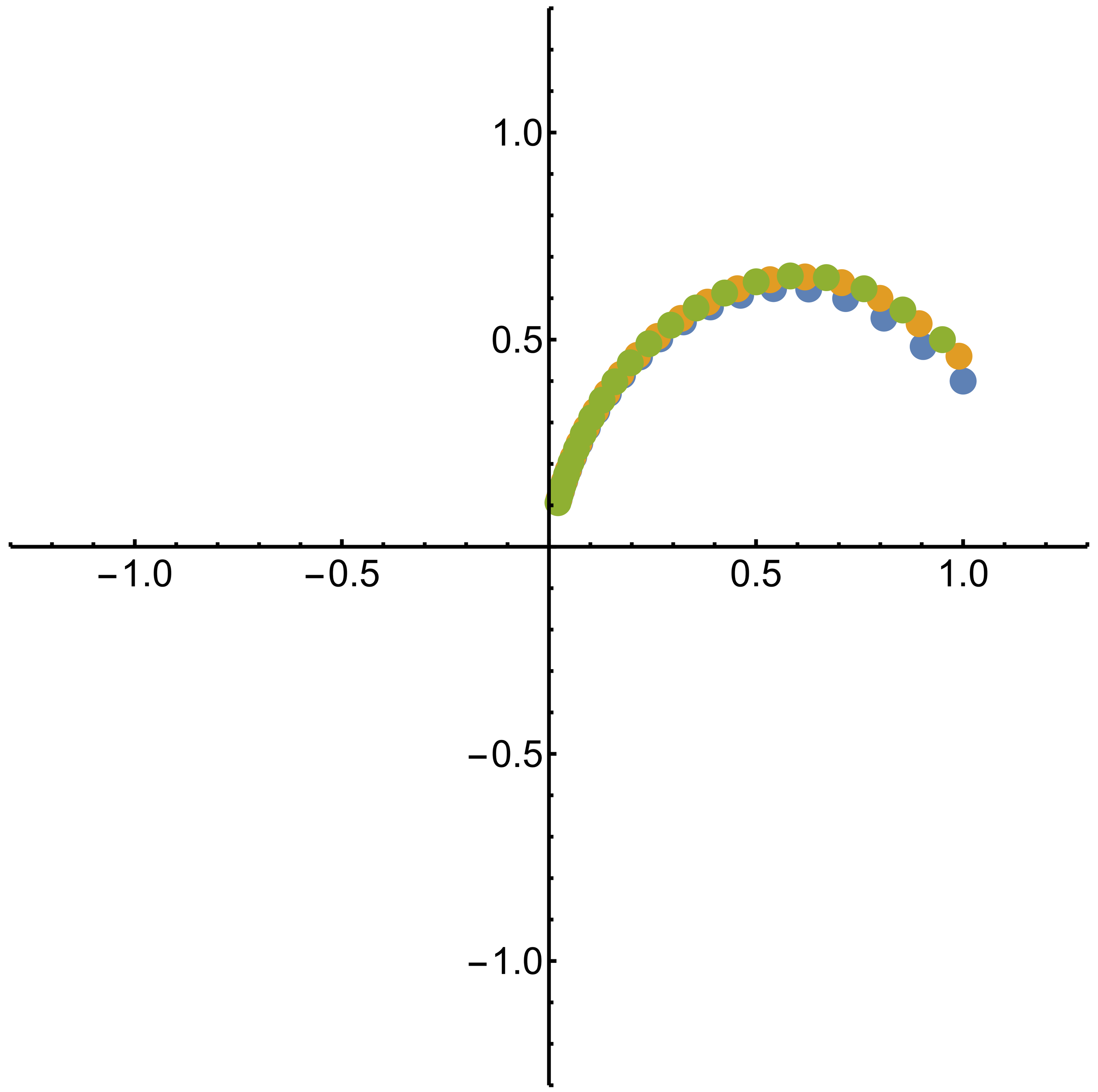




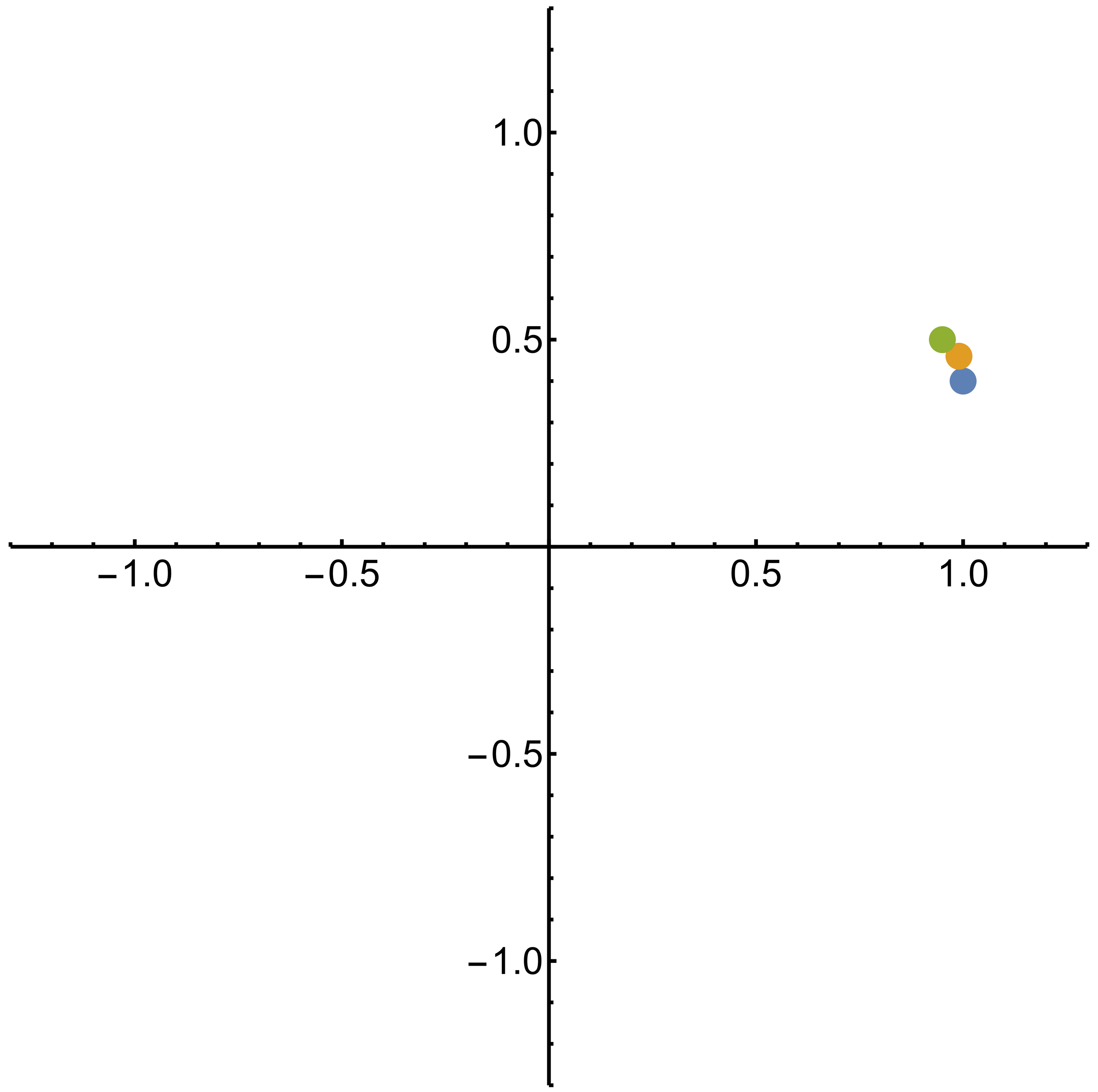
$$W_j = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$



$$W_j = \begin{pmatrix} -2 & 0 \\ 2 & -2 \end{pmatrix}$$



$$W_j = \begin{pmatrix} 2 & -2 \\ 0 & 2 \end{pmatrix}$$



$$W_j = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \qquad W_j = \begin{pmatrix} -2 & 0 \\ 2 & -2 \end{pmatrix} \qquad W_j = \begin{pmatrix} 2 & -2 \\ 0 & 2 \end{pmatrix}$$

Neural nets as ordinary differential equations

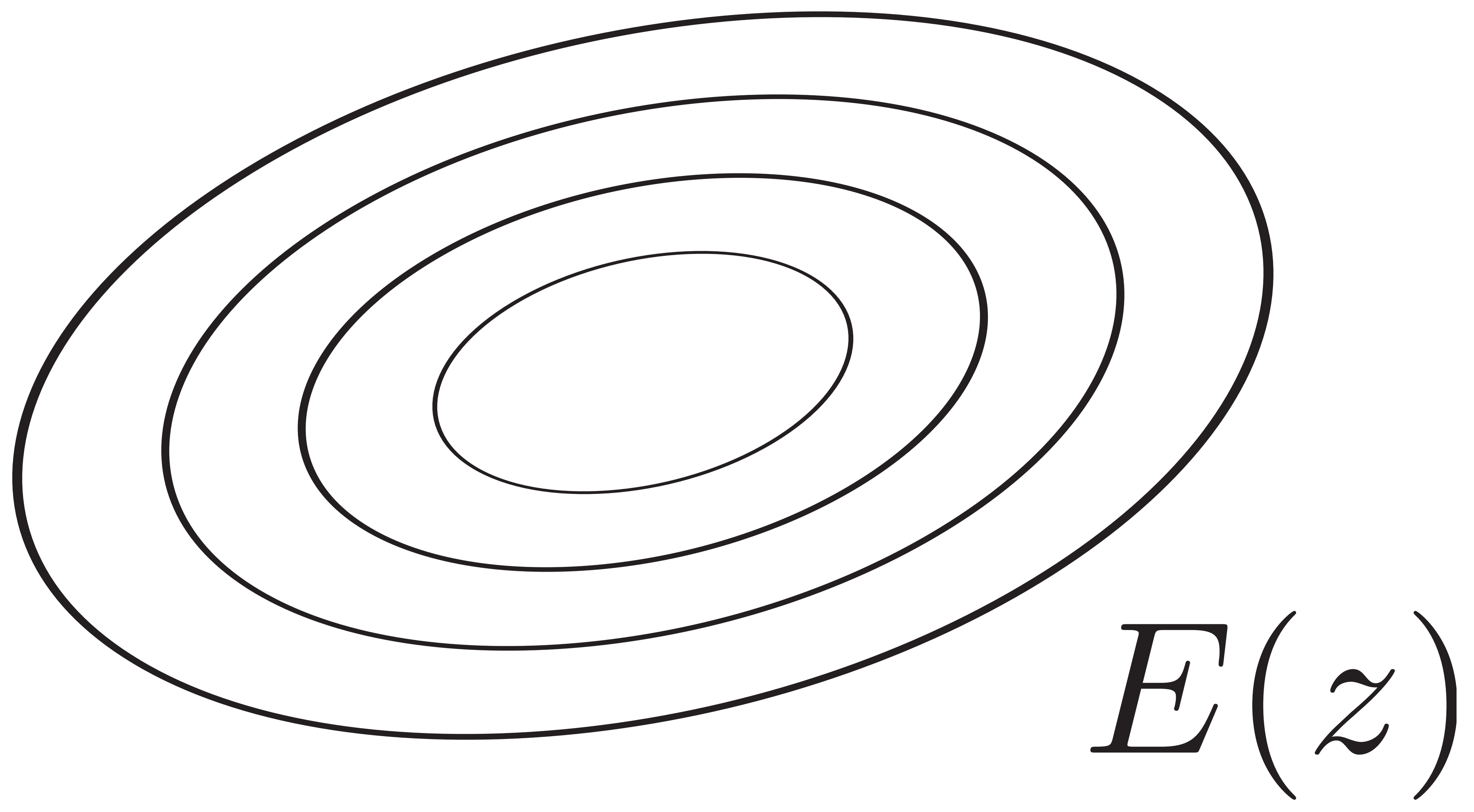
- Stability ≈ generalization

Neural nets as ordinary differential equations

- Stability ≈ generalization
- Stability is necessary condition for well posedness of (inverse) learning problem ≈ vanishing / exploding gradients

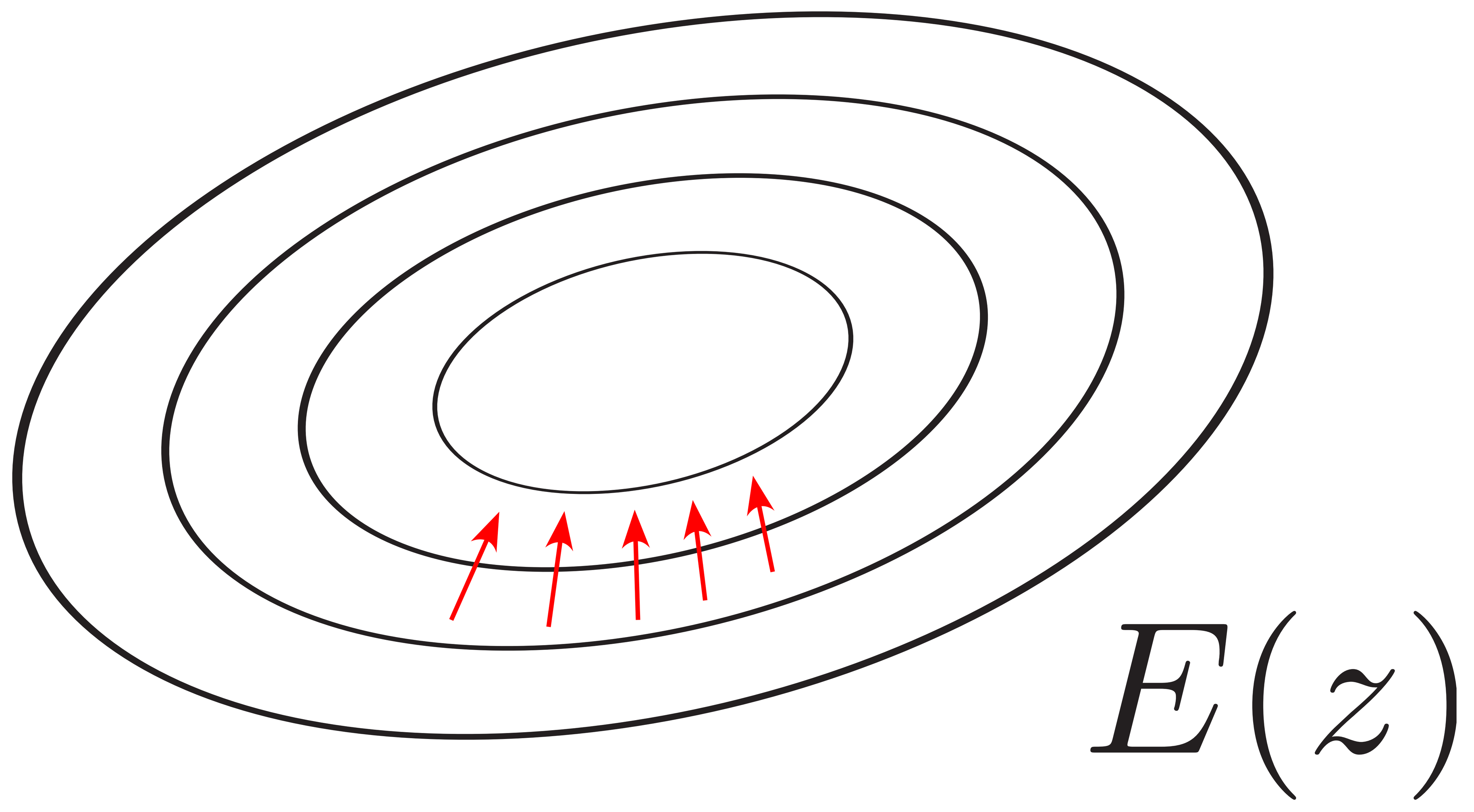
Neural nets as ordinary differential equations

- Neural networks that are intrinsically stable:



Neural nets as ordinary differential equations

- Neural networks that are intrinsically stable:

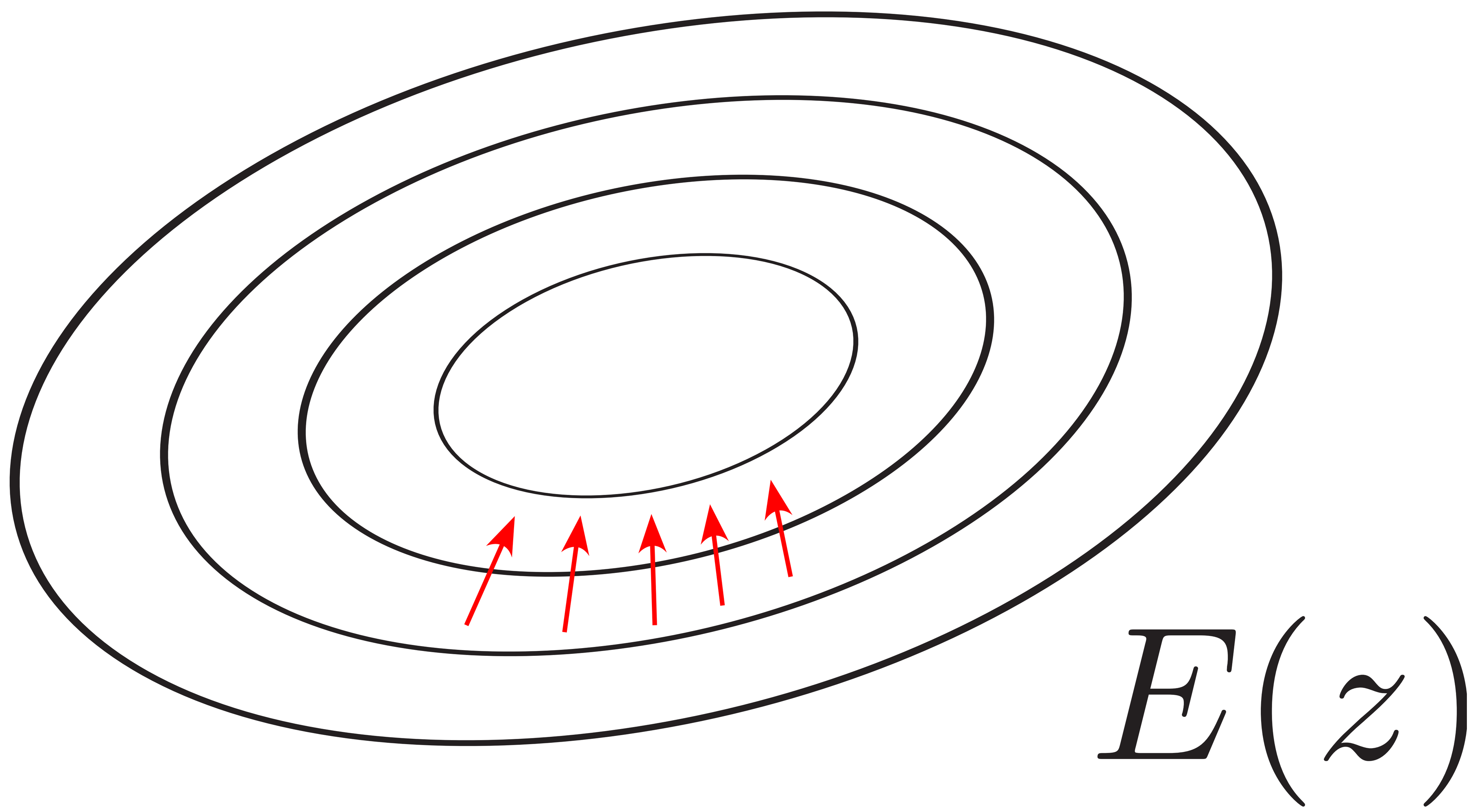


Neural nets as ordinary differential equations

- Neural networks that are intrinsically stable:

Conservation of energy:

$$\dot{z} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \nabla E$$



Neural nets as ordinary differential equations

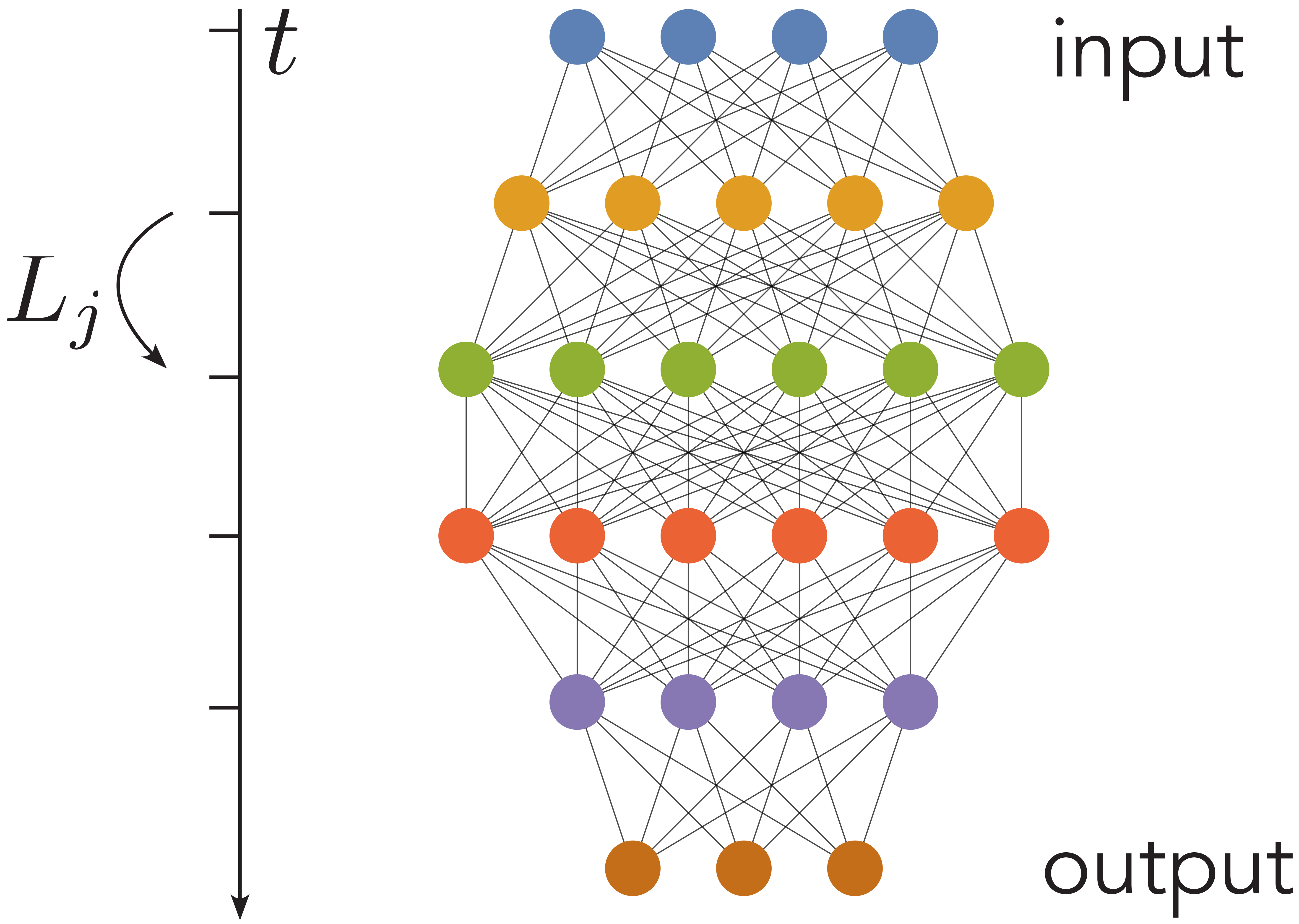
- Neural networks that are intrinsically stable:

$$L_j \, : \, h_{j+1} = h_j + \sigma\left(\begin{pmatrix} 0 & K_j \\ -K_j & 0 \end{pmatrix} h_j + b_j\right)$$

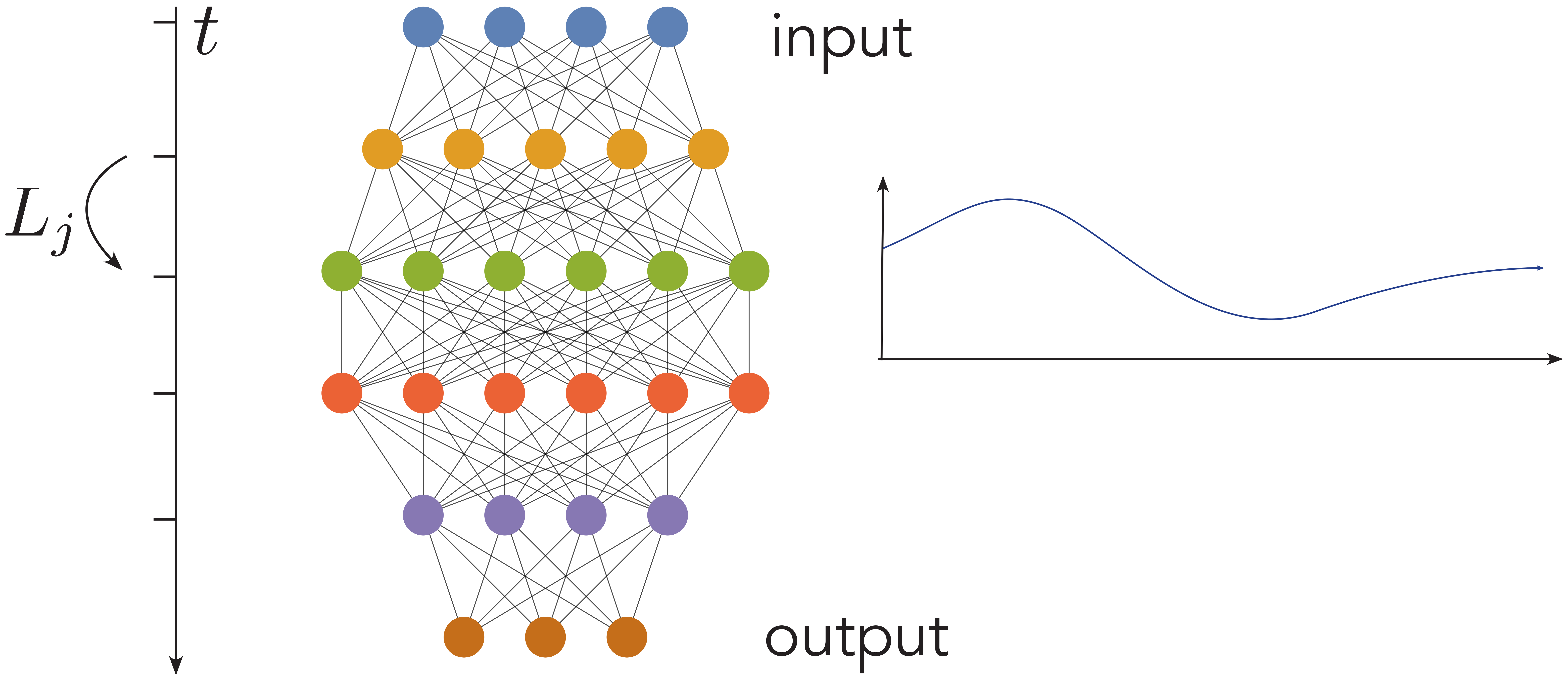
E. Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems, 34(1):014004, dec 2017.

Extension to partial differential equations

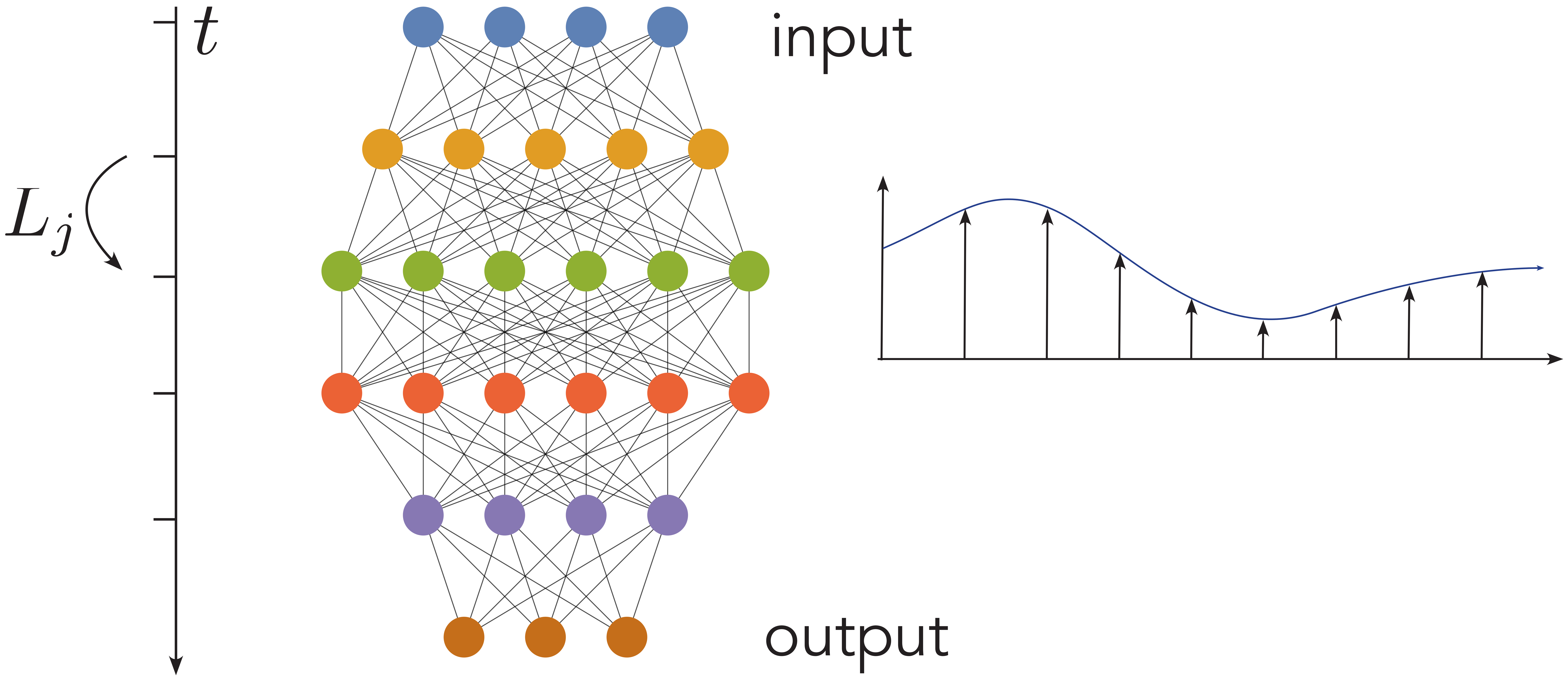
Extension to partial differential equations



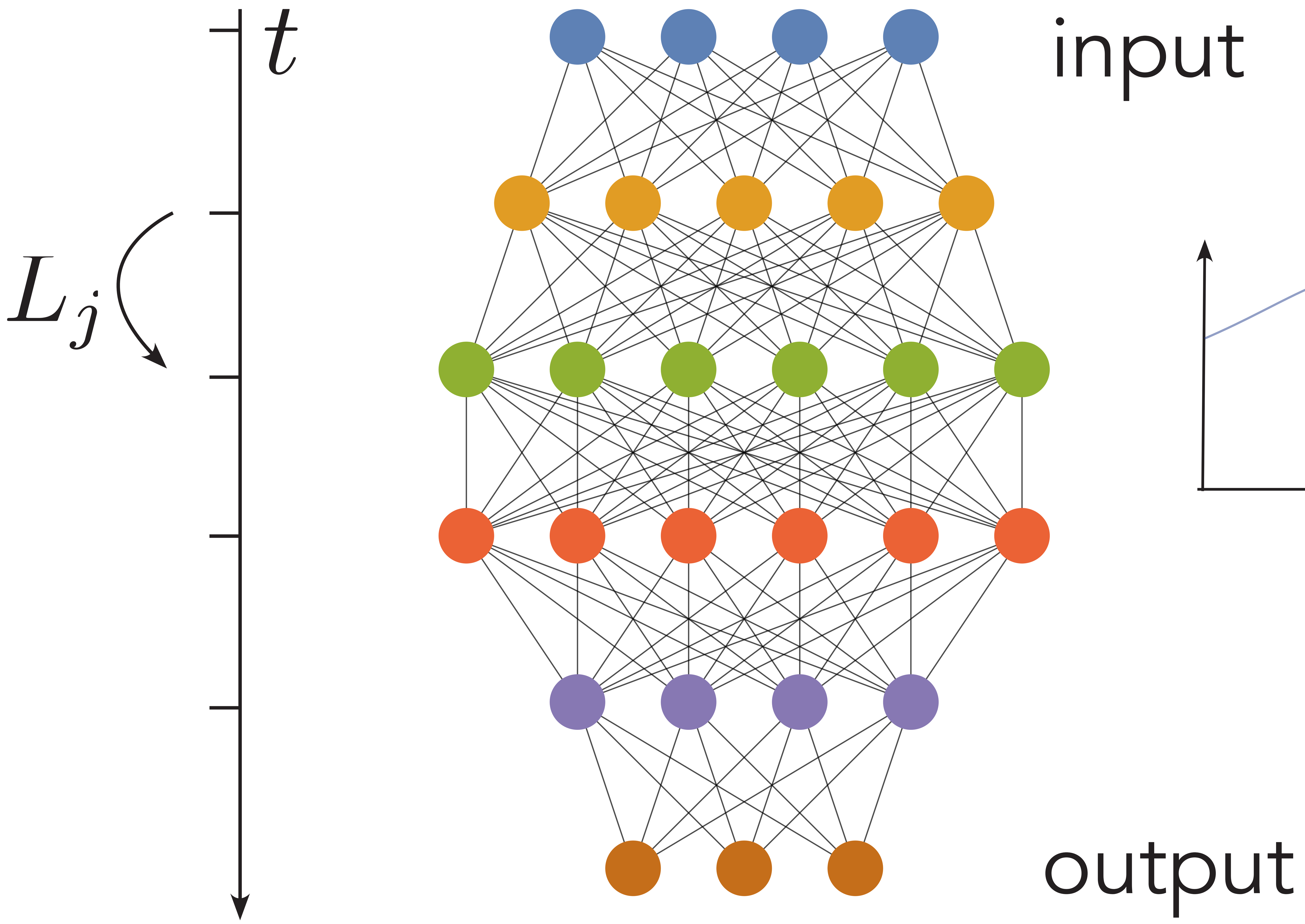
Extension to partial differential equations

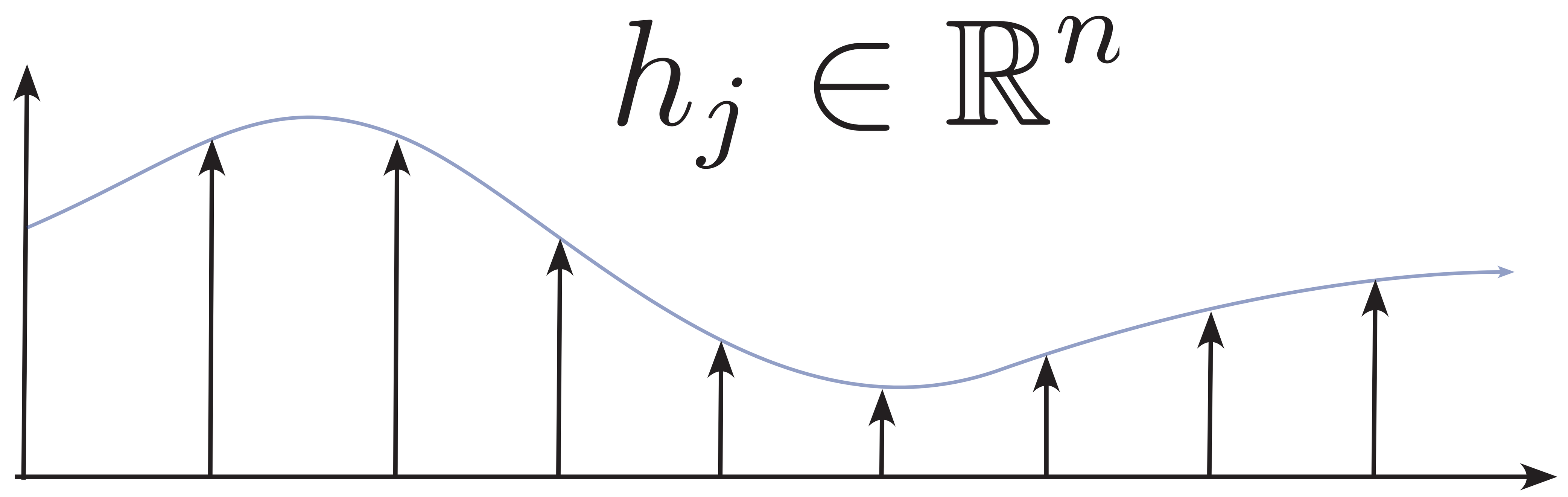


Extension to partial differential equations



Extension to partial differential equations





Extension to partial differential equations

- Neural nets as discretization of partial differential equations
  - Inputs and layers as spatial discretization (finite difference, Galerkin projection, ...)

Extension to partial differential equations

- Neural nets as discretization of partial differential equations
  - Inputs and layers as spatial discretization (finite difference, Galerkin projection, ...)
  - Hyperbolic, parabolic, ... PDEs

L. Ruthotto and E. Haber. Deep neural networks motivated by partial differential equations. Journal of Mathematical Imaging and Vision, 62(3):352–364, 2020.

Extension to partial differential equations

- 3 channel convolution of 1D signal:

$$\begin{pmatrix} — \theta_1 — \\ — \theta_2 — \\ — \theta_3 — \end{pmatrix} \star y \qquad y \in \mathbb{R}^n, \theta_i \in \mathbb{R}^3$$

Extension to partial differential equations

- 3 channel convolution of 1D signal:

$$\underbrace{\begin{pmatrix} \frac{1}{4} & -\frac{1}{2h} & -\frac{1}{h^2} \\ \frac{1}{2} & 0 & \frac{2}{h^2} \\ \frac{1}{4} & \frac{1}{2h} & -\frac{1}{h^2} \end{pmatrix}}_{\text{Non-singular for } h > 0} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} — \theta_1 — \\ — \theta_2 — \\ — \theta_3 — \end{pmatrix}$$

Extension to partial differential equations

- 3 channel convolution of 1D signal:

$$\begin{pmatrix} \frac{1}{4} & -\frac{1}{2h} & -\frac{1}{h^2} \\ \frac{1}{2} & 0 & \frac{2}{h^2} \\ \frac{1}{4} & \frac{1}{2h} & -\frac{1}{h^2} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} \text{---}\,\theta_1\,\text{---} \\ \text{---}\,\theta_2\,\text{---} \\ \text{---}\,\theta_3\,\text{---} \end{pmatrix}$$

Extension to partial differential equations

- 3 channel convolution of 1D signal:

$$\begin{pmatrix} \boxed{\begin{matrix} \frac{1}{4} \\ \frac{1}{2} \\ \frac{1}{4} \end{matrix}} & \begin{matrix} -\frac{1}{2h} \\ 0 \\ \frac{1}{2h} \end{matrix} & \begin{matrix} -\frac{1}{h^2} \\ \frac{2}{h^2} \\ -\frac{1}{h^2} \end{matrix} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} — \theta_1 — \\ — \theta_2 — \\ — \theta_3 — \end{pmatrix}$$

$$\color{blue}{1}$$

Extension to partial differential equations

- 3 channel convolution of 1D signal:

$$\begin{pmatrix} \frac{1}{4} & \boxed{\begin{matrix} -\frac{1}{2h} \\ 0 \\ \frac{1}{2h} \end{matrix}} & \begin{matrix} -\frac{1}{h^2} \\ \frac{2}{h^2} \\ -\frac{1}{h^2} \end{matrix} \\ \frac{1}{2} & & \\ \frac{1}{4} & & \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} - \theta_1 - \\ - \theta_2 - \\ - \theta_3 - \end{pmatrix}$$

$$\frac{\partial}{\partial x}$$

Extension to partial differential equations

- 3 channel convolution of 1D signal:

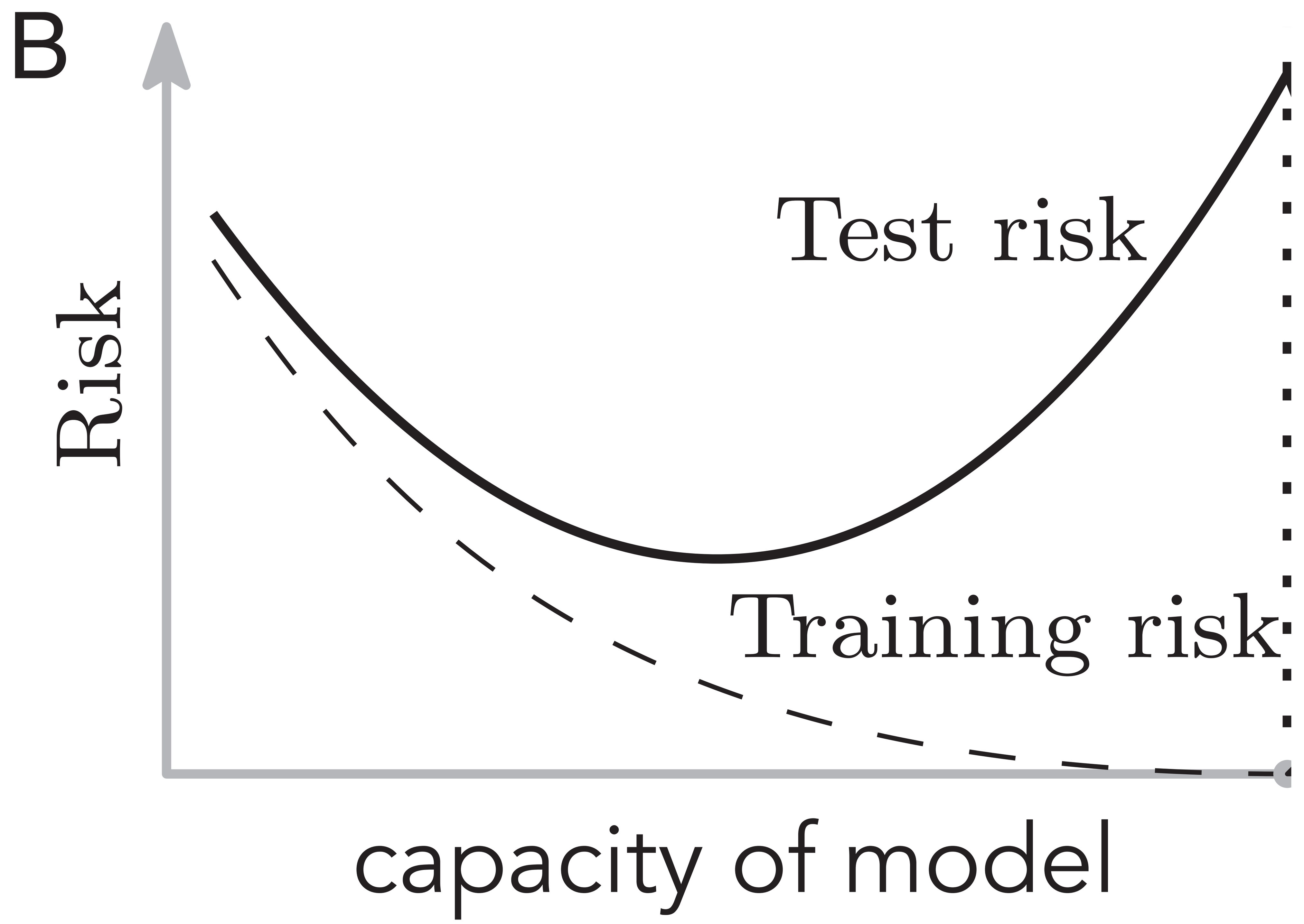
$$\begin{pmatrix} \frac{1}{4} & -\frac{1}{2h} & \boxed{\begin{matrix} -\frac{1}{h^2} \\ \frac{2}{h^2} \\ -\frac{1}{h^2} \end{matrix}} \\ \frac{1}{2} & 0 & \\ \frac{1}{4} & \frac{1}{2h} & \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} \text{---} \; \theta_1 \; \text{---} \\ \text{---} \; \theta_2 \; \text{---} \\ \text{---} \; \theta_3 \; \text{---} \end{pmatrix}$$

$$\color{blue}{\frac{\partial^2}{\partial^2 x}}$$

Benign overfitting

Benign overfitting

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. PNAS, 116(32):15849–15854, 2019.



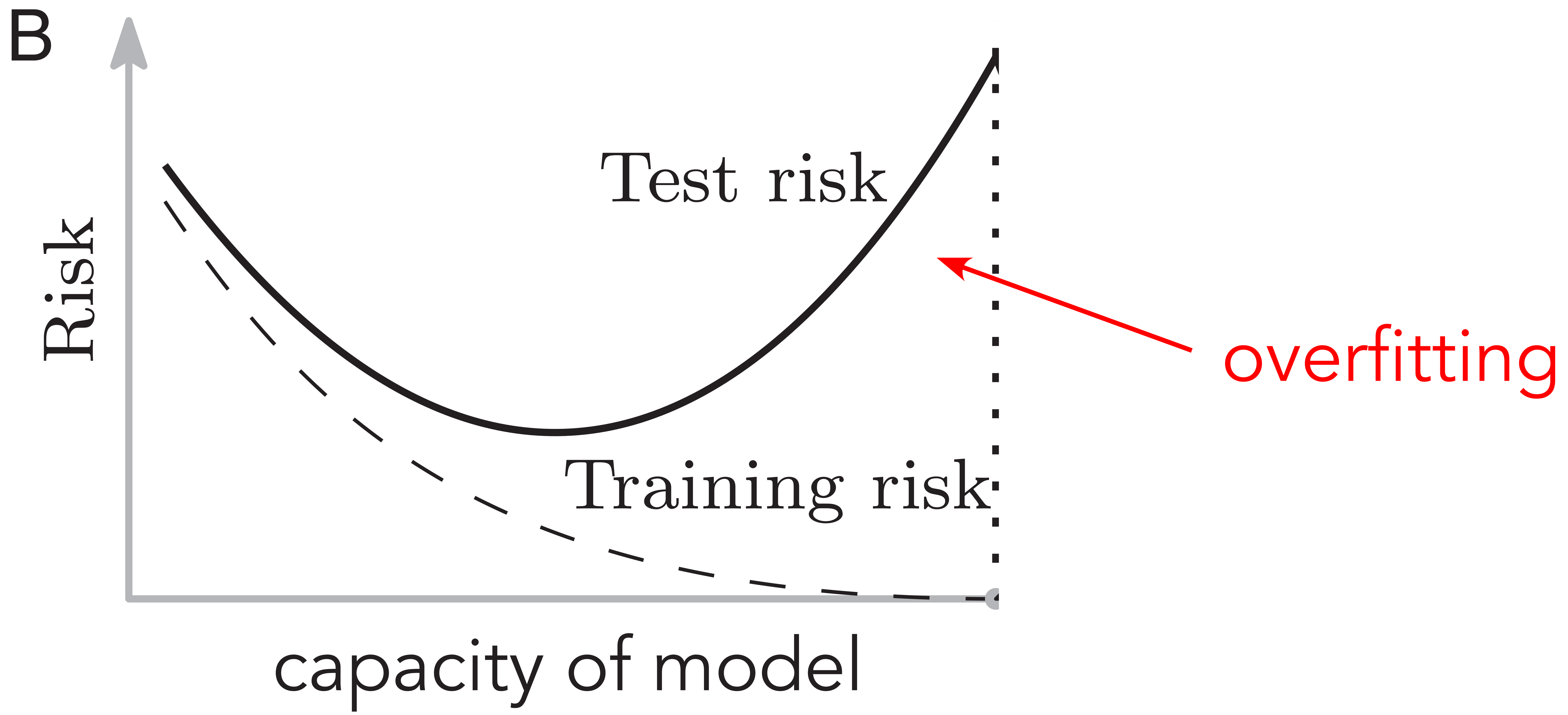
M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. In J. Dy and A. Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 541–549. PMLR, 10–15 Jul 2018.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. Proceedings of the National Academy of Sciences, 116(32):15849–15854, 2019.

P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. Benign overfitting in linear regression. Proceedings of the National Academy of Sciences, 2020.

Benign overfitting

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. PNAS, 116(32):15849–15854, 2019.

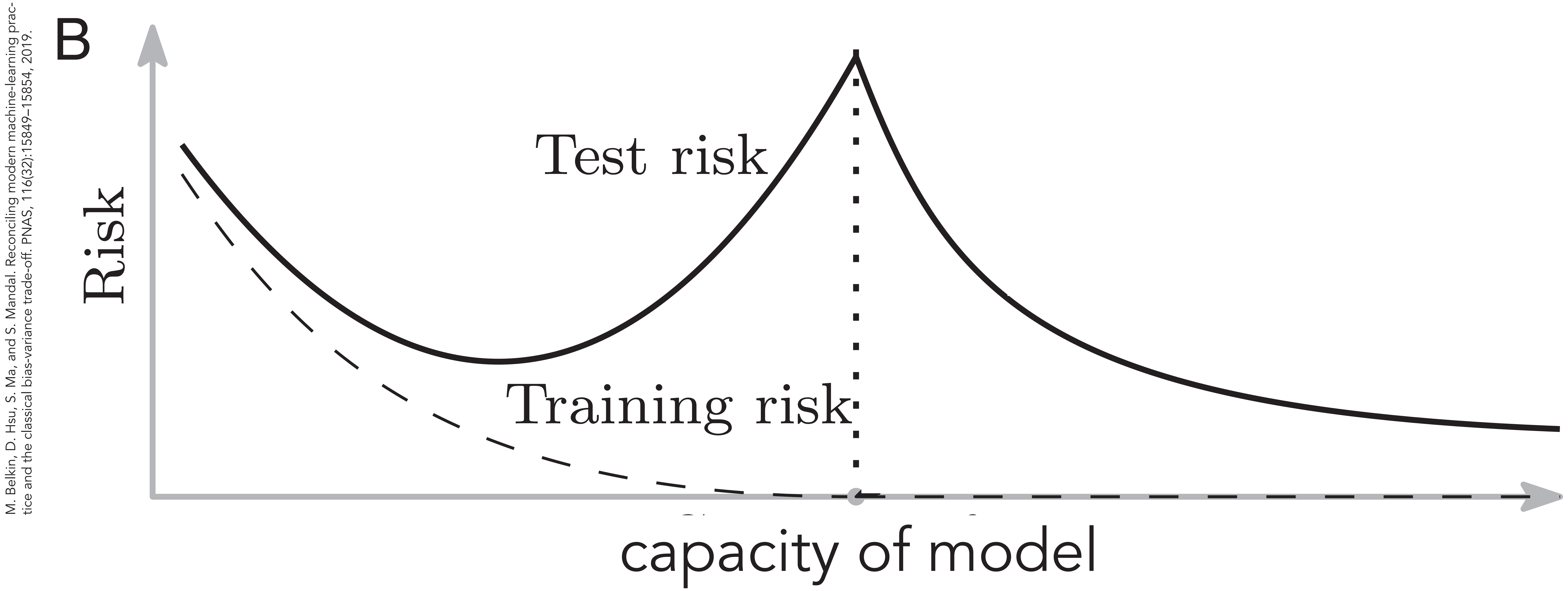


M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. In J. Dy and A. Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 541–549. PMLR, 10–15 Jul 2018.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. Proceedings of the National Academy of Sciences, 116(32):15849–15854, 2019.

P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. Benign overfitting in linear regression. Proceedings of the National Academy of Sciences, 2020.

Benign overfitting

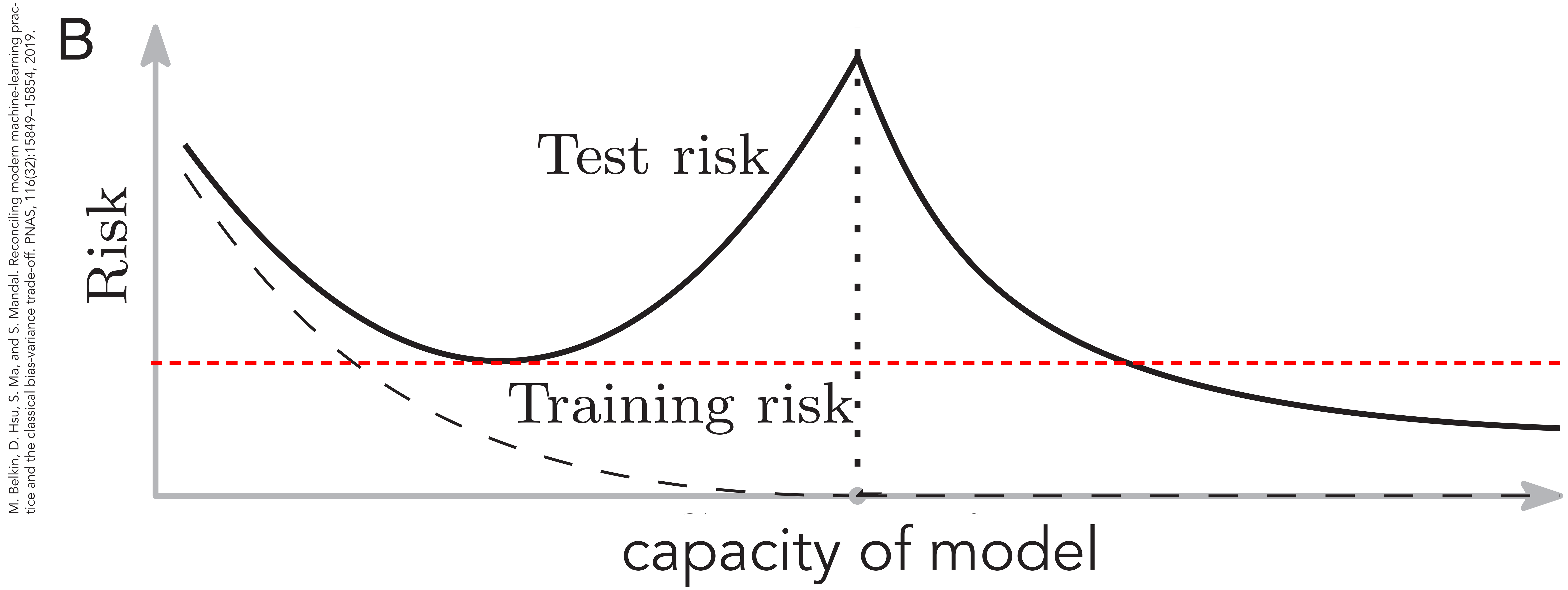


M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. In J. Dy and A. Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 541–549. PMLR, 10–15 Jul 2018.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. Proceedings of the National Academy of Sciences, 116(32):15849–15854, 2019.

P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. Benign overfitting in linear regression. Proceedings of the National Academy of Sciences, 2020.

Benign overfitting



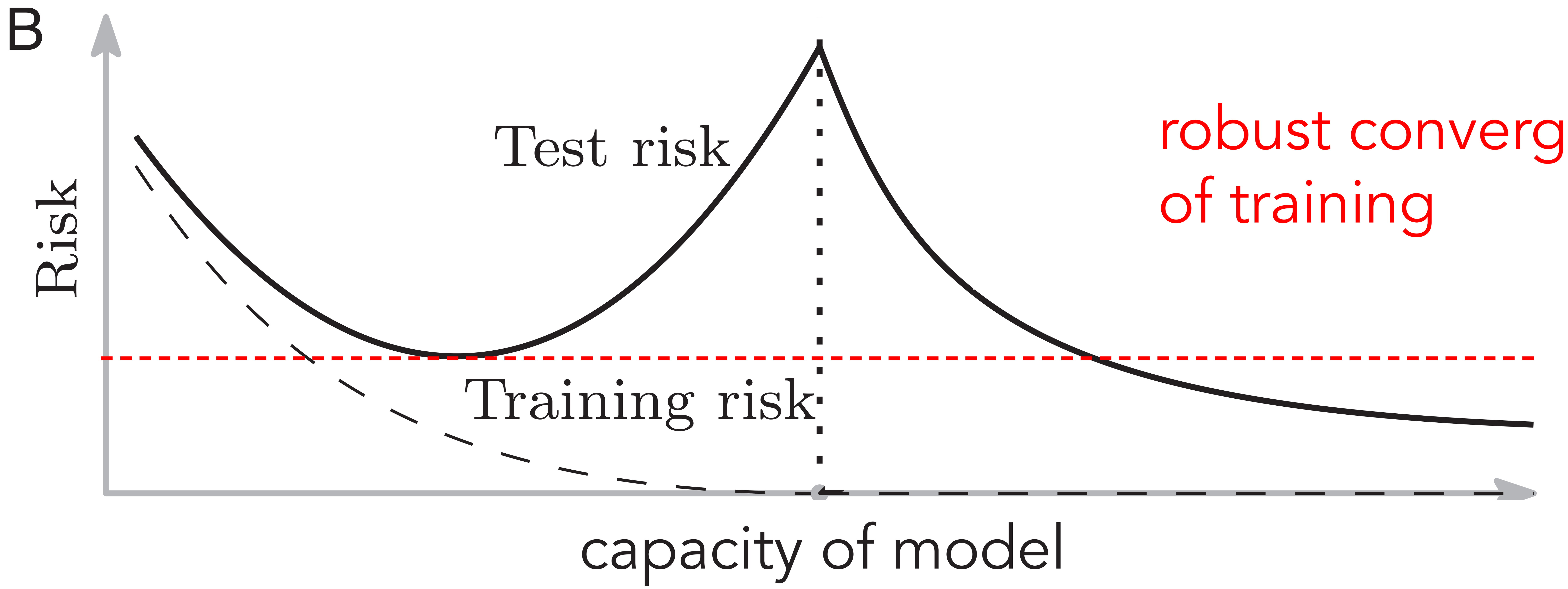
M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. In J. Dy and A. Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 541–549. PMLR, 10–15 Jul 2018.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. Proceedings of the National Academy of Sciences, 116(32):15849–15854, 2019.

P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. Benign overfitting in linear regression. Proceedings of the National Academy of Sciences, 2020.

Benign overfitting

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. PNAS, 116(32):15849–15854, 2019.



robust convergence of training

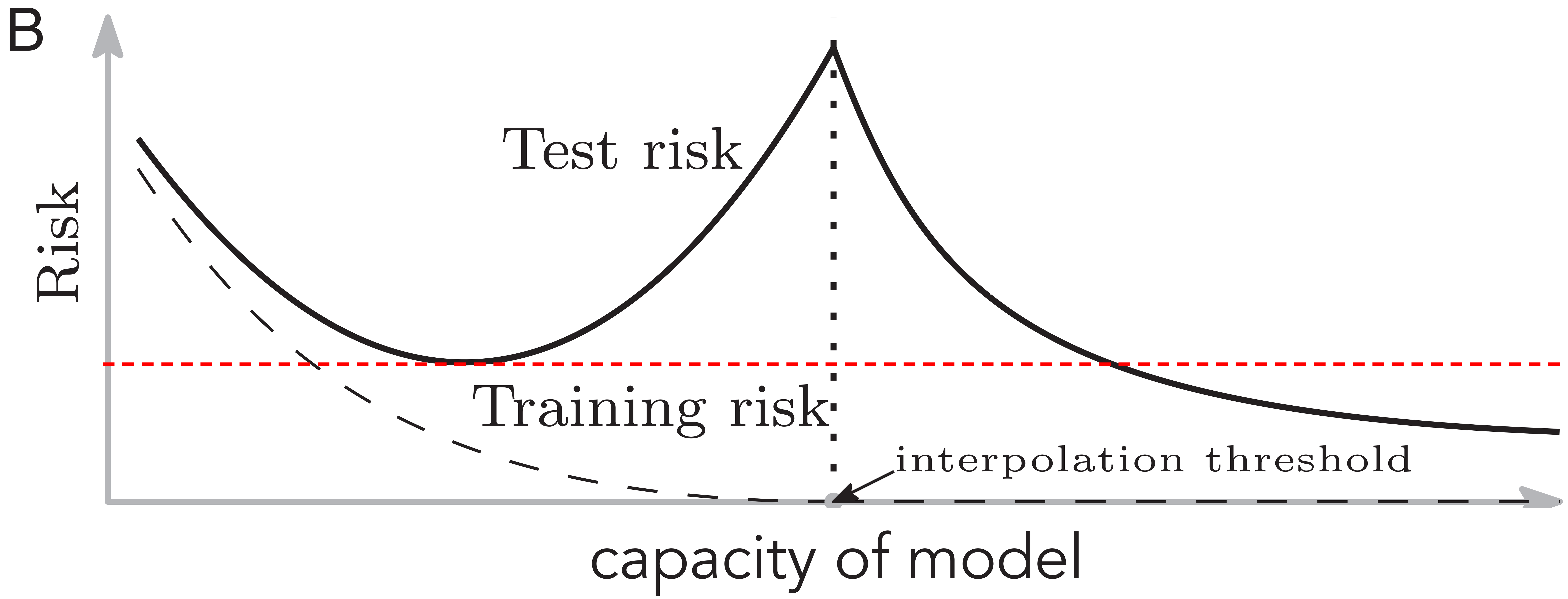
M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. In J. Dy and A. Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 541–549. PMLR, 10–15 Jul 2018.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. Proceedings of the National Academy of Sciences, 116(32):15849–15854, 2019.

P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. Benign overfitting in linear regression. Proceedings of the National Academy of Sciences, 2020.

Benign overfitting

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. PNAS, 116(32):15849–15854, 2019.

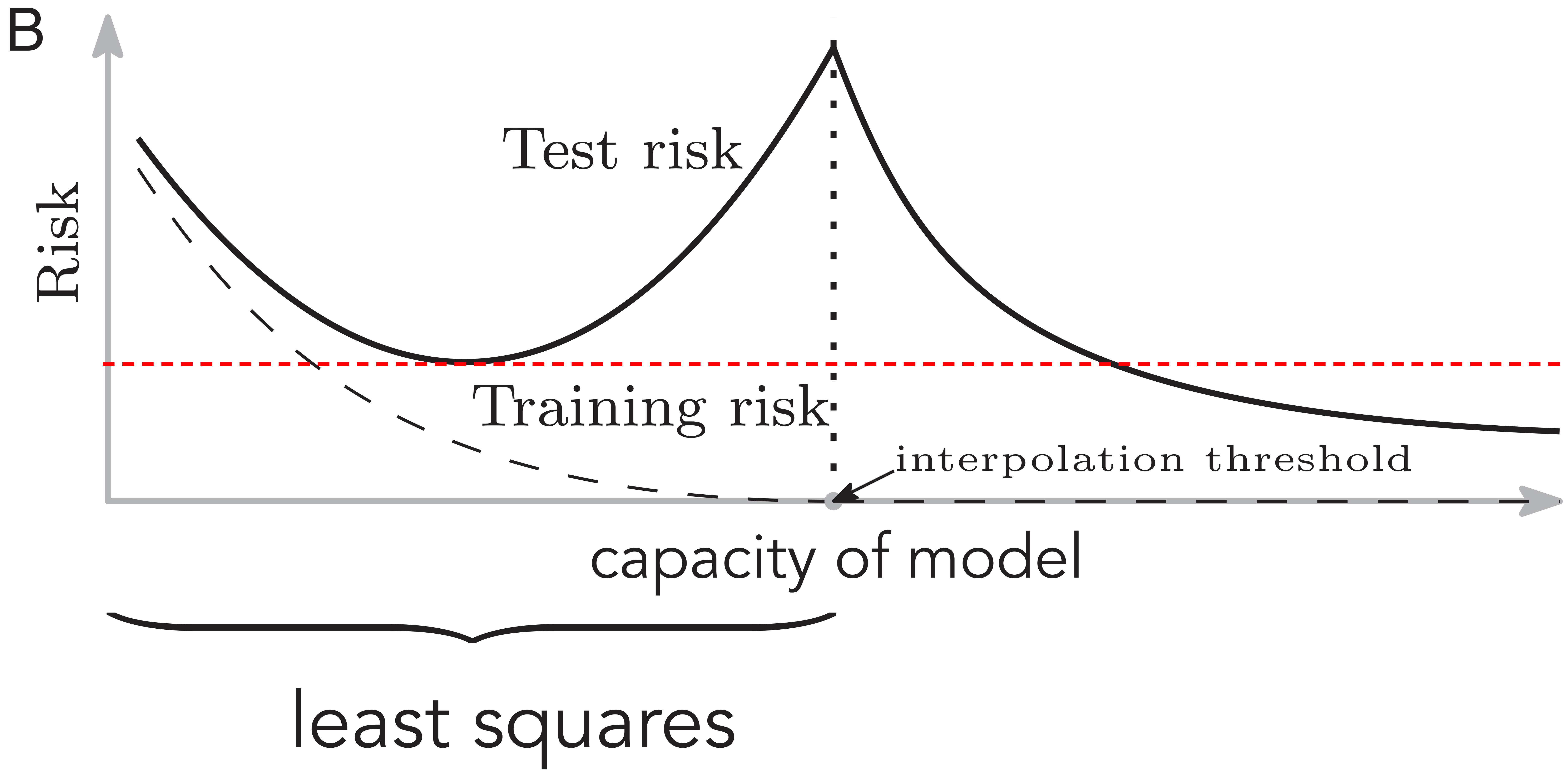


M. Belkin, S. Ma, and S. Mandal. To understand deep learning we need to understand kernel learning. In J. Dy and A. Krause, editors, Proceedings of the 35th International Conference on Machine Learning, volume 80 of Proceedings of Machine Learning Research, pages 541–549. PMLR, 10–15 Jul 2018.

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. Proceedings of the National Academy of Sciences, 116(32):15849–15854, 2019.

P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. Benign overfitting in linear regression. Proceedings of the National Academy of Sciences, 2020.

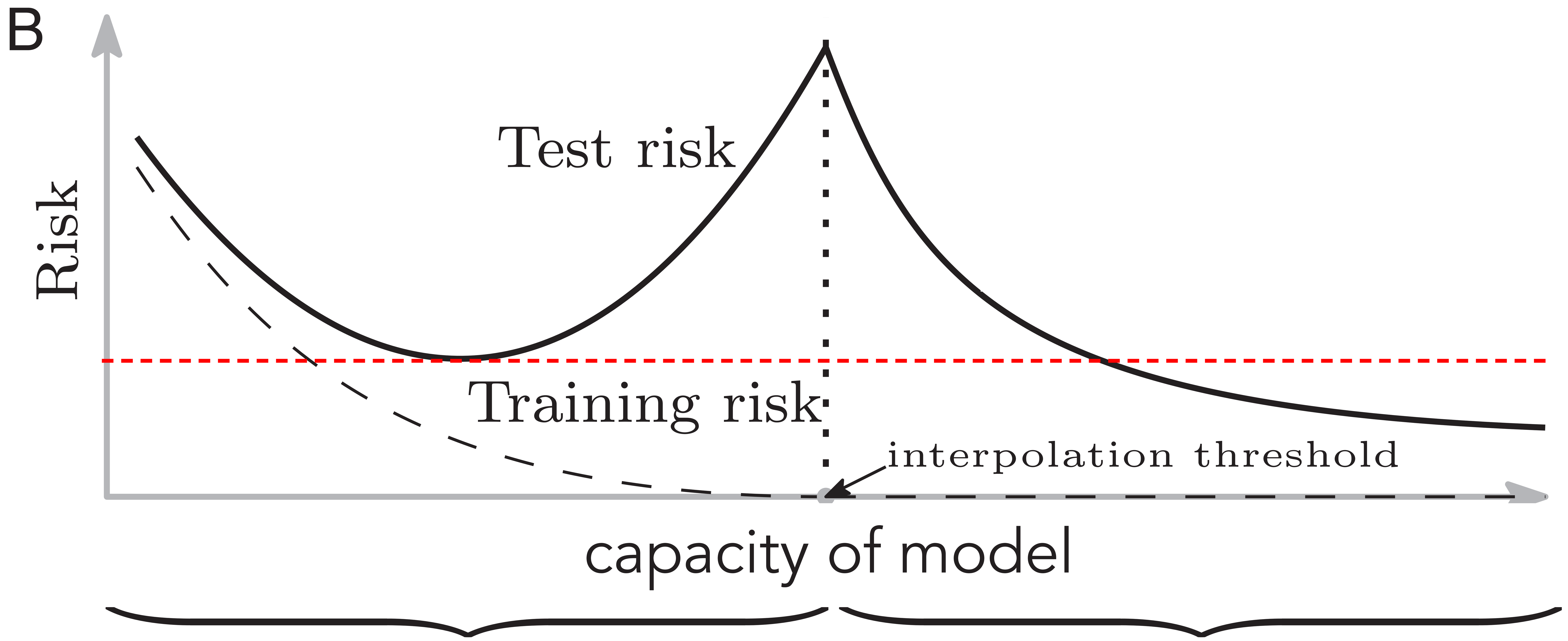
Benign overfitting



M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. PNAS, 116(32):15849–15854, 2019.

Benign overfitting

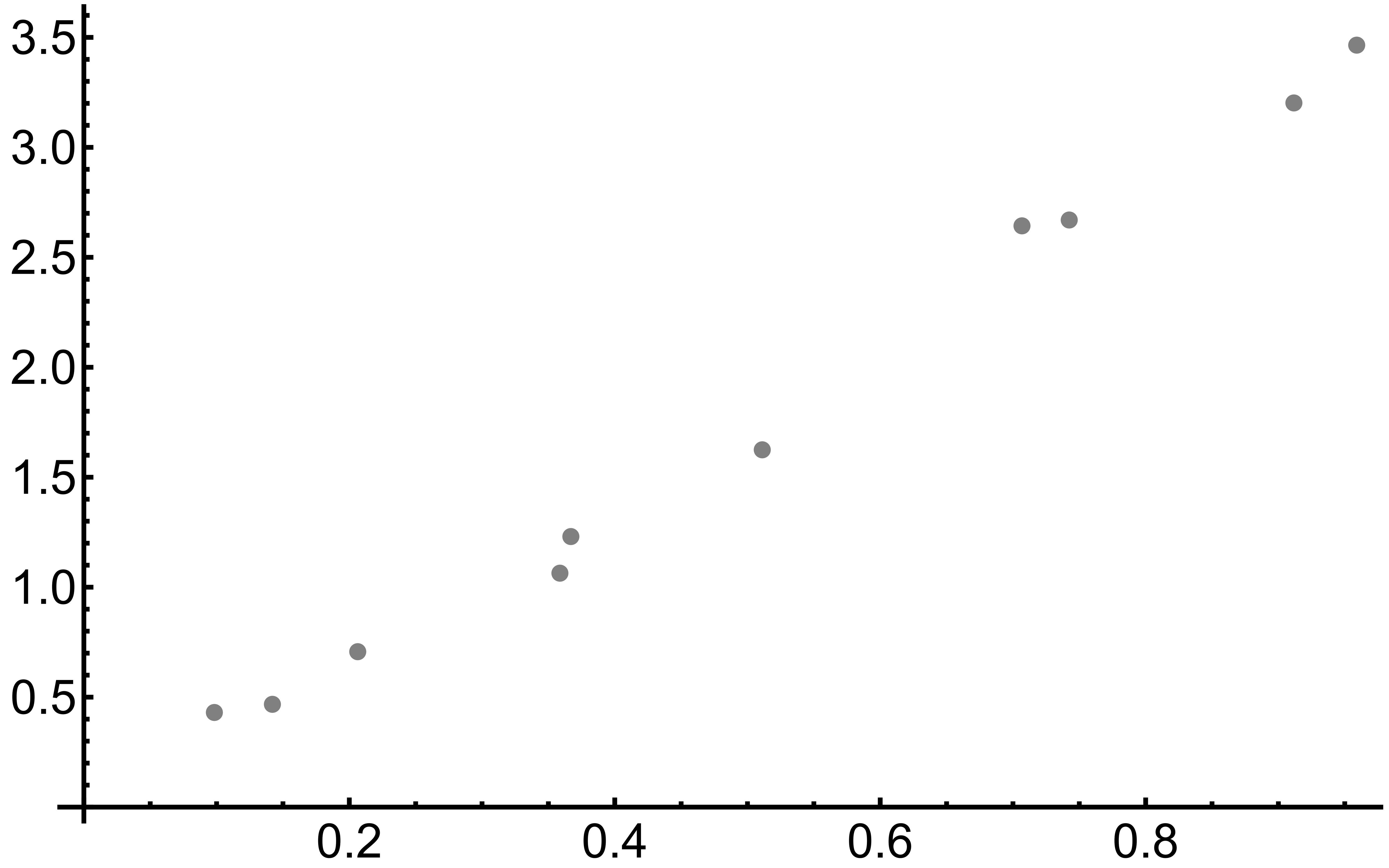
M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. PNAS, 116(32):15849–15854, 2019.



least squares

under-determined regime

Benign overfitting



Benign overfitting

- Model:

$$y = a_1 x + a_0 \qquad \theta = \{a_0, a_1\}$$

Benign overfitting

- Model:

$$y = a_1 x + a_0 \qquad \underbrace{\theta = \{a_0, a_1\}}_{M = 2}$$

Benign overfitting

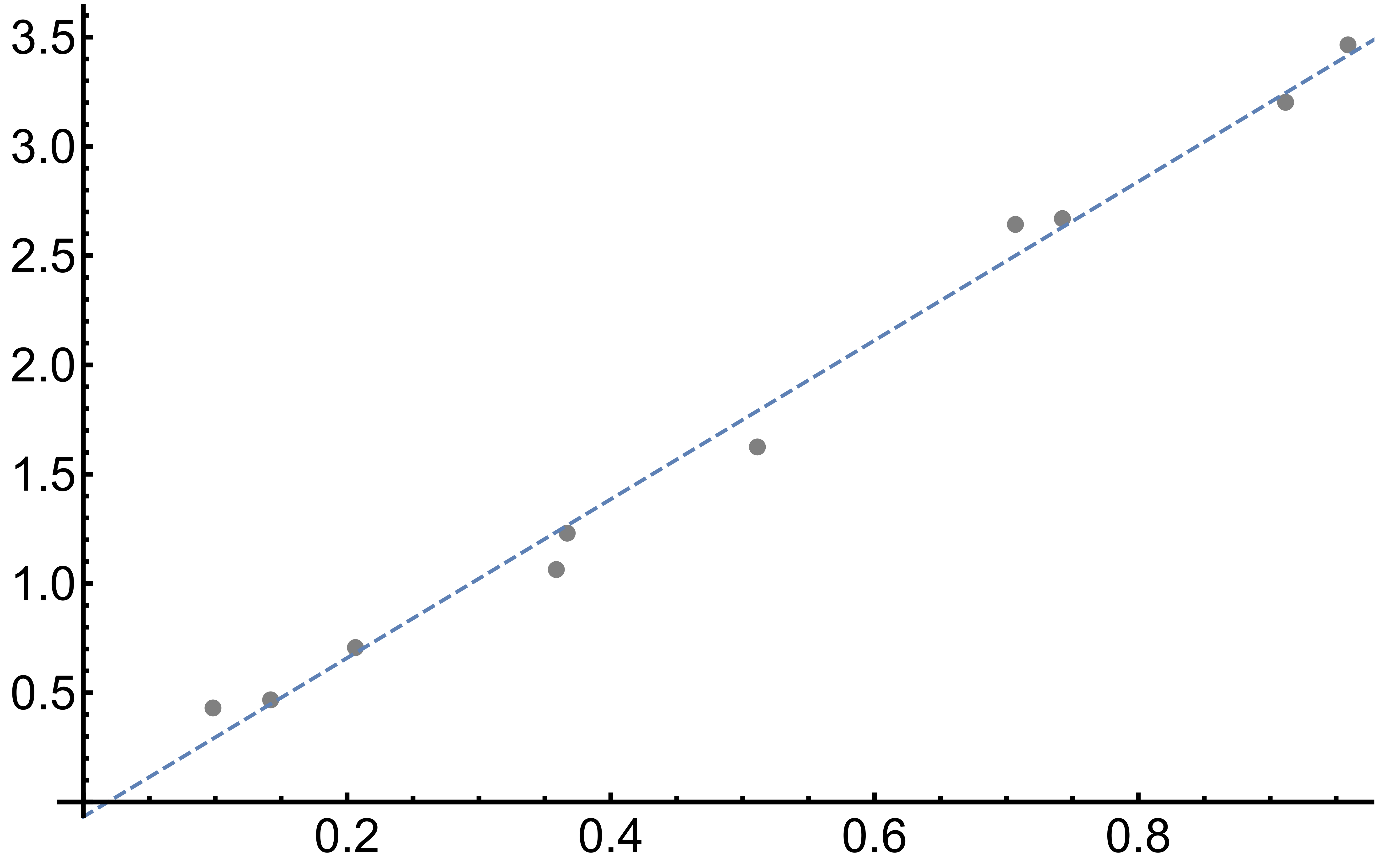
- Model:

$$y = a_1 x + a_0 \qquad\qquad \theta = \{a_0, a_1\}$$

- Loss function:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \left\| \tilde{y}_i - y(x_i) \right\|^2$$

Benign overfitting



Benign overfitting

- Model:

$$y = \sum_{n=0}^{M} a_n x^n \qquad \theta = \{a_n\}_{n=0}^{M}$$

Benign overfitting

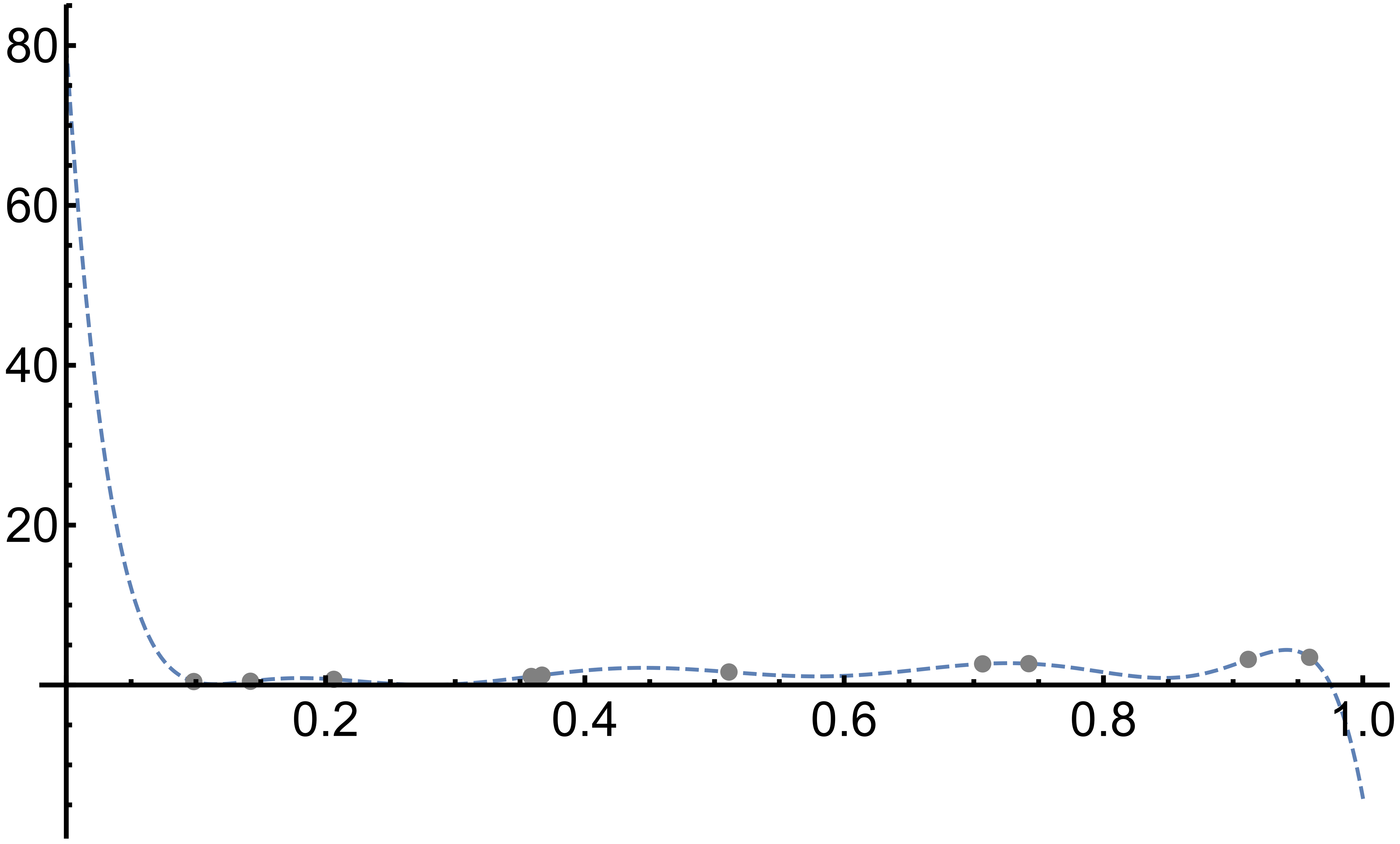
- Model:

$$y = \sum_{n=0}^{M} a_n x^n \qquad\qquad \theta = \{a_n\}_{n=0}^{M}$$

- Loss function:

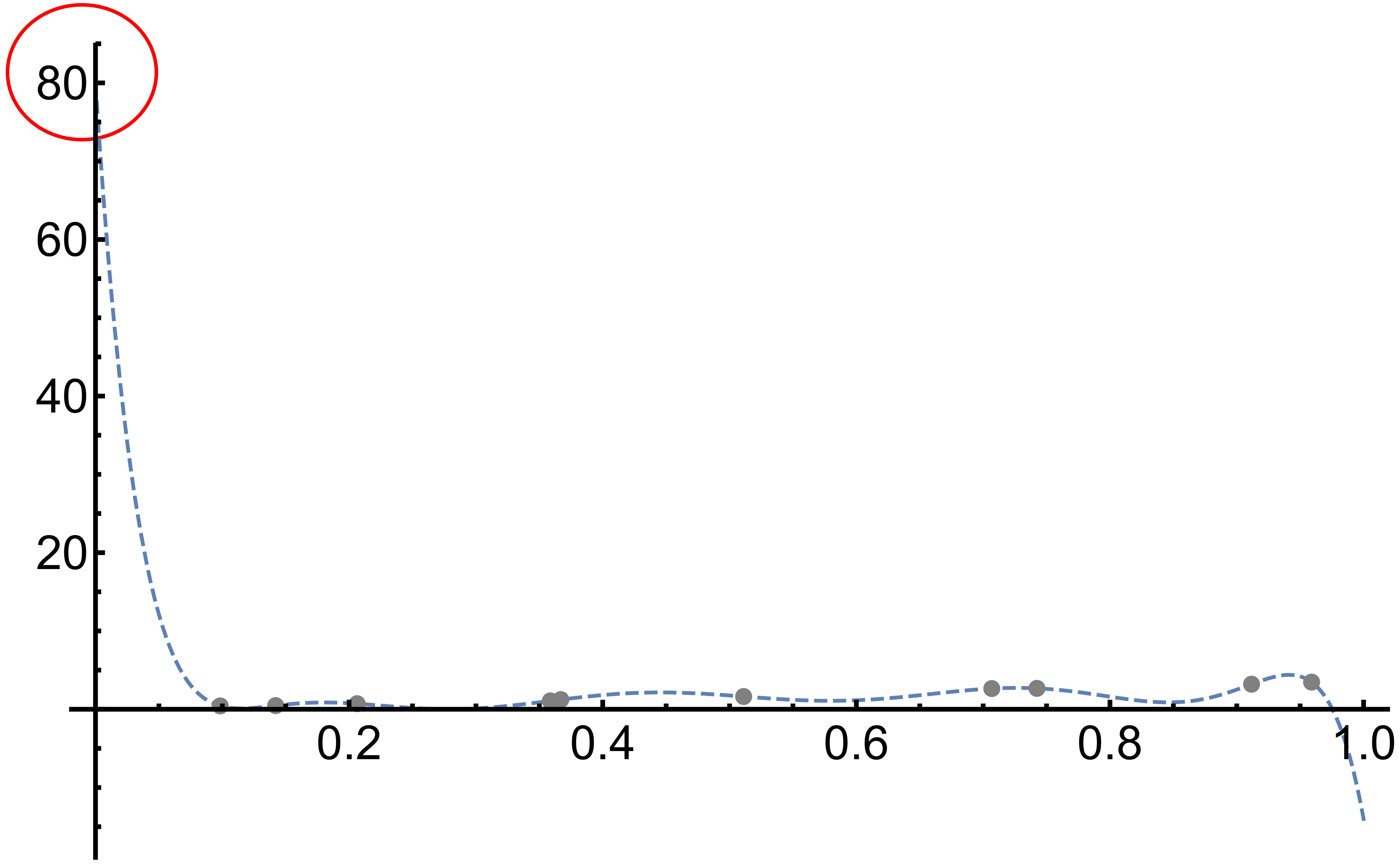
$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \left\| \tilde{y}_i - y(x_i) \right\|^2$$

Benign overfitting



$$M = 10$$

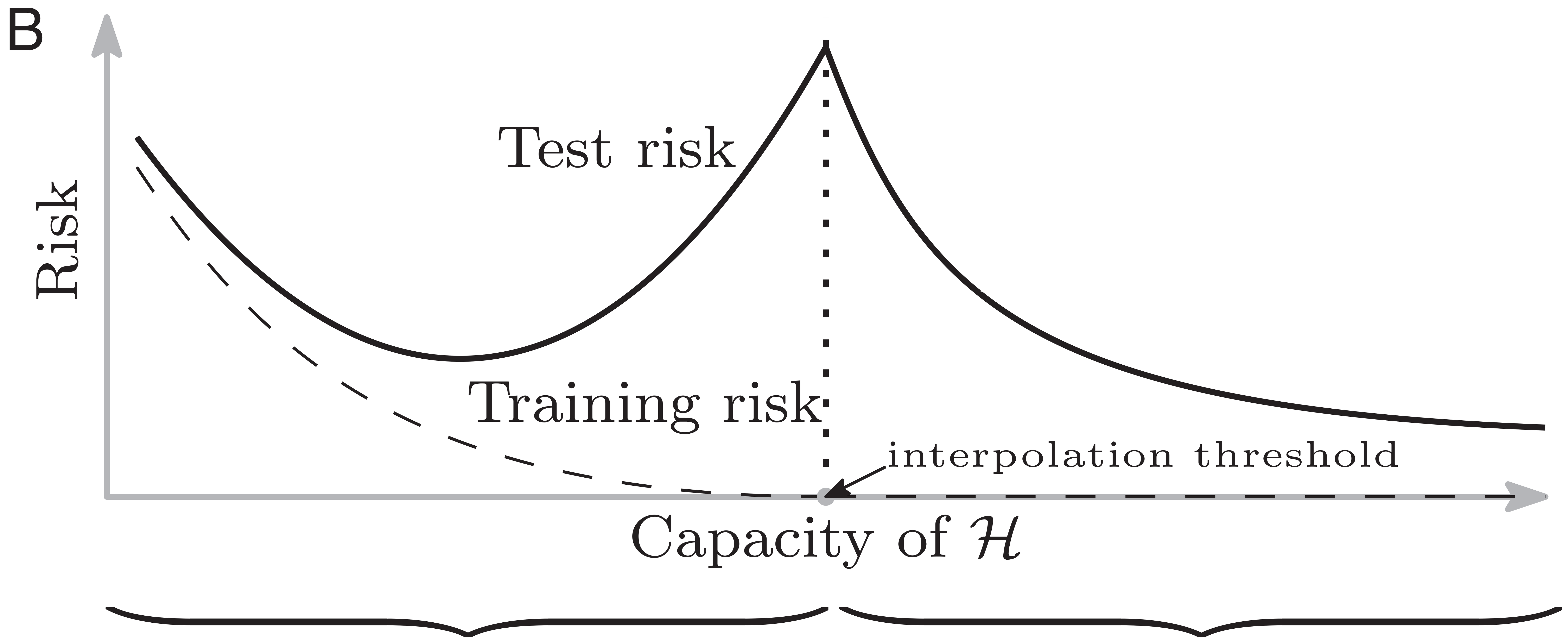
Benign overfitting



$$M = 10$$

Benign overfitting

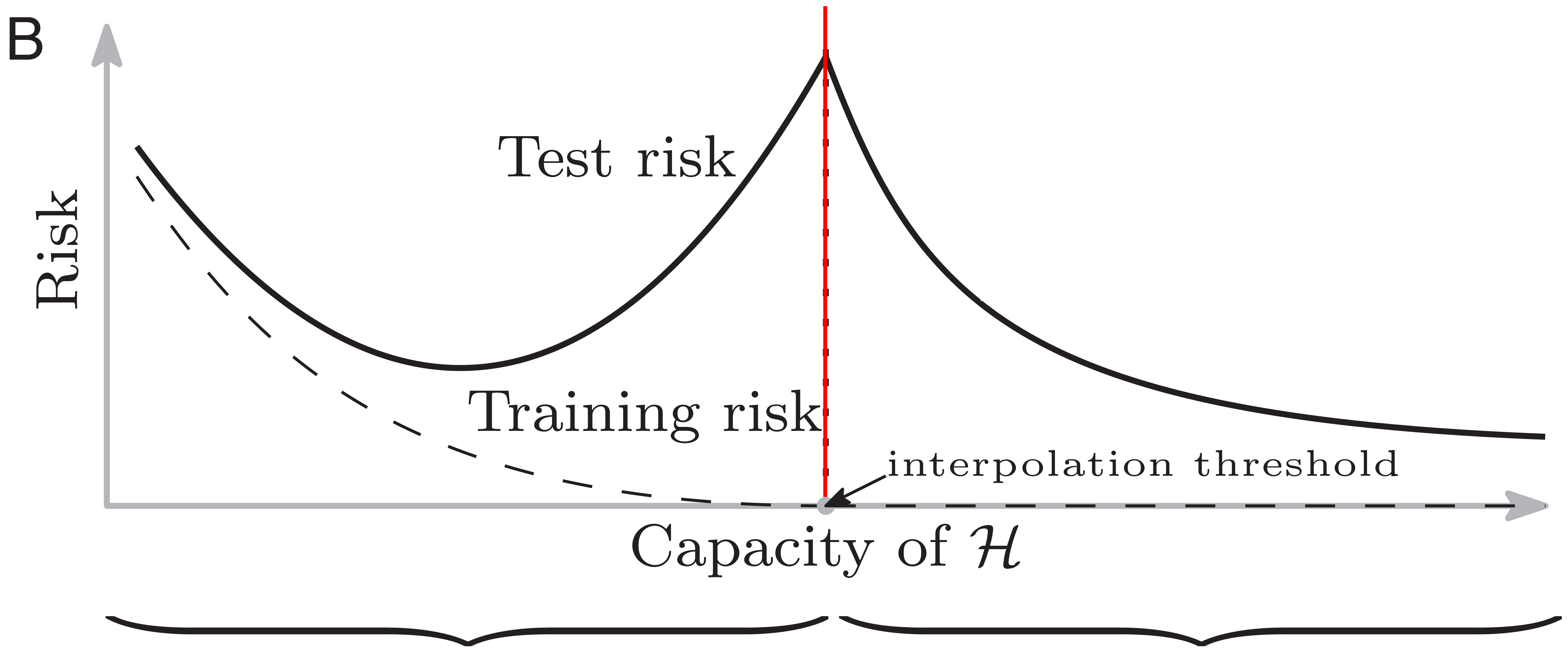
M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. PNAS, 116(32):15849–15854, 2019.



least squares under-determined

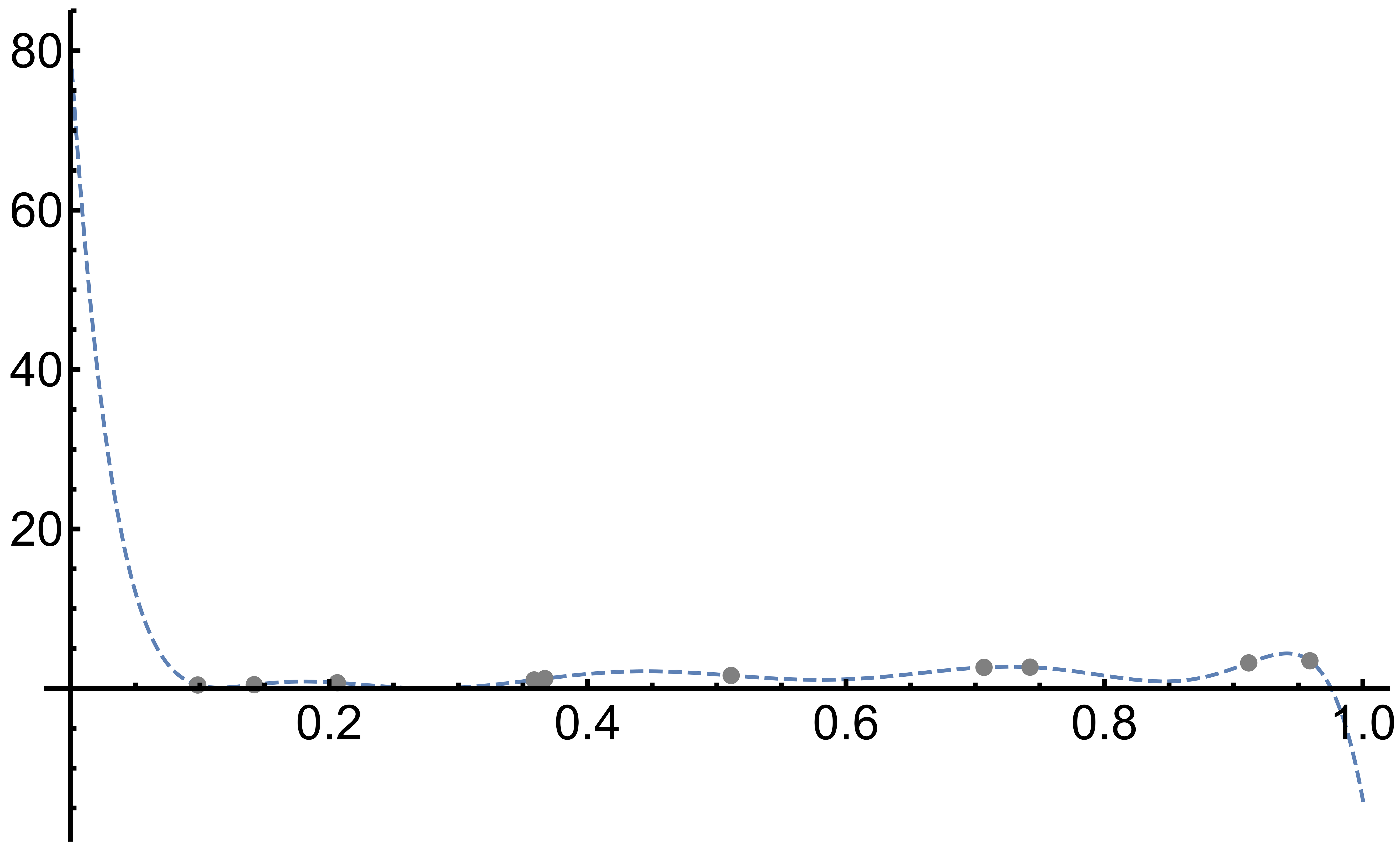
Benign overfitting

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. PNAS, 116(32):15849–15854, 2019.



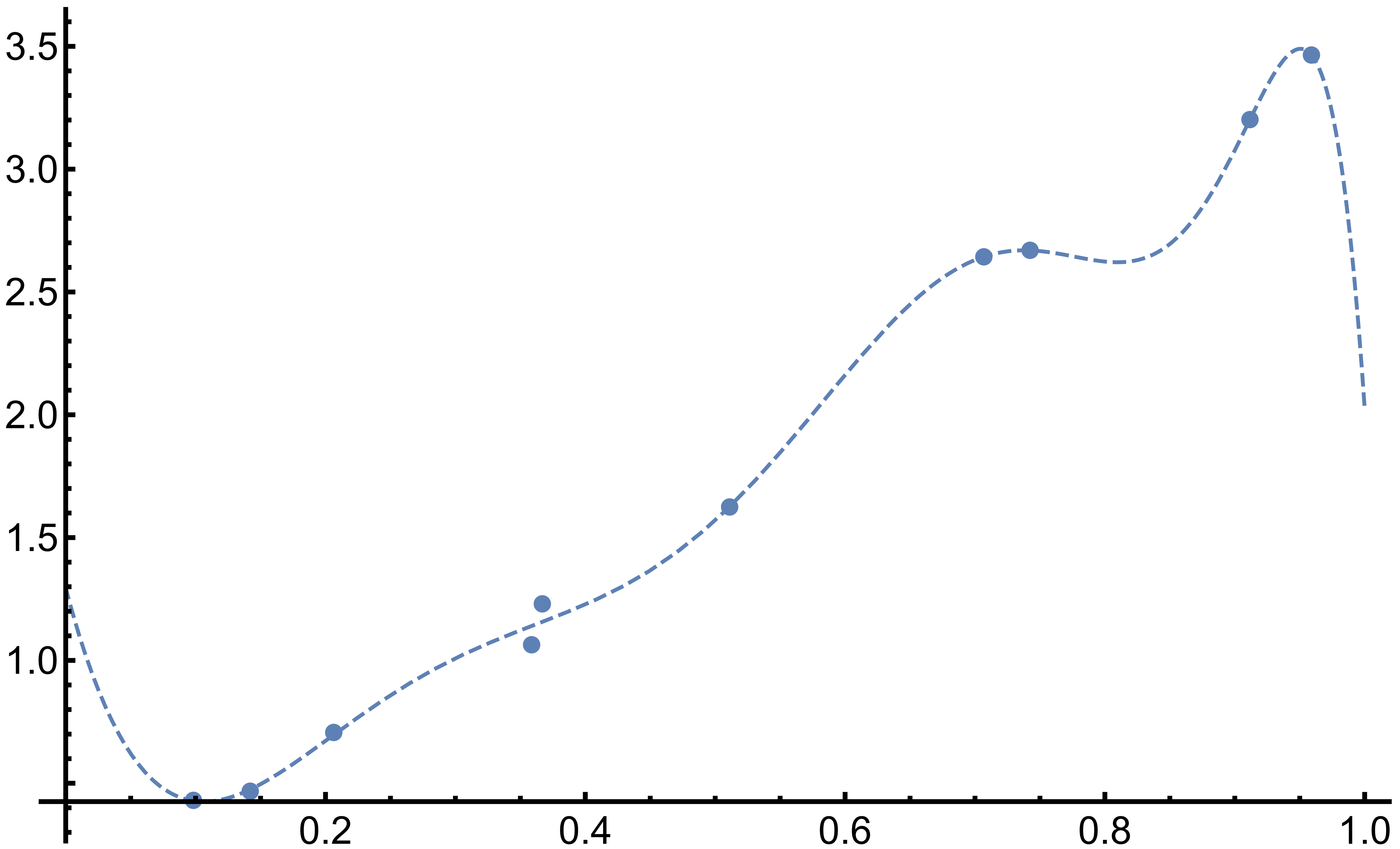
least squares under-determined

Benign overfitting



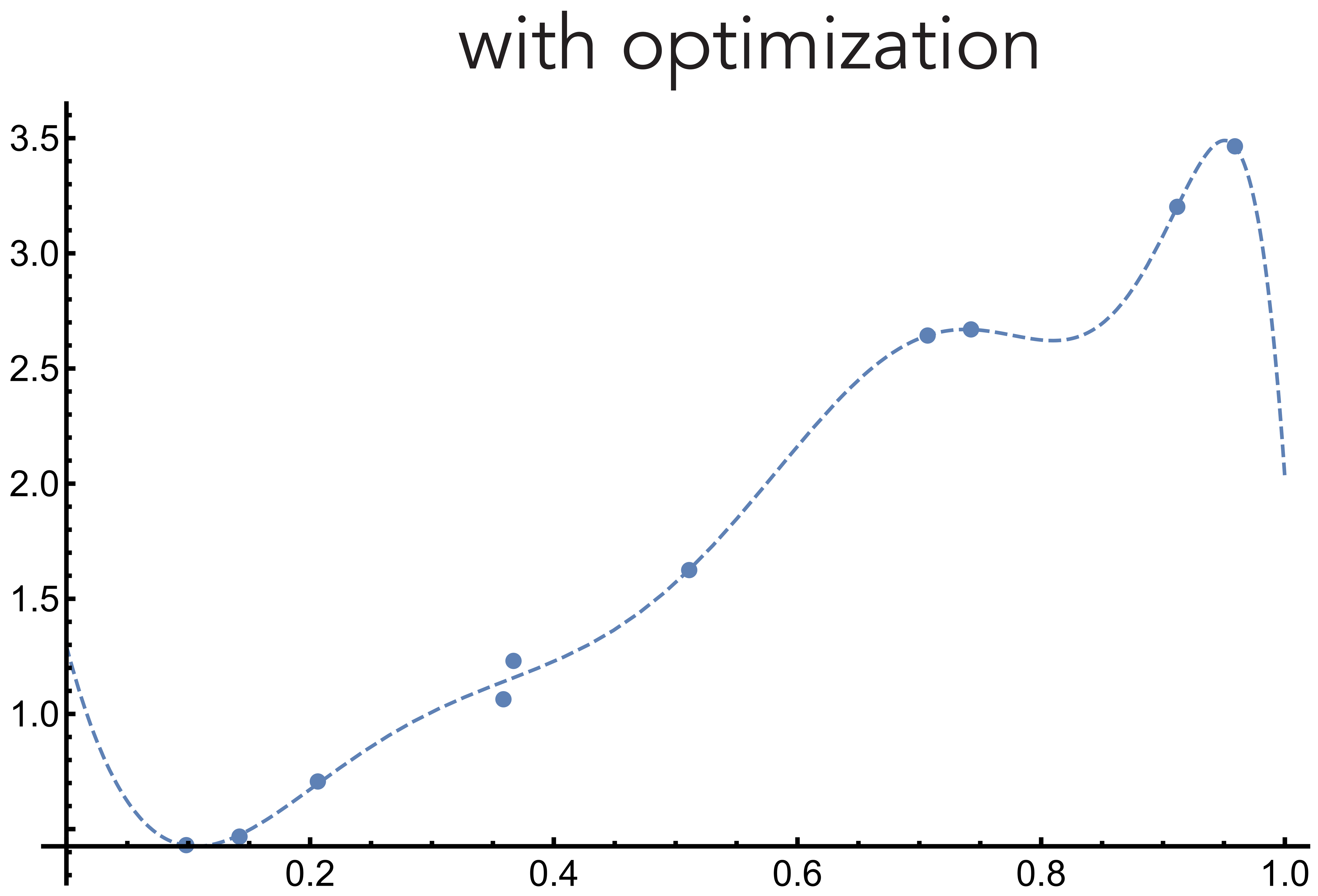
$$M = 10$$

Benign overfitting



$$M = 20$$

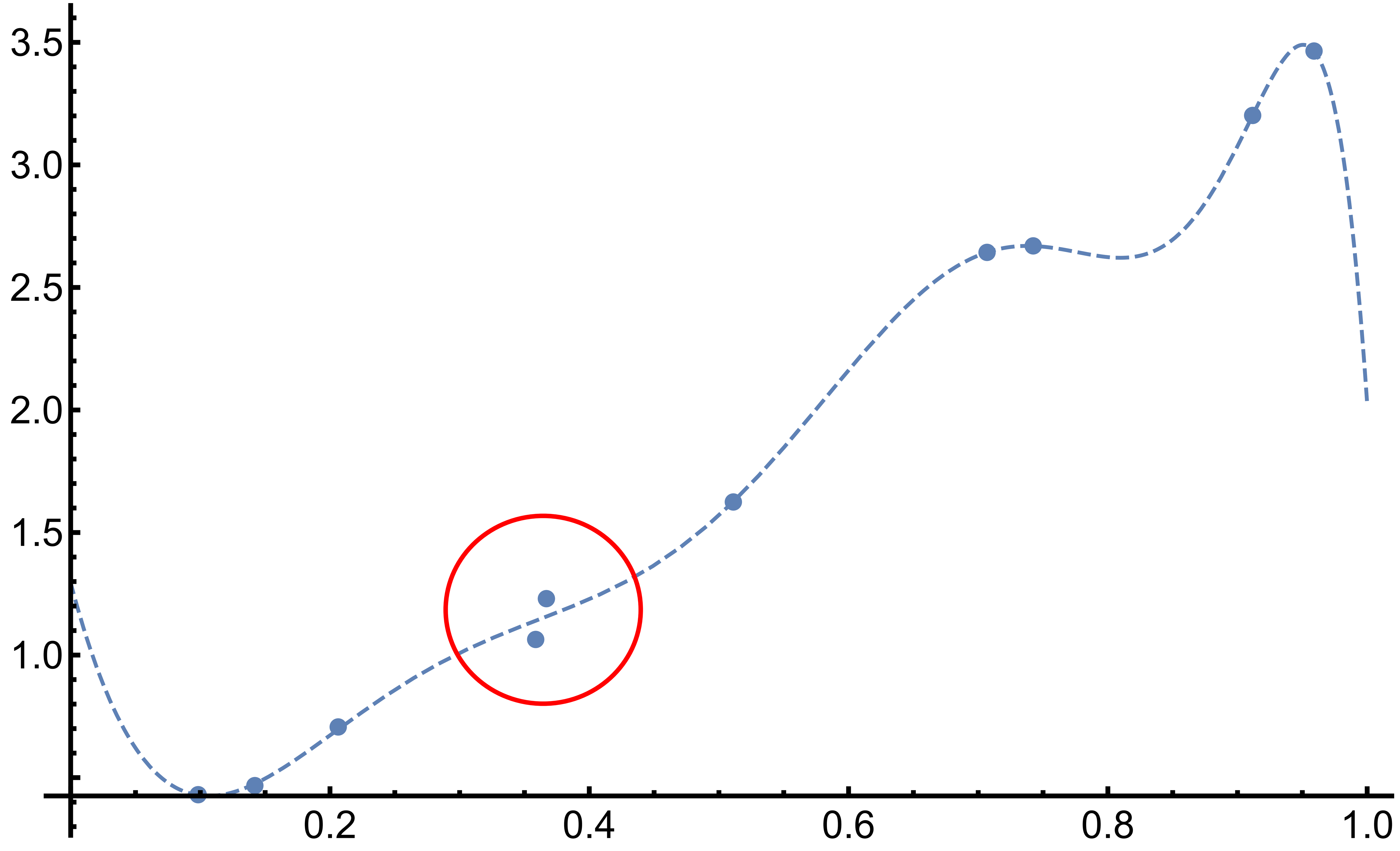
Benign overfitting



$$M = 20$$

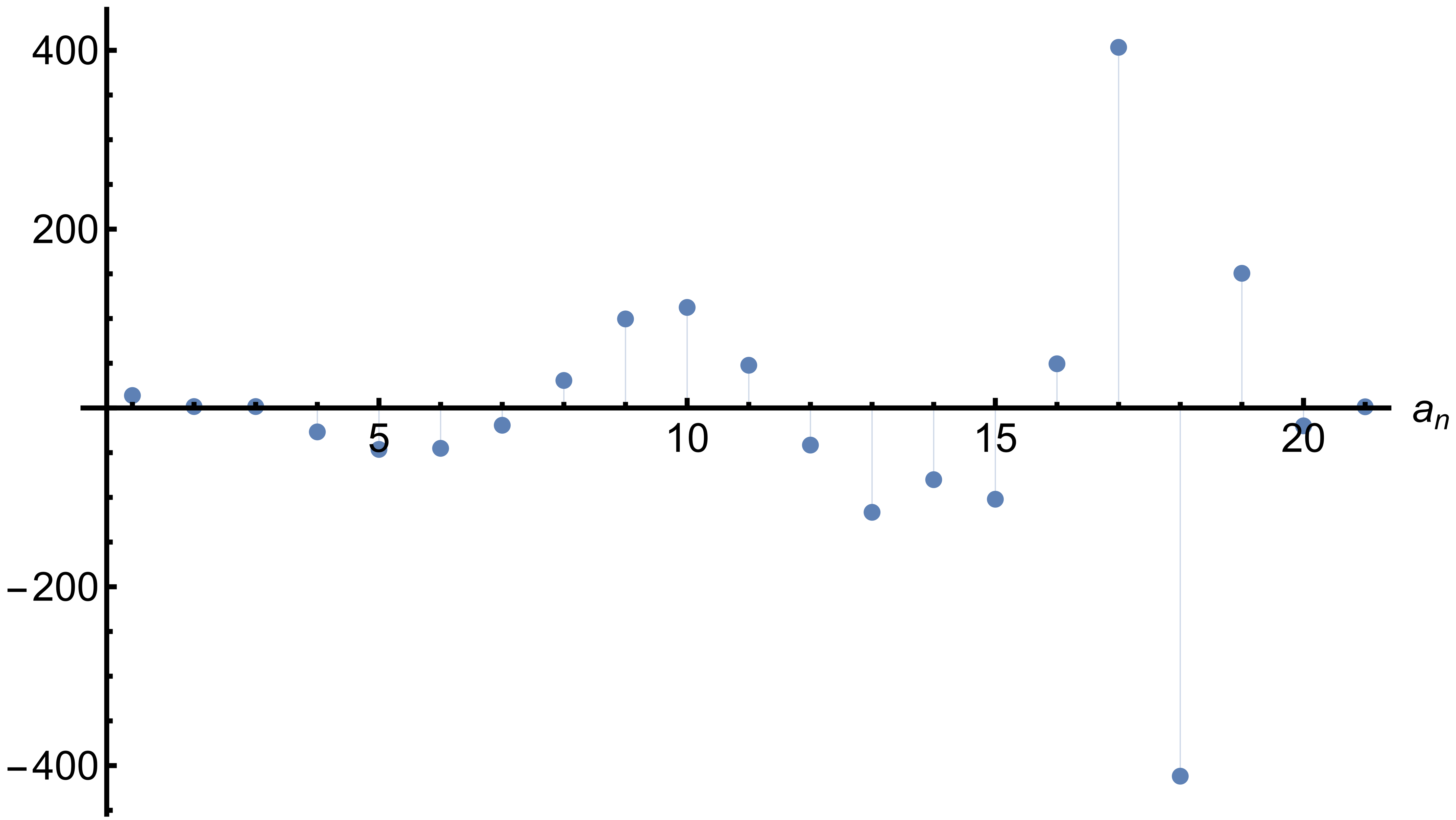
Benign overfitting

with optimization



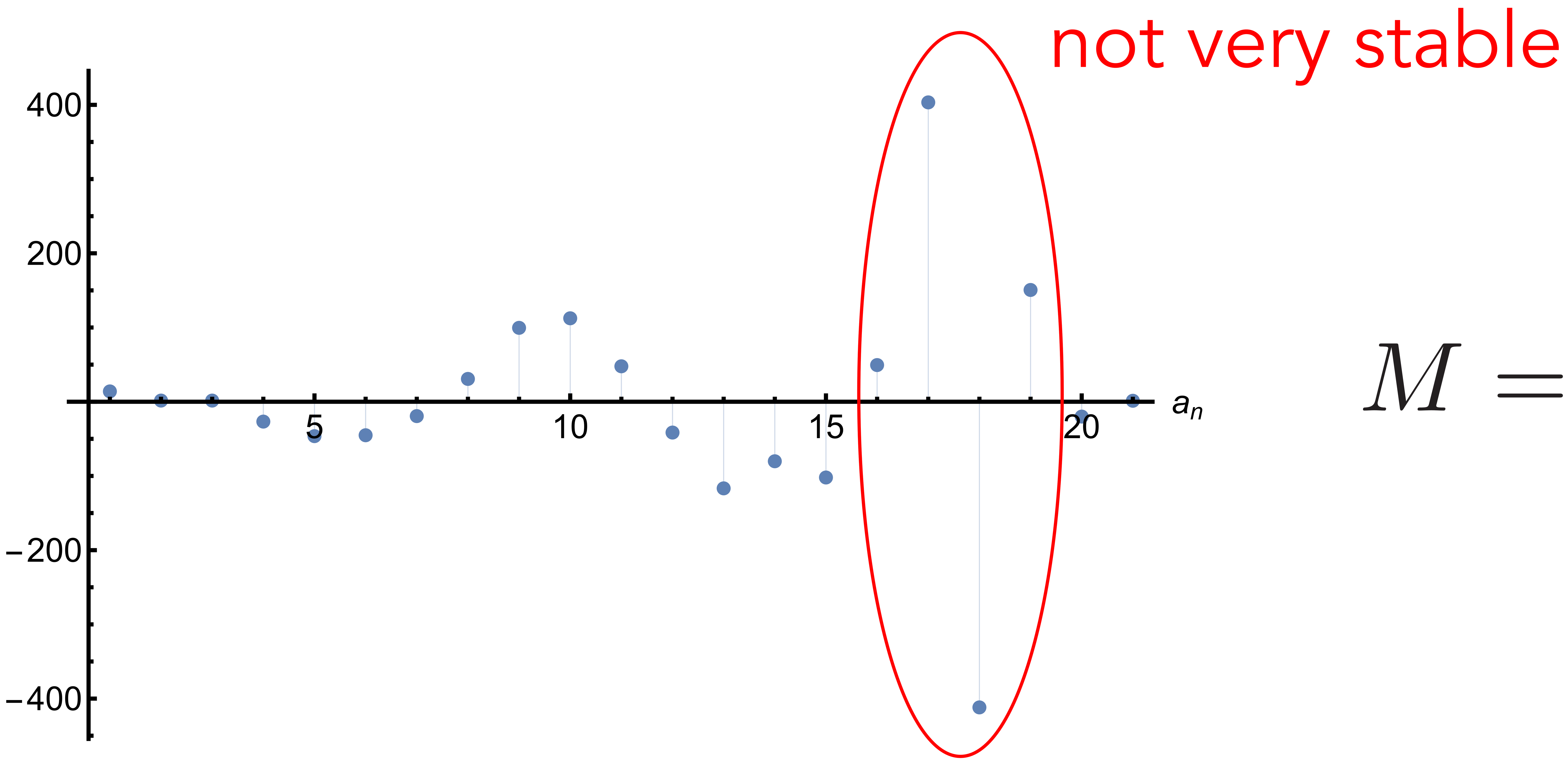
$$M = 20$$

Benign overfitting



$$M = 20$$

Benign overfitting



$$M = 20$$

Benign overfitting

- Model:

$$y = \sum_{n=0}^{M} a_n x^n \qquad\qquad \theta = \{a_n\}_{n=1}^{M}$$

- Loss function:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \left\| \tilde{y}_i - y(x_i) \right\|^2$$

Benign overfitting

- Model:

$$y = \sum_{n=0}^{M} a_n x^n \qquad \theta = \{a_n\}_{n=1}^{M}$$

- Loss function:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \left\| \tilde{y}_i - y(x_i) \right\|^2 + \sum_{n=0}^{M} n\, a_n^2$$

Benign overfitting

- Model:

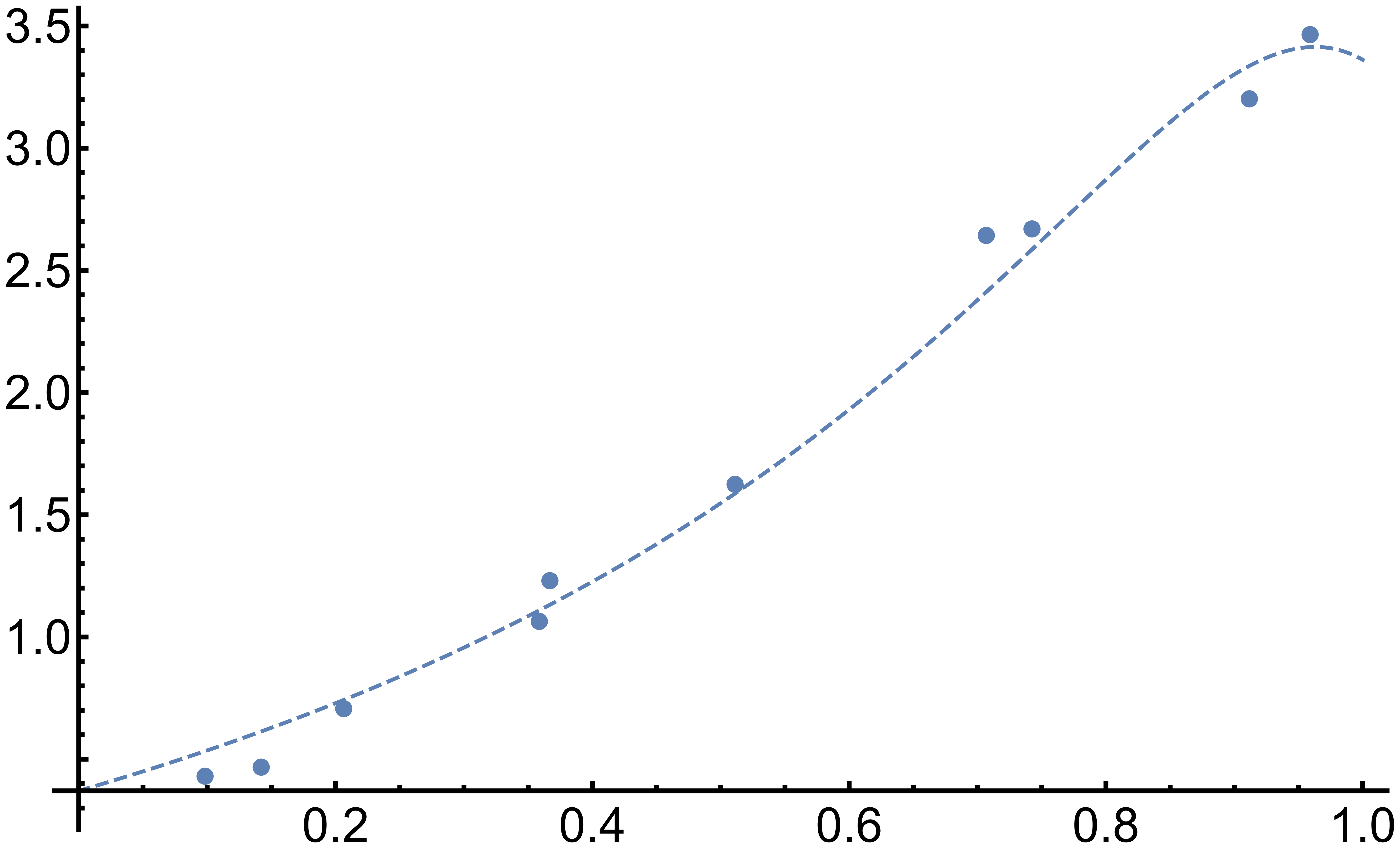
$$y = \sum_{n=0}^{M} a_n x^n \qquad \theta = \{a_n\}_{n=1}^{M}$$

- Loss function:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \left\| \tilde{y}_i - y(x_i) \right\|^2 + \sum_{n=0}^{M} n\, a_n^2$$

Benign overfitting

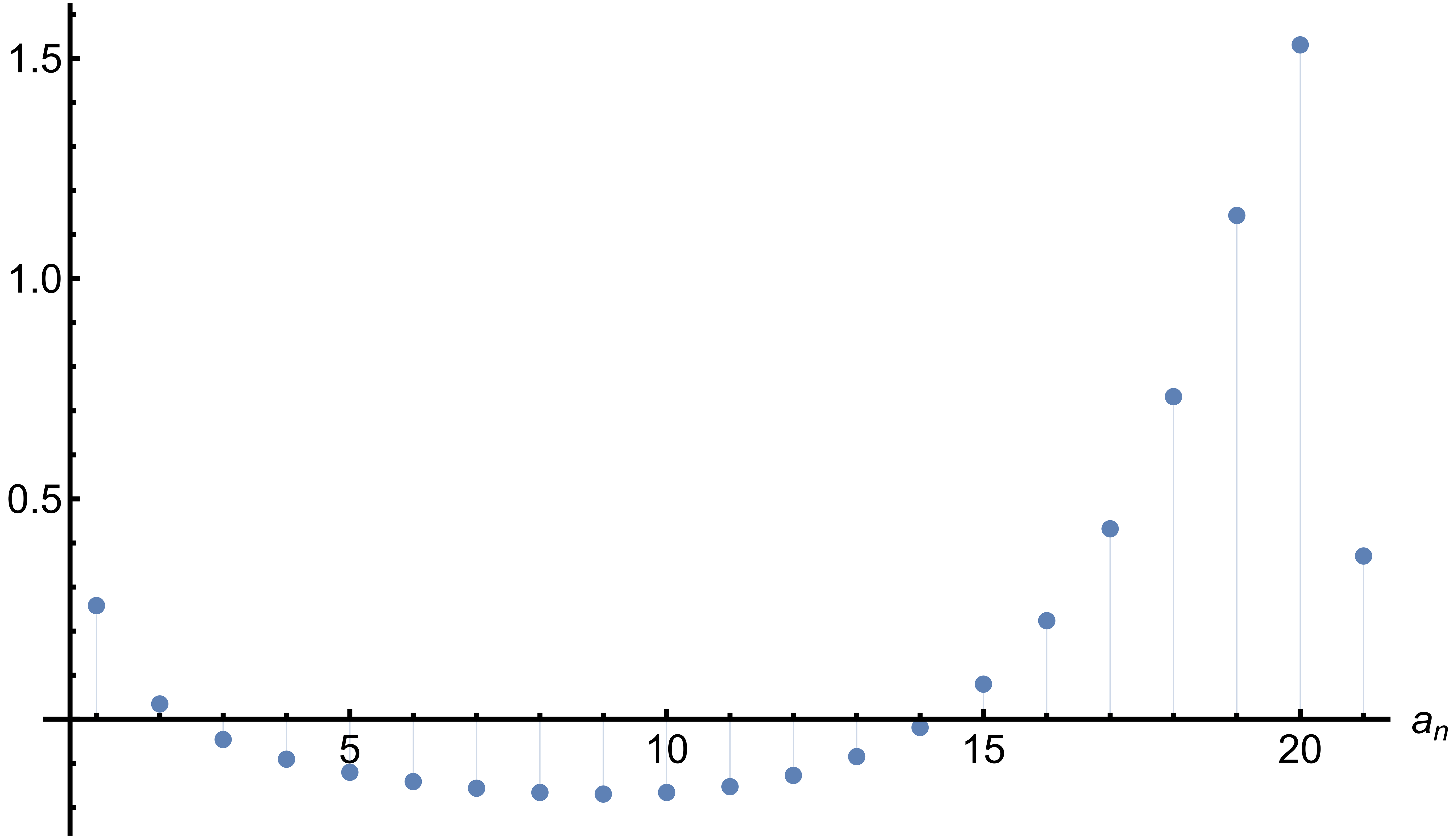
with optimization + regularization



$$M = 20$$

Benign overfitting

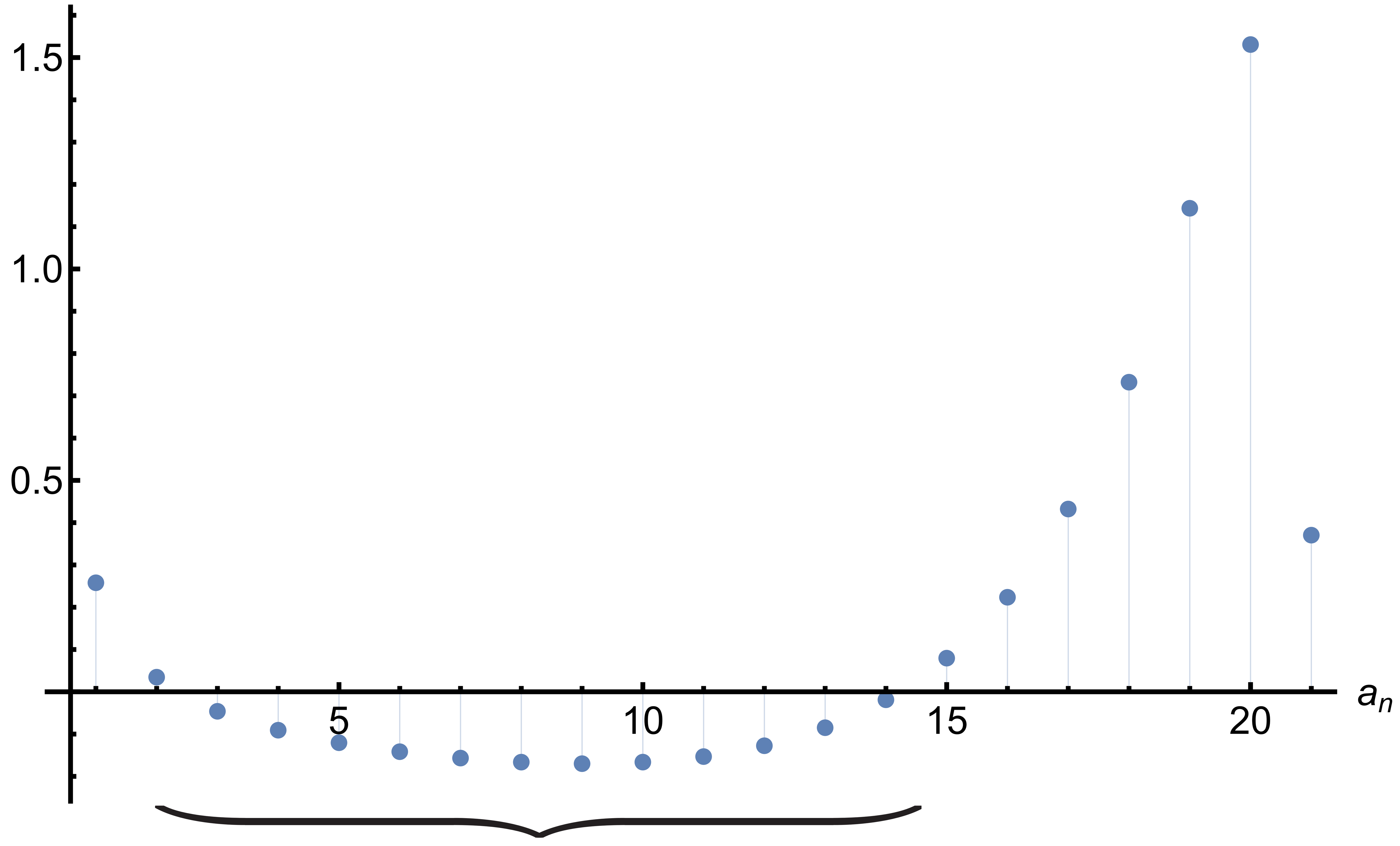
with optimization + regularization



$$M = 20$$

Benign overfitting

with optimization + regularization

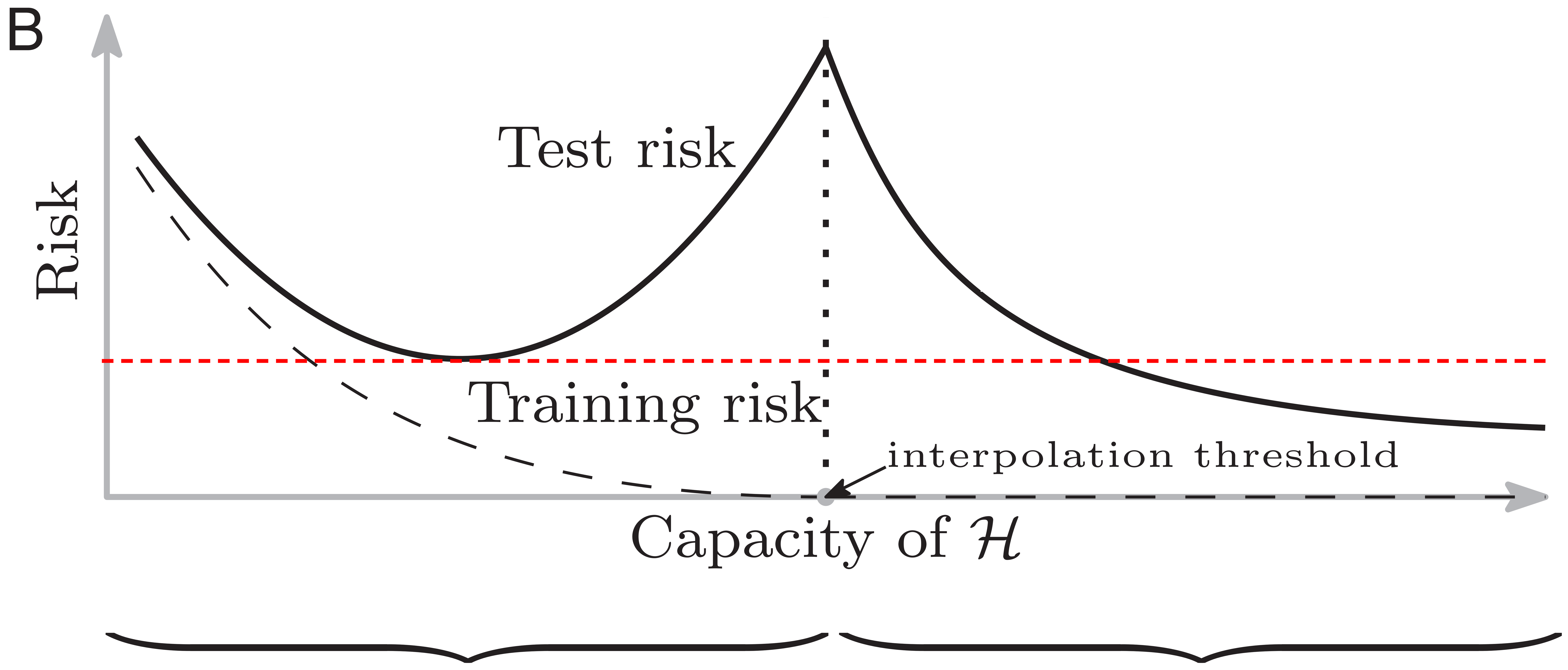


space is barely spanned

$$M = 20$$

Benign overfitting

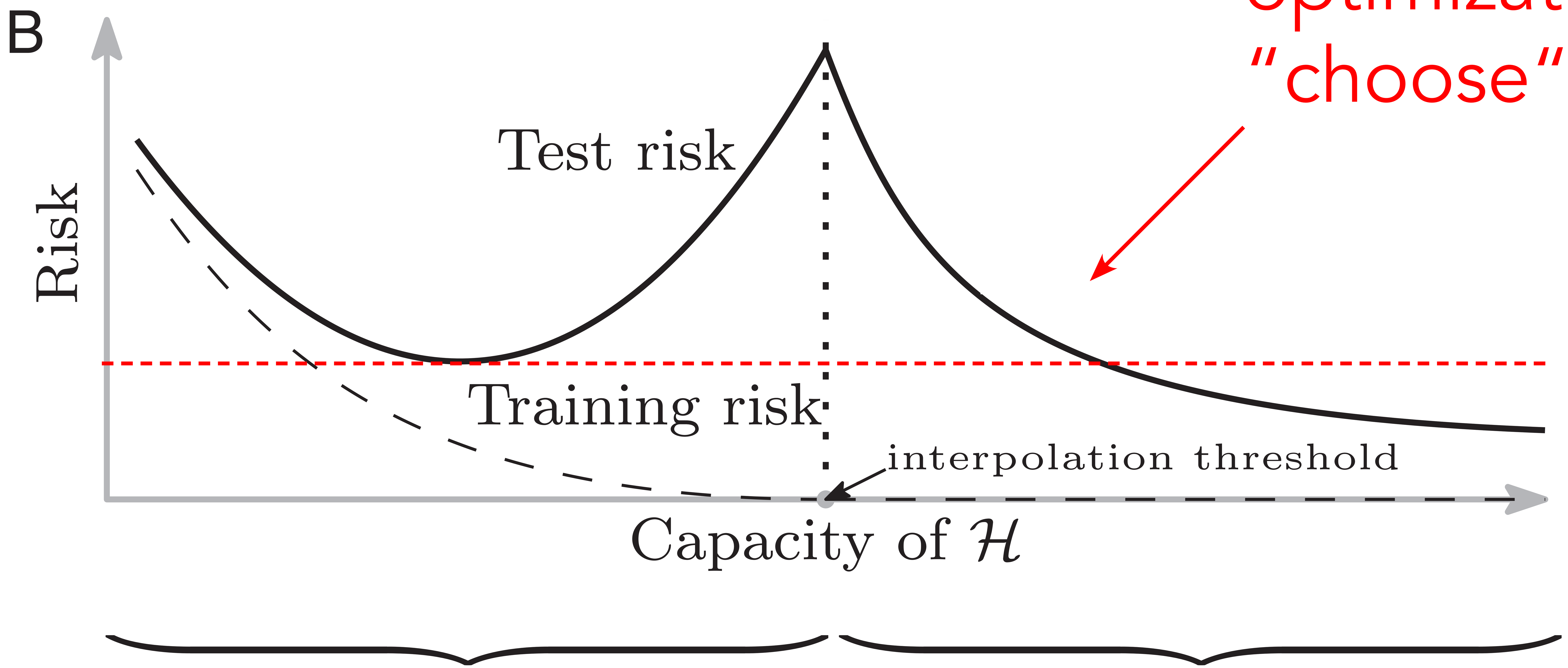
M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. PNAS, 116(32):15849–15854, 2019.



least squares under-determined

Benign overfitting

M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. PNAS, 116(32):15849–15854, 2019.



optimization can "choose" sub-space

least squares

under-determined

Conclusion and Outlook

- “Classical“ mathematatial tools can be useful to model and understand neural networks
  - Neural networks as ODEs and PDEs (stability, gradient computation, training, ...)
  - Benign overfitting

Conclusion and Outlook

- “Classical“ mathematial tools can be useful to model and understand neural networks
  - Neural networks as ODEs and PDEs (stability, gradient computation, training, ...)
  - Benign overfitting
- Still significant gap between theory and practice

Questions?

http://graphics.cs.uni-magdeburg.de/teaching/2021/imprs