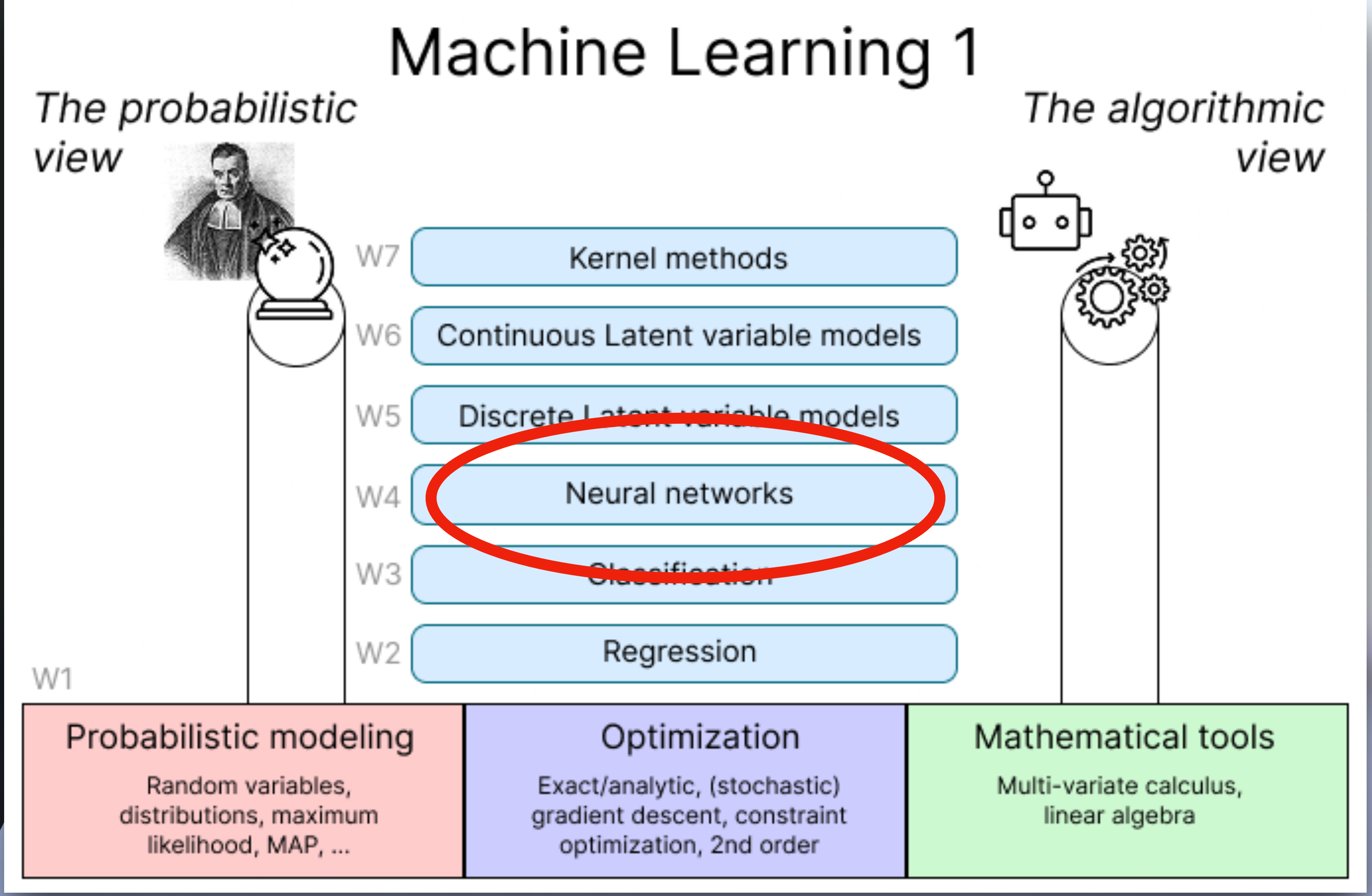
University of Amsterdam
Informatics Institute

AMLAB
Amsterdam
Machine Learning Lab

Machine Learning 1

Lecture 8 - Neural Networks

Erik Bekkers



Machine Learning 1

The probabilistic view

The algorithmic view

W7 Kernel methods
W6 Continuous Latent variable models
W5 Discrete Latent variable models
W4 Neural networks
W3 Classification
W2 Regression
W1

Probabilistic modeling
Random variables, distributions, maximum likelihood, MAP, ...

Optimization
Exact/analytic, (stochastic) gradient descent, constraint optimization, 2nd order

Mathematical tools
Multi-variate calculus, linear algebra

Image credit: Kirillm | Getty Images

UNIVERSITY OF AMSTERDAM
Informatics Institute

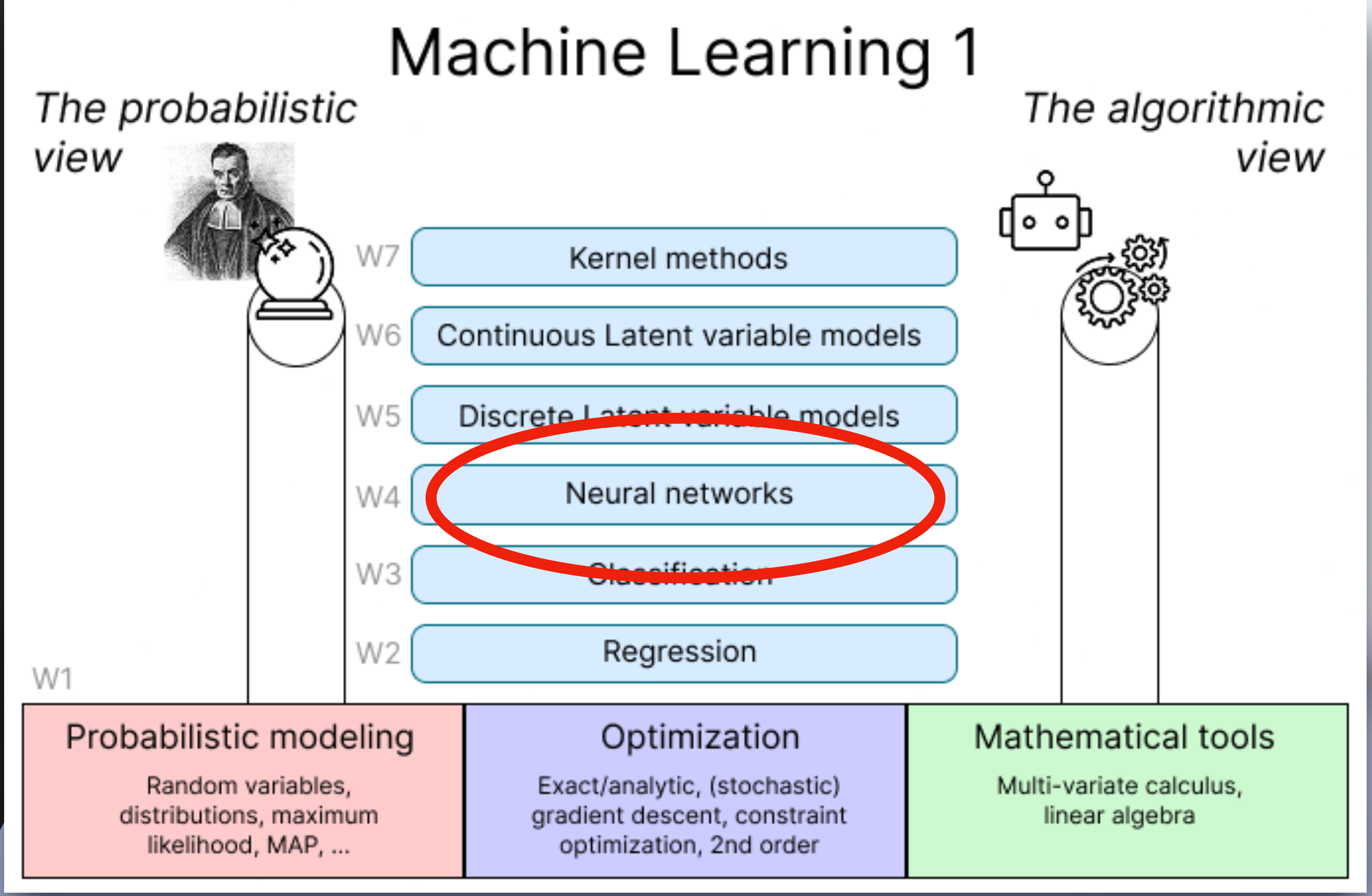
AMLAB Amsterdam Machine Learning Lab

Machine Learning 1

Lecture 8.1 - Supervised Learning
Neural Networks

Erik Bekkers

(Bishop 5.1)



Machine Learning 1

The probabilistic view

The algorithmic view

- W7 Kernel methods
- W6 Continuous Latent variable models
- W5 Discrete Latent variable models
- W4 Neural networks
- W3 Classification
- W2 Regression
- W1

Probabilistic modeling	Optimization	Mathematical tools
Random variables, distributions, maximum likelihood, MAP, ...	Exact/analytic, (stochastic) gradient descent, constraint optimization, 2nd order	Multi-variate calculus, linear algebra

Slide credits: Patrick Forré and Rianne van den Berg

Image credit: Kirillm | Getty Images

Why activation funct. are needed?

$$\underline{h\left(W^{(3)^T} h\left(W^{(2)^T} h\left(\overset{M \times D}{W^{(1)^T}} \underline{x}\right.\right.\right.}\left.\left.\left.\right)\right)\right)$$

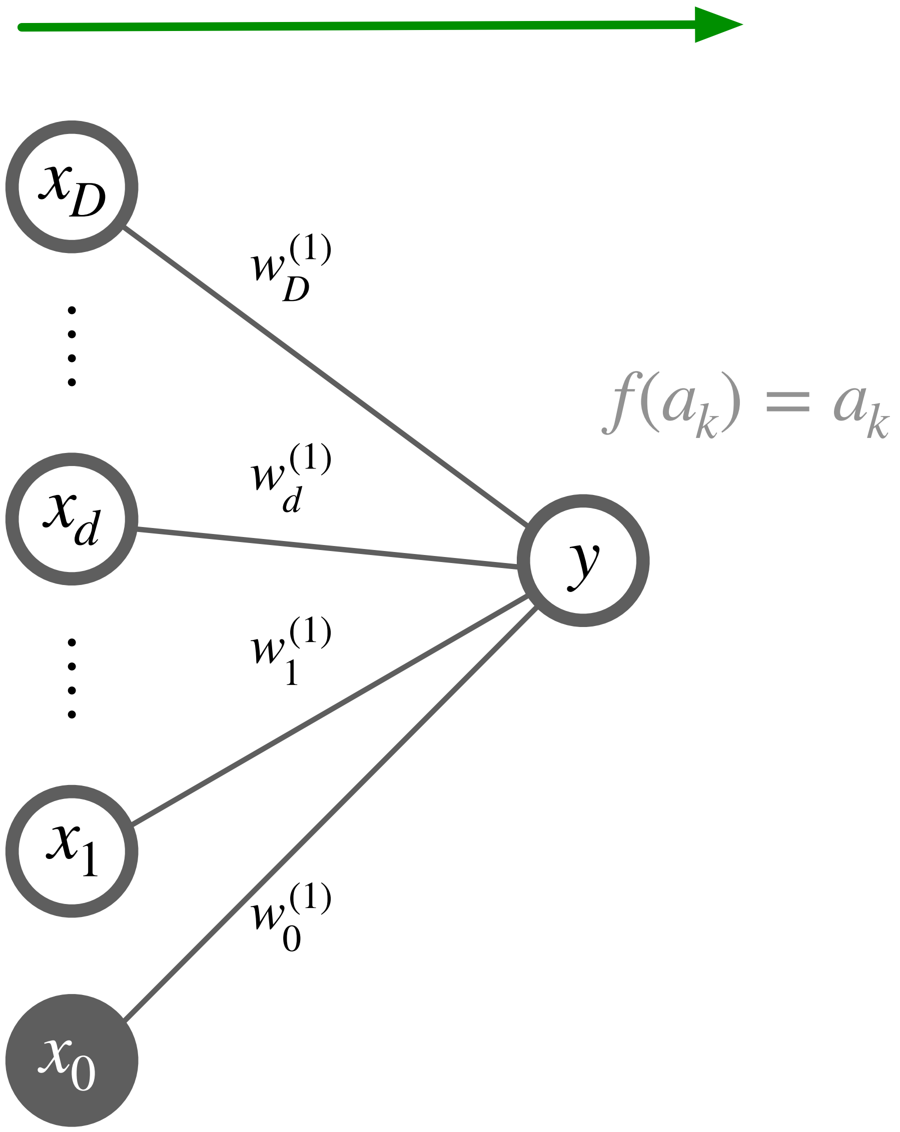
$$M'' \leftarrow M' \leftarrow M \leftarrow D$$

still linear

$$\underline{\overset{M'' \times D}{W^{(3,2,1)^T}} \underline{x}}$$

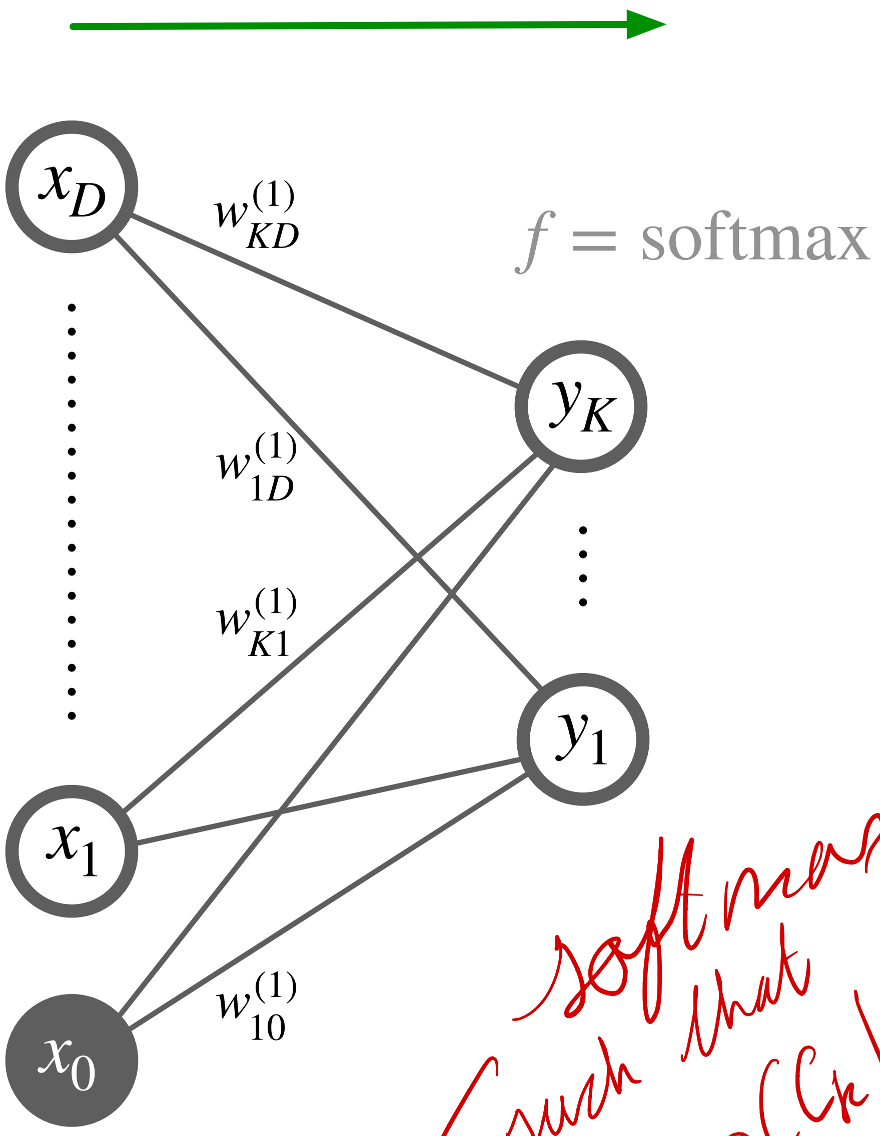
without act-fn it is still effectively one matrix vector mult!

Linear regression & classification as NN



$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \mathbf{x}$$

Figure: Linear regression as 1-layer NN



softmax such that $= p(C_k | x)$

$$y_k(\mathbf{x}, \mathbf{w}) = f(\mathbf{w}_k^T \mathbf{x})$$

Figure: Linear Classification with K classes as 1-layer NN

Feed-Forward Networks: Skip Connections

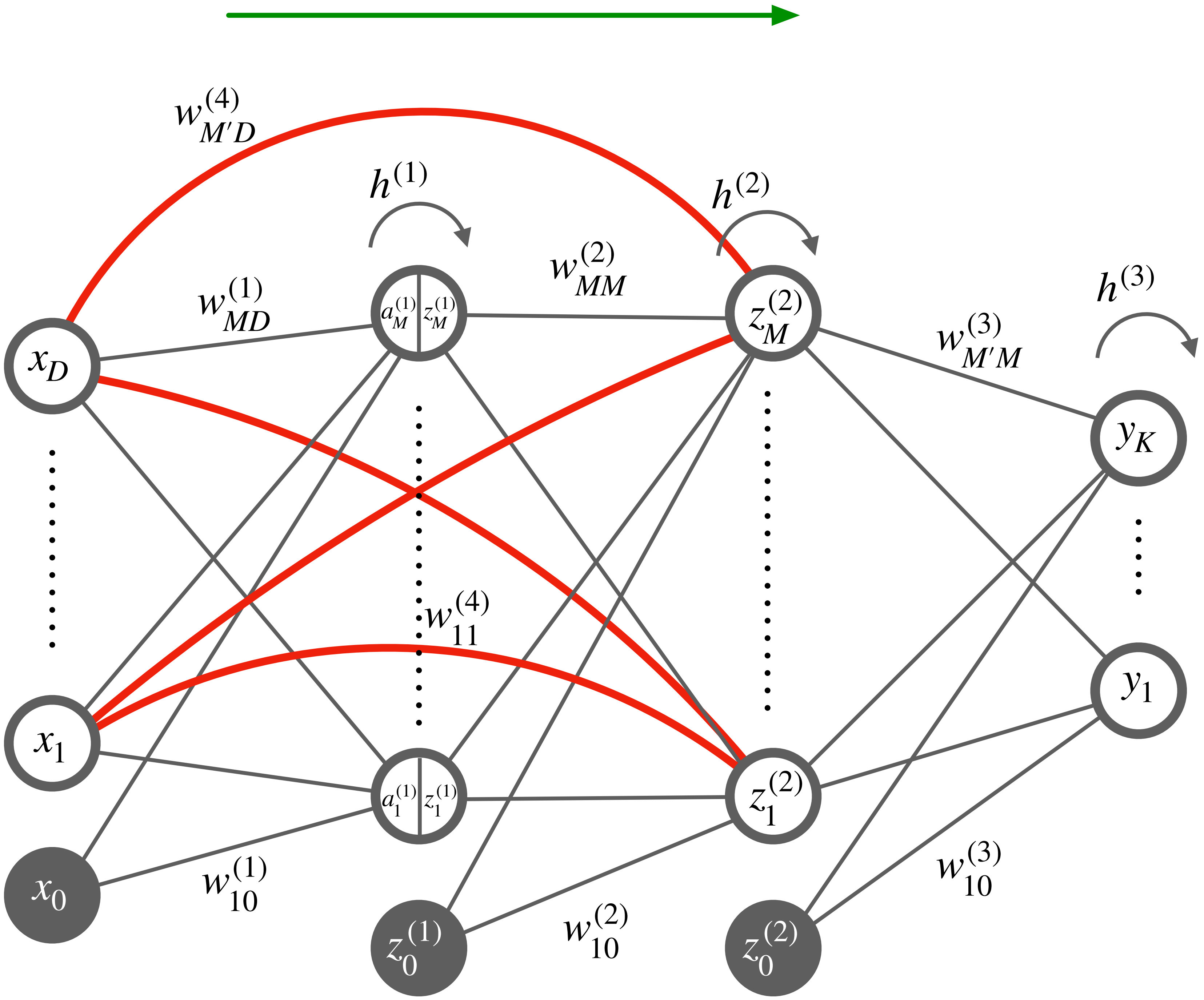
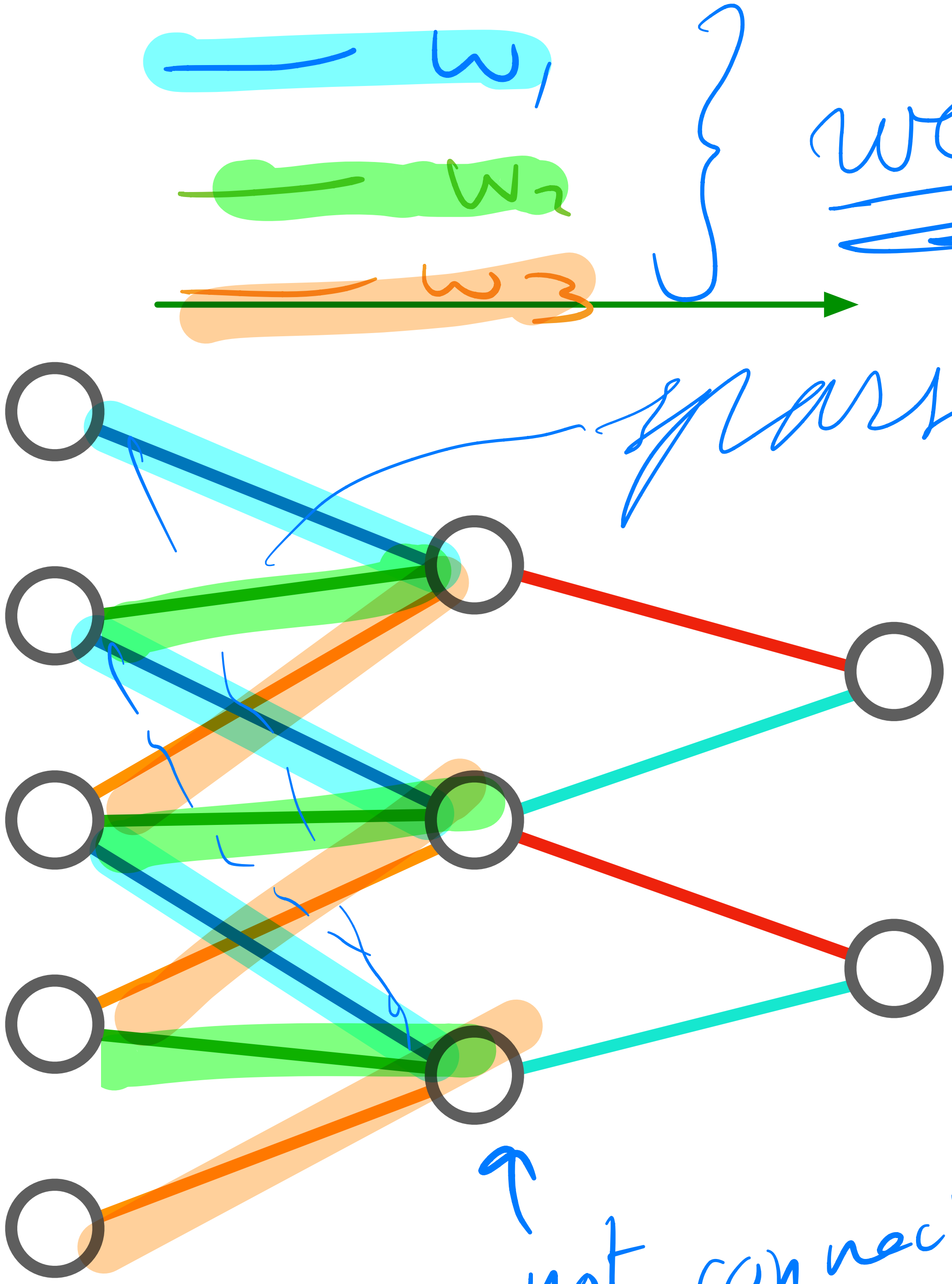


Figure: 3 layer feed-forward net with skip connections

Feed-Forward Networks: Sparse Connections



weight sharing

sparse: not all connection exist

not connected to all previous nodes: sparsely connected

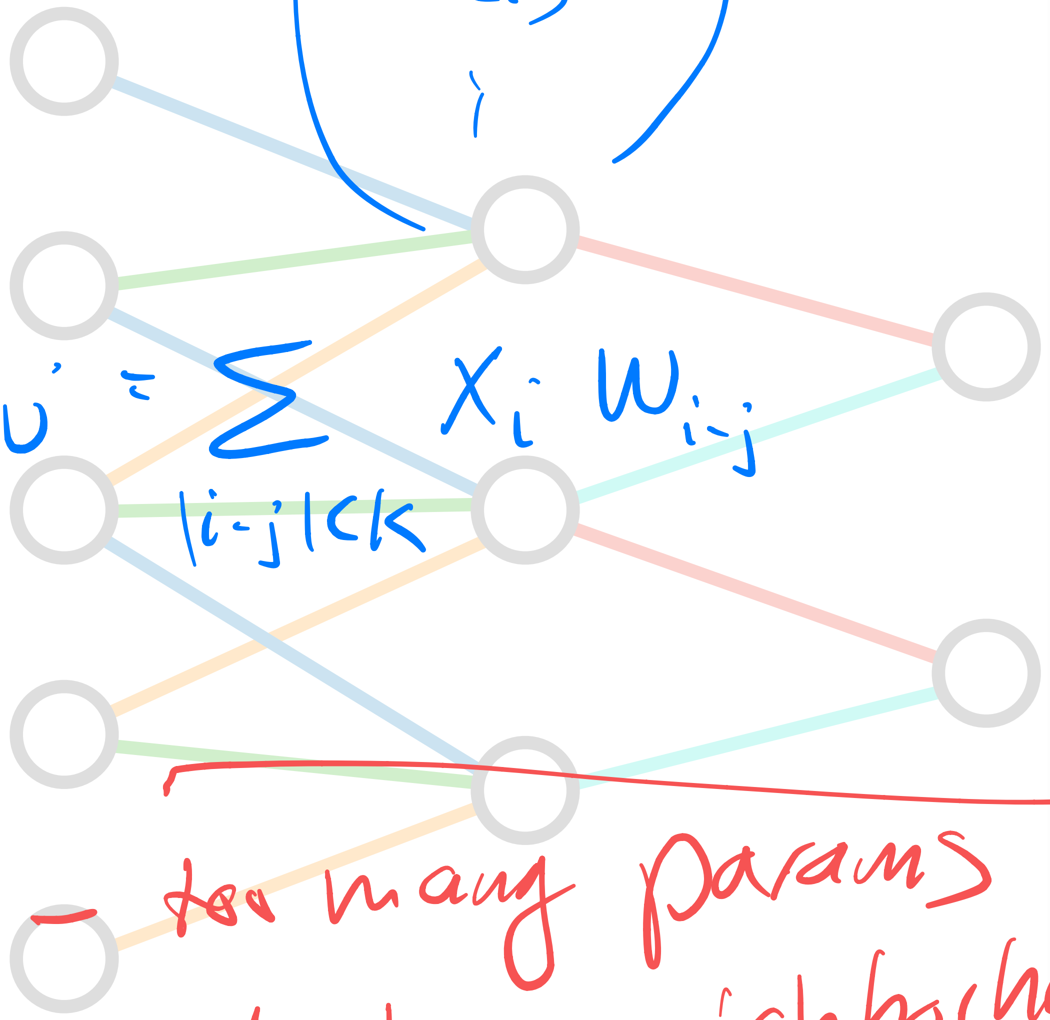
Figure: Feed-forward architecture with sparse connections. With "special" weight sharing --> Convolutional Neural Nets (Le Cun et al 1989)

layer is equivariant

Feed-Forward Networks: Sparse Connections

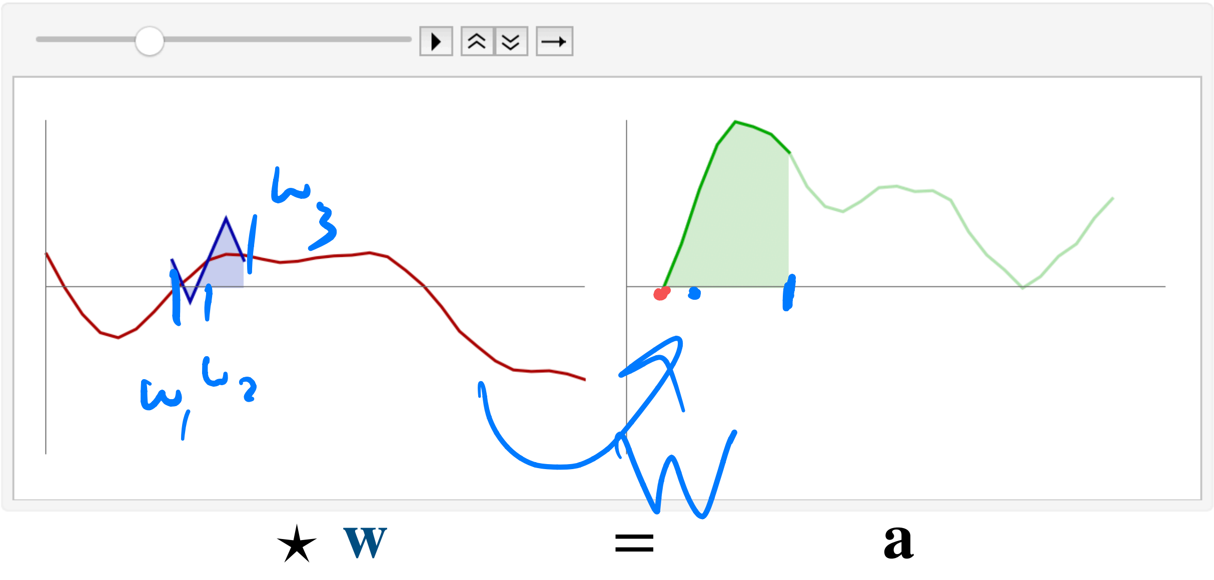
Here are $\underline{a}, \underline{x}$ treated as signals

$$\underline{a} = \begin{pmatrix} a(t_1) \\ a(t_2) \\ \vdots \end{pmatrix}$$



$$a_j = \sum_{|i-j|<k} x_i w_{ij}$$

- too many params
- destroy neighborhood info



Example: 1D convolution = sparse + weightsharing:

$$\begin{pmatrix} a_1 \\ a_2 \\ a_2 \\ \vdots \\ a_M \end{pmatrix} = \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} & \cdots \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} & \cdots \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} & \cdots \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} & \cdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_D \end{pmatrix}$$

$$= \begin{pmatrix} w_1 & w_2 & w_3 & 0 & 0 & \cdots \\ 0 & w_1 & w_2 & w_3 & 0 & \cdots \\ 0 & 0 & w_1 & w_2 & w_3 & \cdots \\ 0 & 0 & 0 & w_1 & w_2 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_D \end{pmatrix}$$

Figure: Feed-forward architecture with sparse connections. With special weight sharing --> Convolutional Neural Nets (Le Cun et al 1989)

University of Amsterdam
Informatics Institute

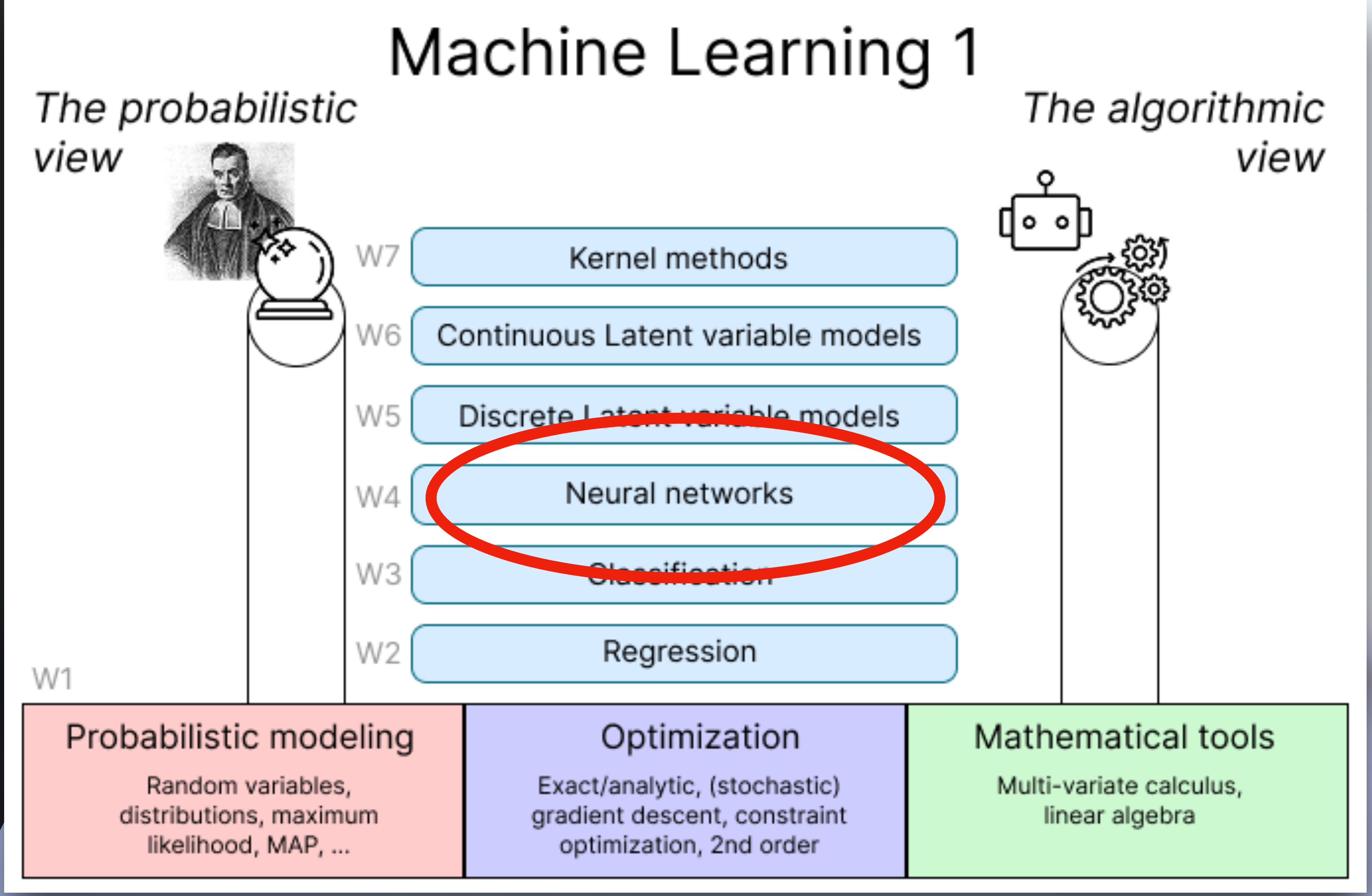
AMLAB
Amsterdam
Machine Learning Lab

Machine Learning 1

Lecture 8.2 - Supervised Learning
Neural Networks - Universal Approximators

Erik Bekkers

(Bishop 5.1)



Machine Learning 1

The probabilistic view

The algorithmic view

W7 Kernel methods
W6 Continuous Latent variable models
W5 Discrete Latent variable models
W4 Neural networks
W3 Classification
W2 Regression
W1

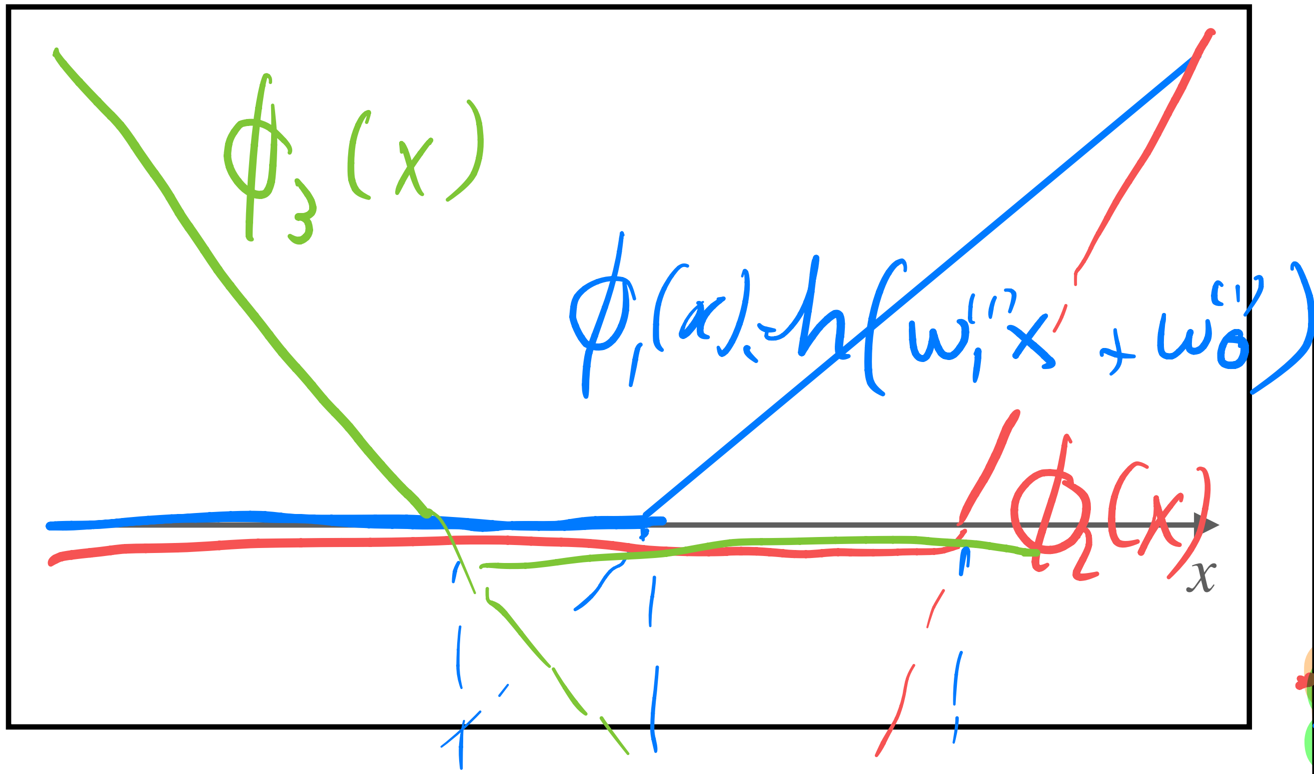
Probabilistic modeling	Optimization	Mathematical tools
Random variables, distributions, maximum likelihood, MAP, ...	Exact/analytic, (stochastic) gradient descent, constraint optimization, 2nd order	Multi-variate calculus, linear algebra

Slide credits: Patrick Forré and Rianne van den Berg

Image credit: Kirillm | Getty Images

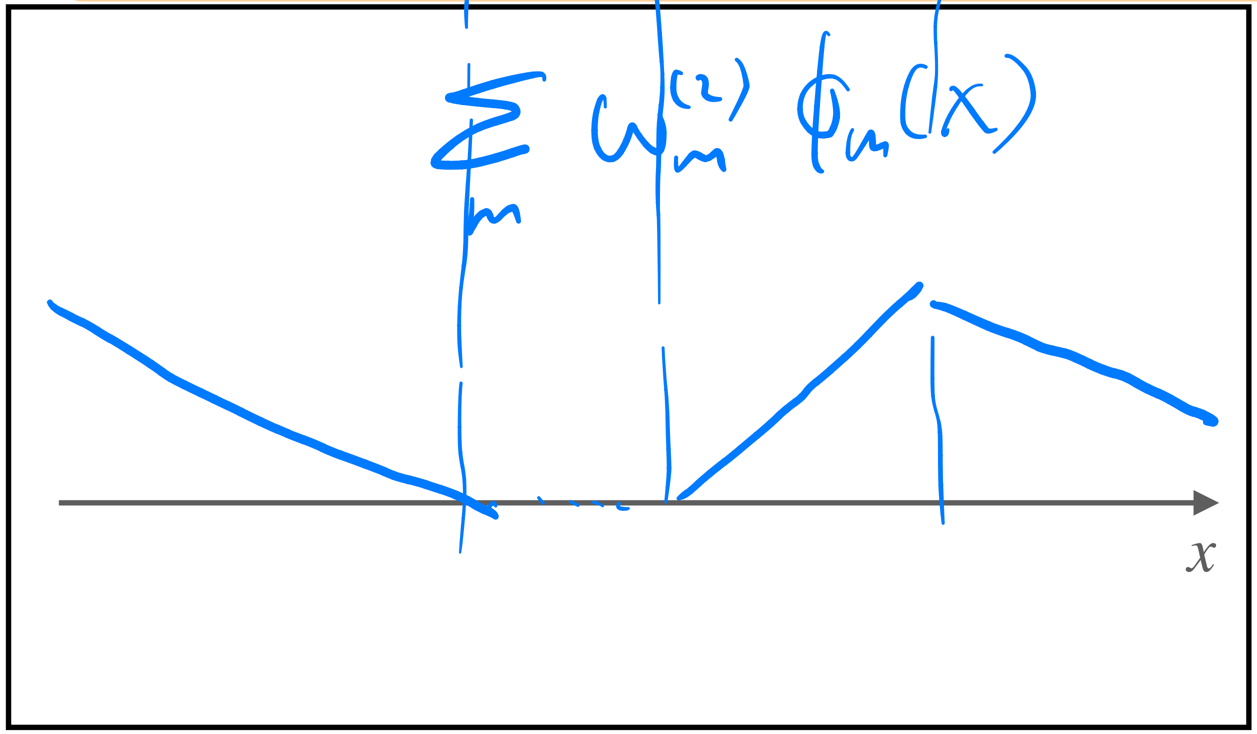
Neural Networks with ReLU

Learned basis functions with ReLU



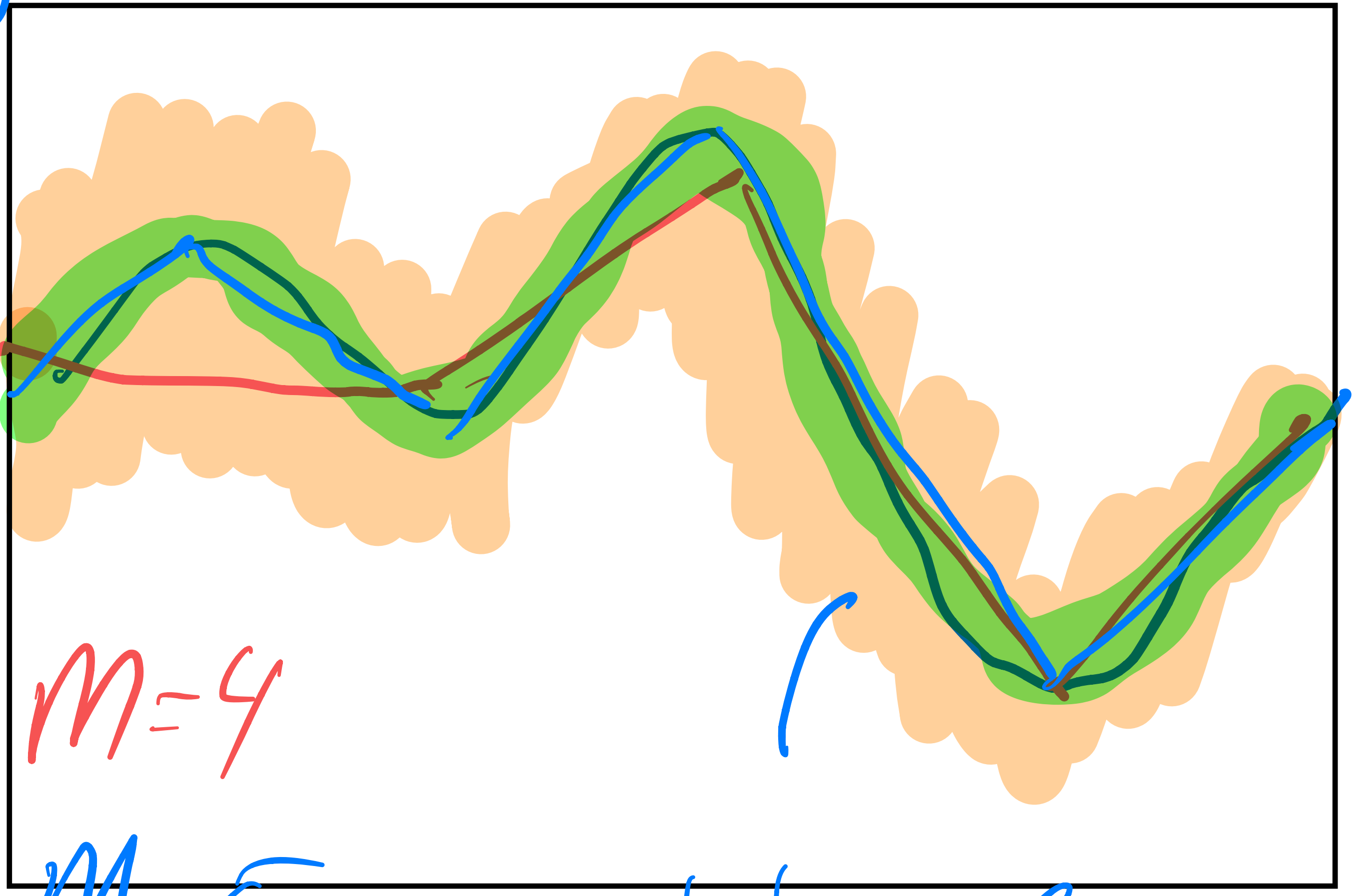
$\phi_1(x) = h(w_1^{(1)} x + w_0^{(1)})$

Leads to piece-wise linear (PWL) functions



$\sum_m w_m^{(2)} \phi_m(x)$

Approximation with ReLU NNs/PWL functions



$M=4$

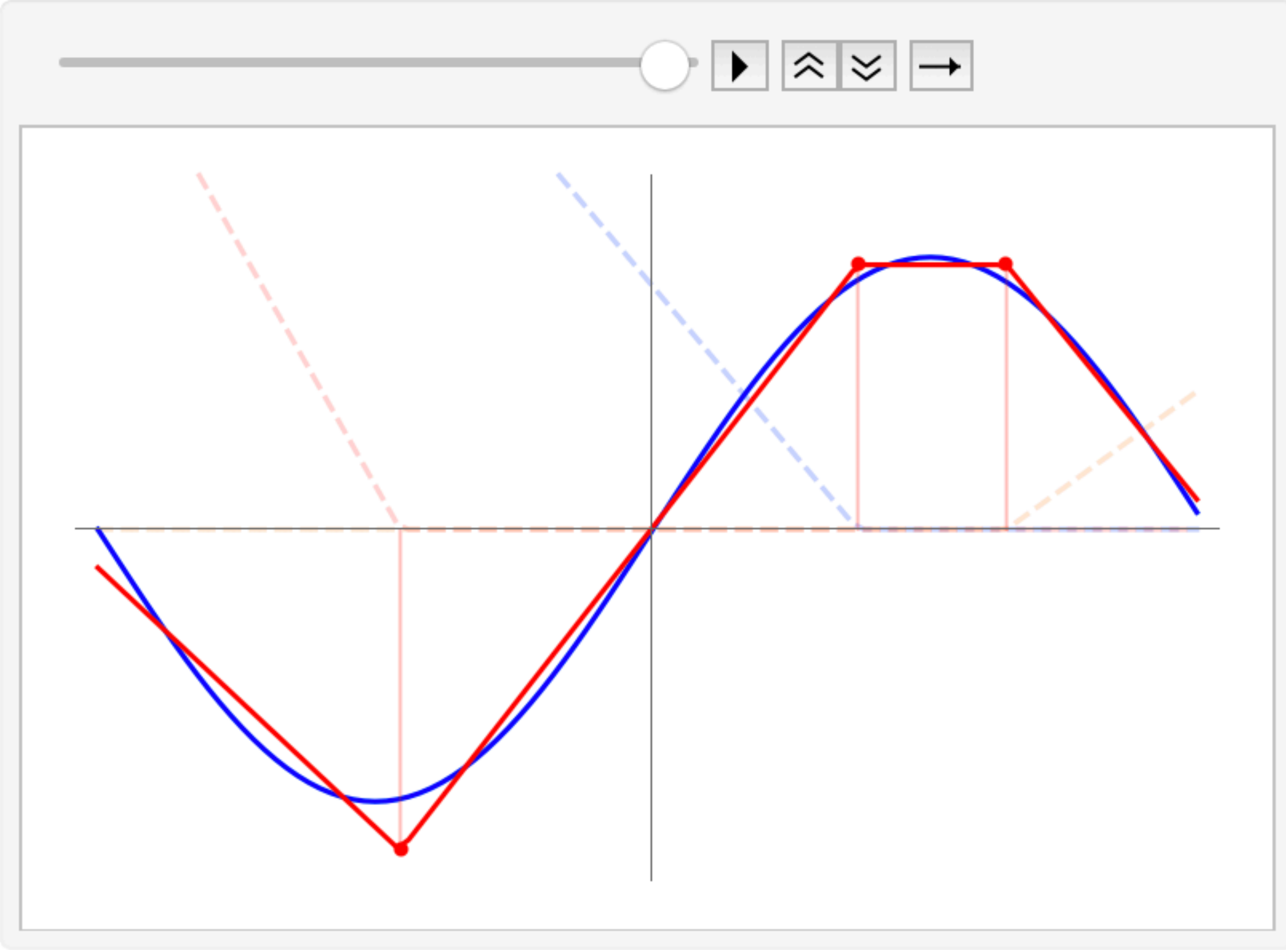
$M=5$

tolerance

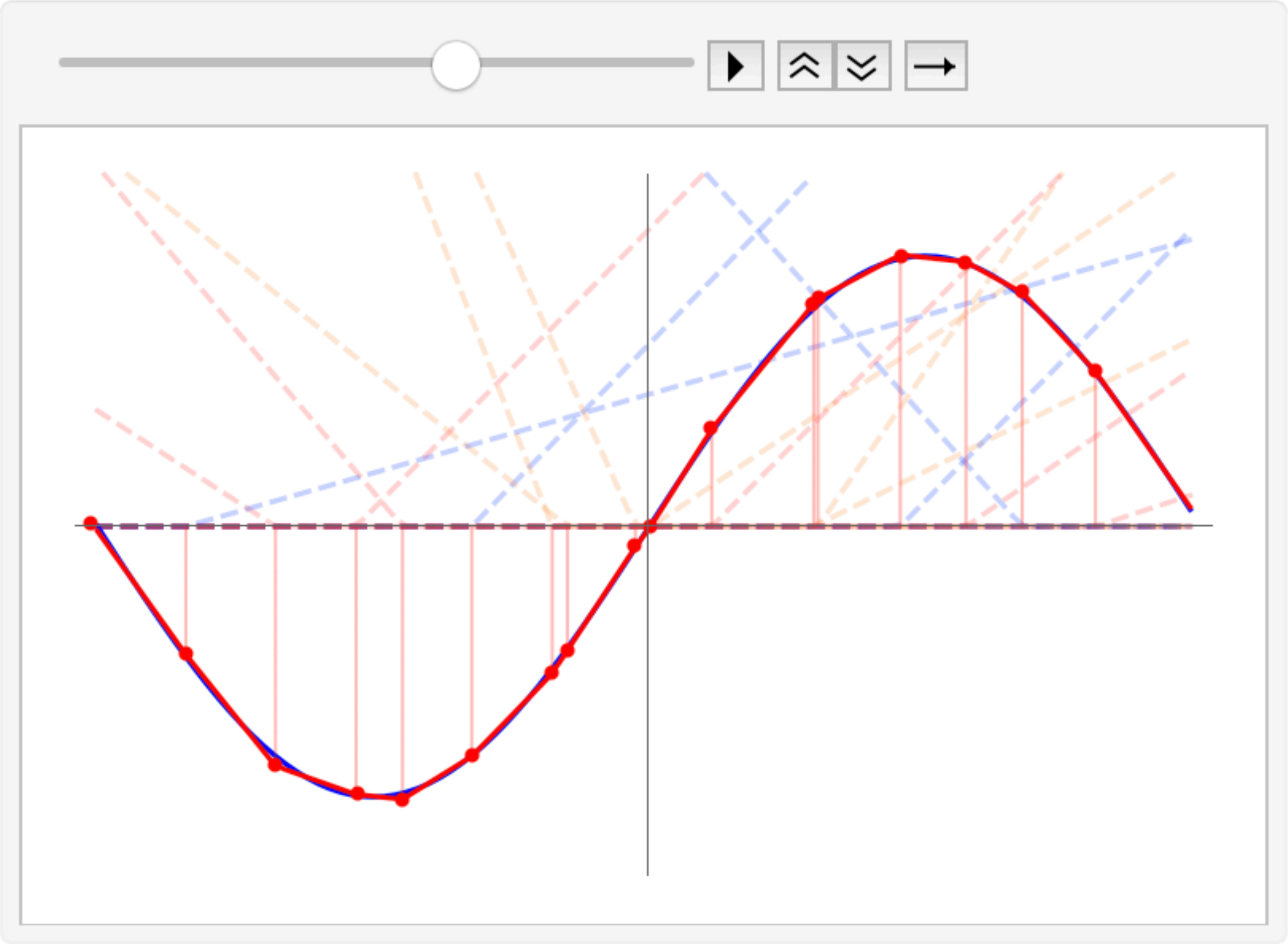
ε

Example: Function Approximations

Piece-wise linear approximation with 2-layer NN



M=3 hidden units



M=20 hidden units

Example: Classification with Neural Nets

MLP:

- 2 layers
- # of inputs: 2
- 2 hidden units with tanh activation function
- # of outputs: 1 or 2
- Output activation function: σ softmax
- Red line: MLP decision boundary
- Green line: optimal decision boundary from synthetic data distribution

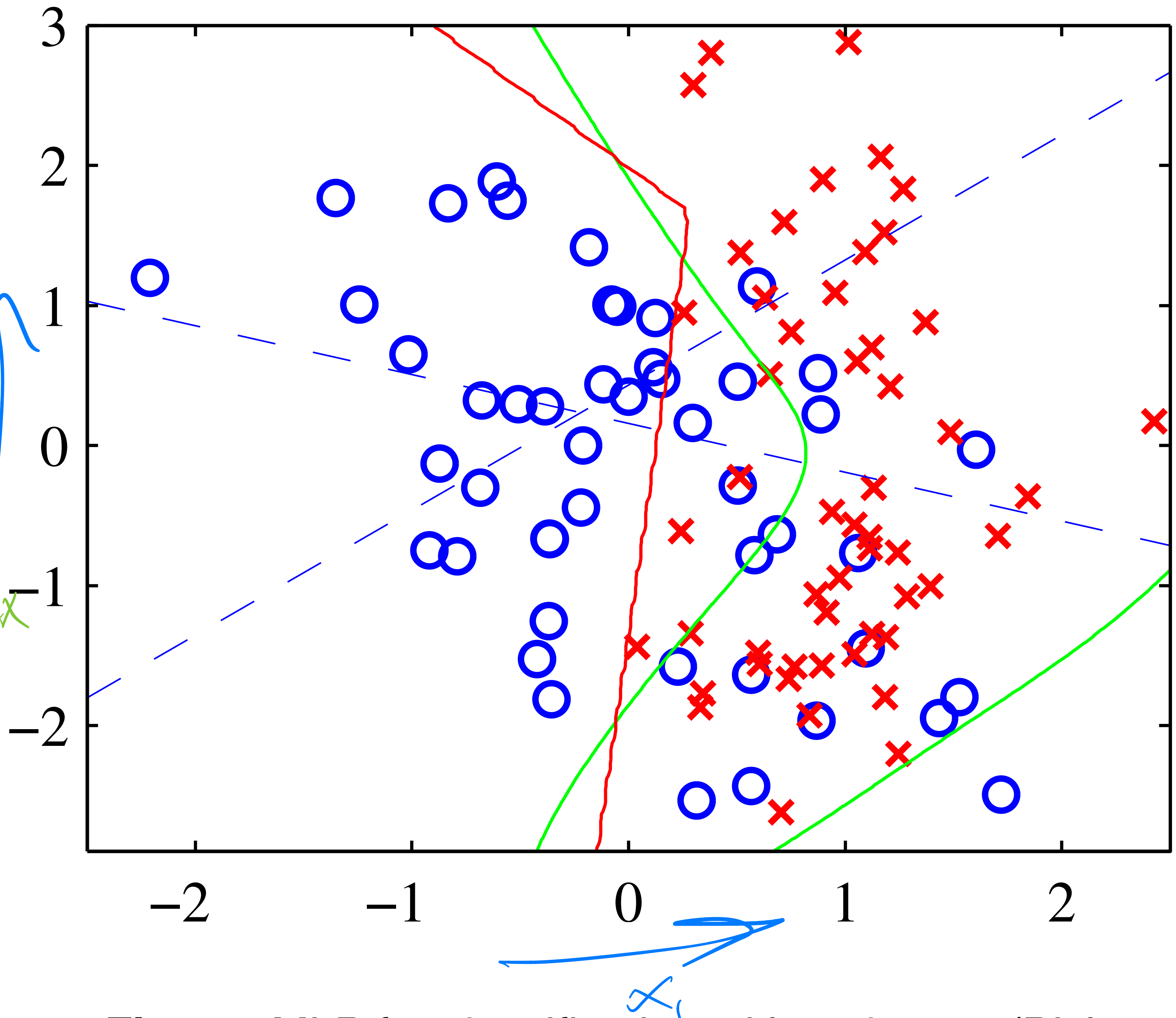


Figure: MLP for classification with 2 classes (Bishop5.4)

University of Amsterdam
Informatics Institute

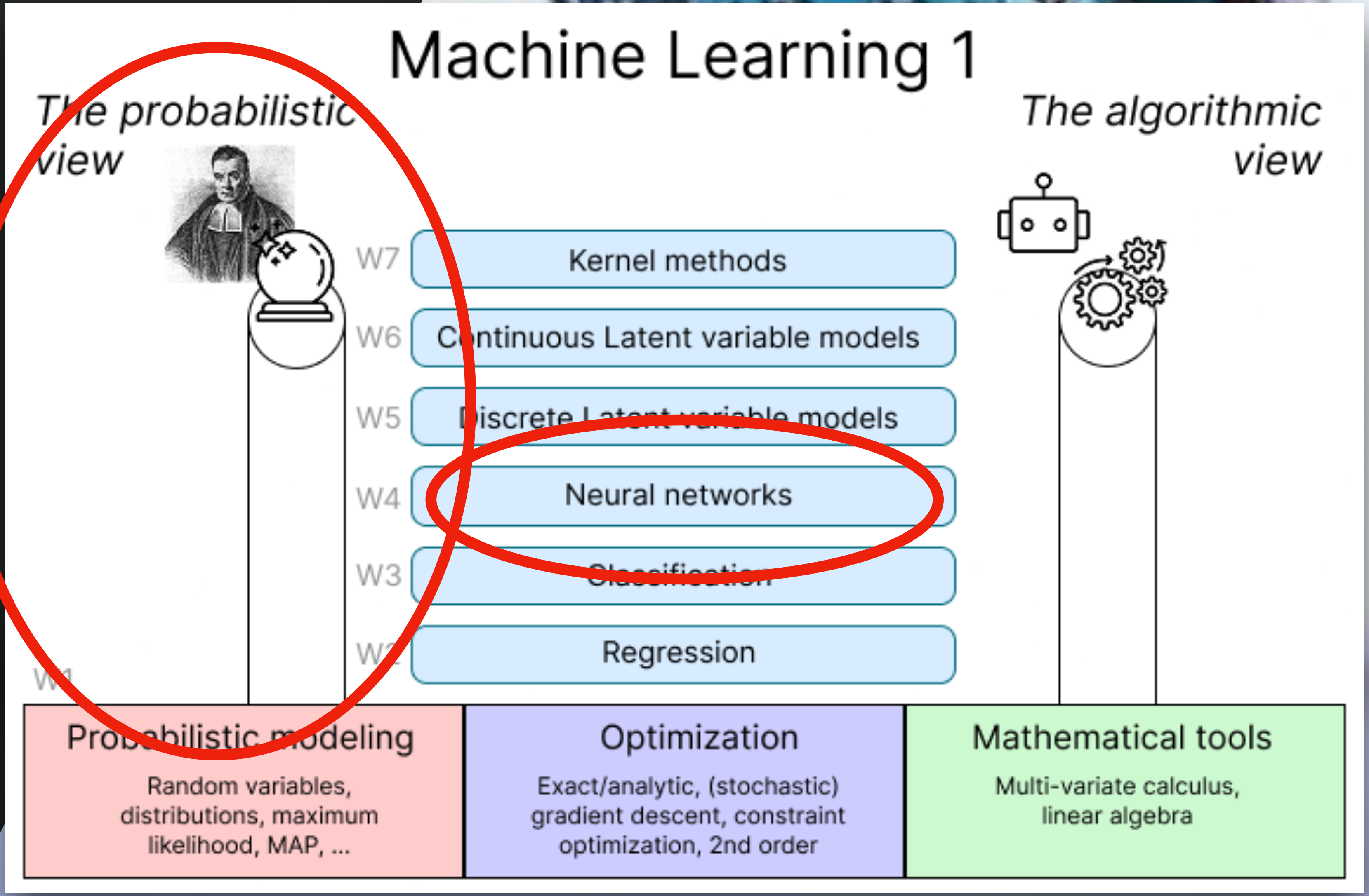
AMLAB Amsterdam Machine Learning Lab

Machine Learning 1

Lecture 8.3 - Supervised Learning
Neural Networks - Losses

Erik Bekkers

(Bishop 5.2.0)



Machine Learning 1

The probabilistic view

The algorithmic view

- W7 Kernel methods
- W6 Continuous Latent variable models
- W5 Discrete Latent variable models
- W4 Neural networks
- W3 Classification
- W2 Regression
- W1

Probabilistic modeling
Random variables, distributions, maximum likelihood, MAP, ...

Optimization
Exact/analytic, (stochastic) gradient descent, constraint optimization, 2nd order

Mathematical tools
Multi-variate calculus, linear algebra

Slide credits: Patrick Forré and Rianne van den Berg

Image credit: KirilIm | Getty Images

UNIVERSITY OF AMSTERDAM
Informatics Institute

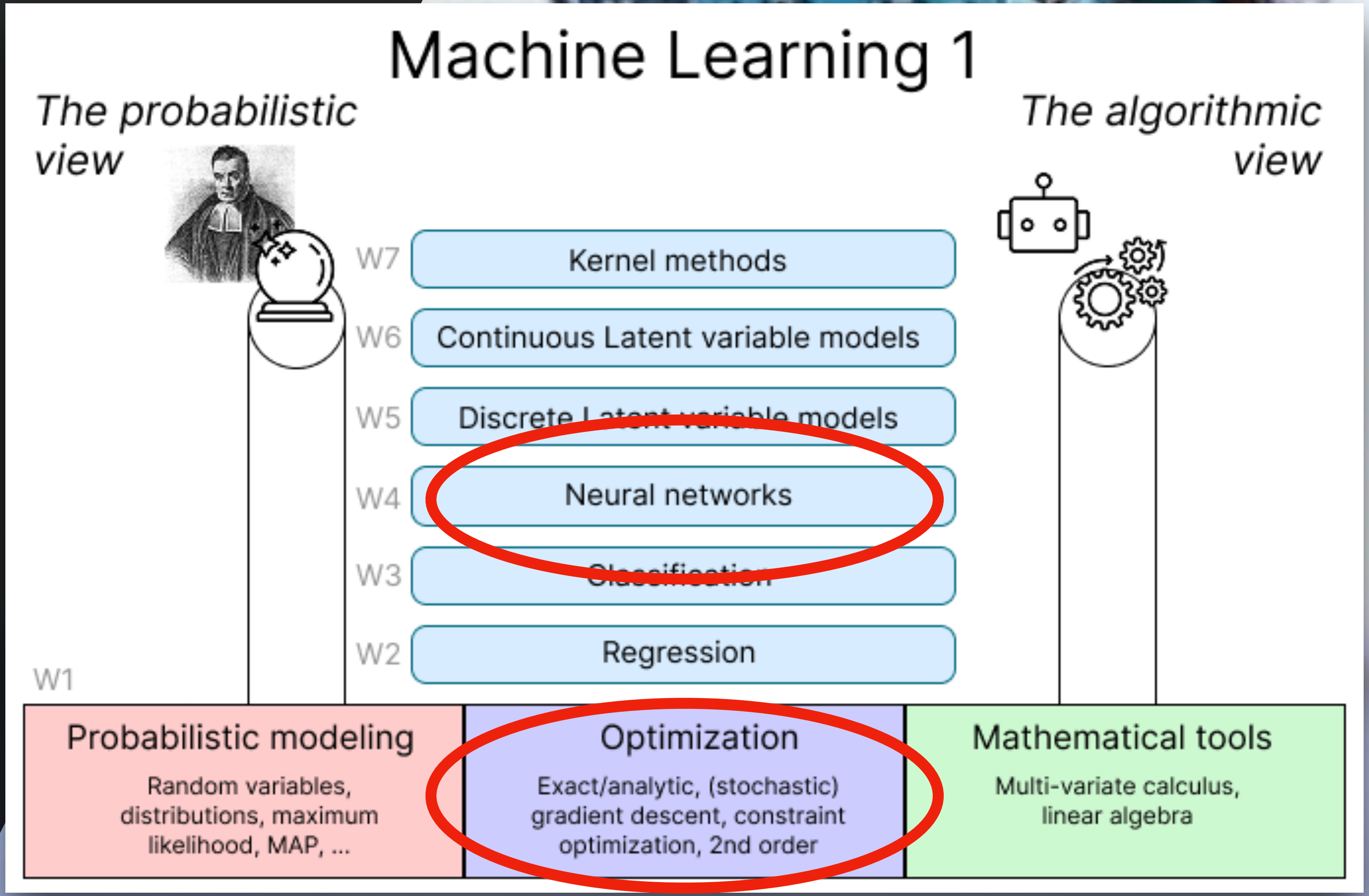
AMLAB
Amsterdam
Machine Learning Lab

Machine Learning 1

Lecture 8.4 - Supervised Learning
Neural Networks - Training

Erik Bekkers

(Bishop 5.2)



Machine Learning 1

The probabilistic view

The algorithmic view

W7 Kernel methods

W6 Continuous Latent variable models

W5 Discrete Latent variable models

W4 Neural networks

W3 Classification

W2 Regression

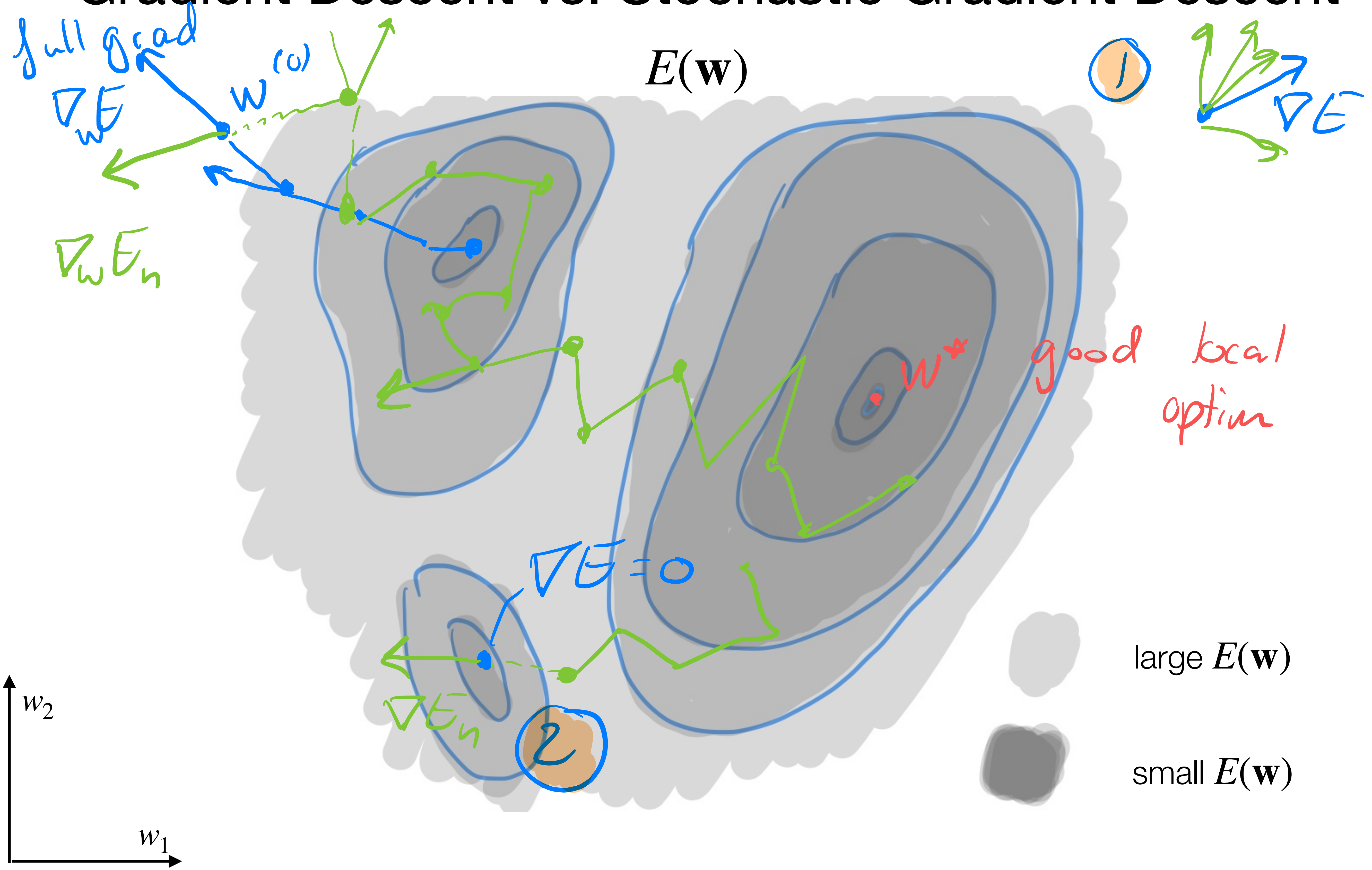
W1

Probabilistic modeling	Optimization	Mathematical tools
Random variables, distributions, maximum likelihood, MAP, ...	Exact/analytic, (stochastic) gradient descent, constraint optimization, 2nd order	Multi-variate calculus, linear algebra

Slide credits: Patrick Forré and Rianne van den Berg

Image credit: Kirillm | Getty Images

Gradient Descent vs. Stochastic Gradient Descent



Gradient Descent vs. Stochastic Gradient Descent

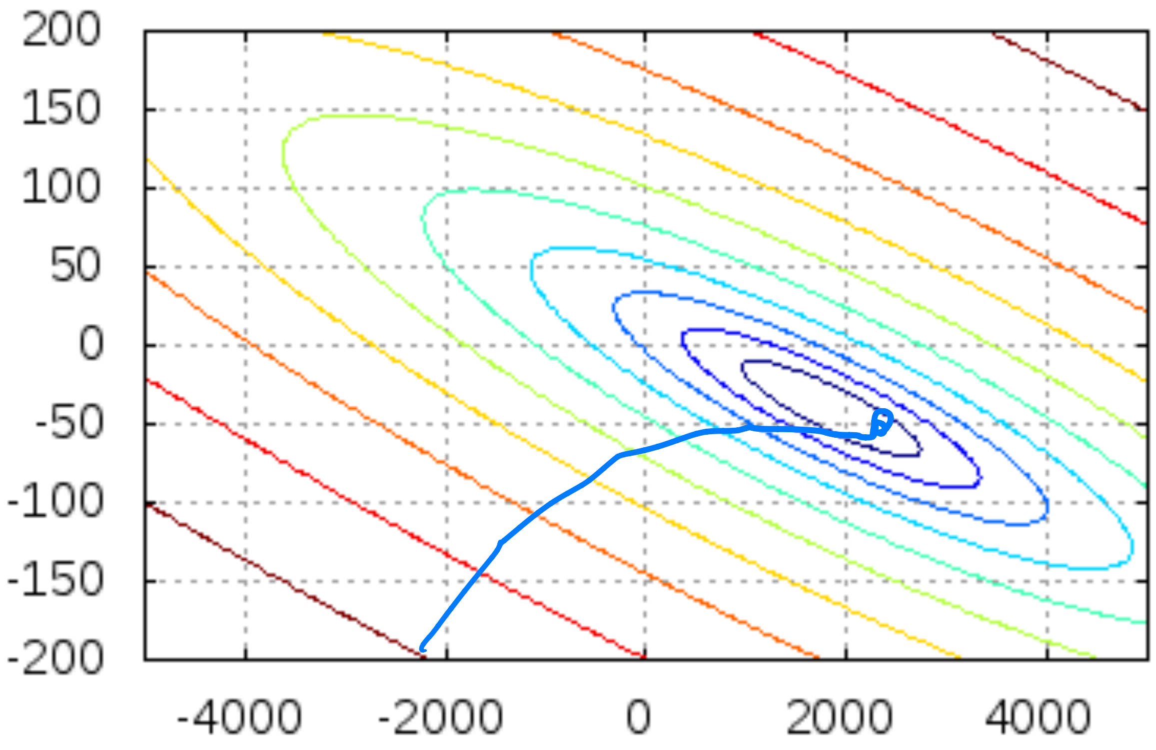
- About the learning rate:
  - If learning rate is too small: slow convergence
  - If learning rate is too large: oscillations around local minimum
  - Use learning rate scheduling with smaller learning rate over time
- Why SGD over full batch gradient descent
  - ① At the beginning of learning all gradients $\nabla E_n(\mathbf{w})$ will roughly point in the same general direction. Full batch gradient descent computes redundant number of gradients!

    Efficient + accurate enough

  - ② SGD is more likely to escape a local minimum since $\nabla E(\mathbf{w}) = 0$ does **not** necessarily imply $\nabla E_n(\mathbf{w}) = 0$!

Gradient Descent vs. Stochastic Gradient Descent

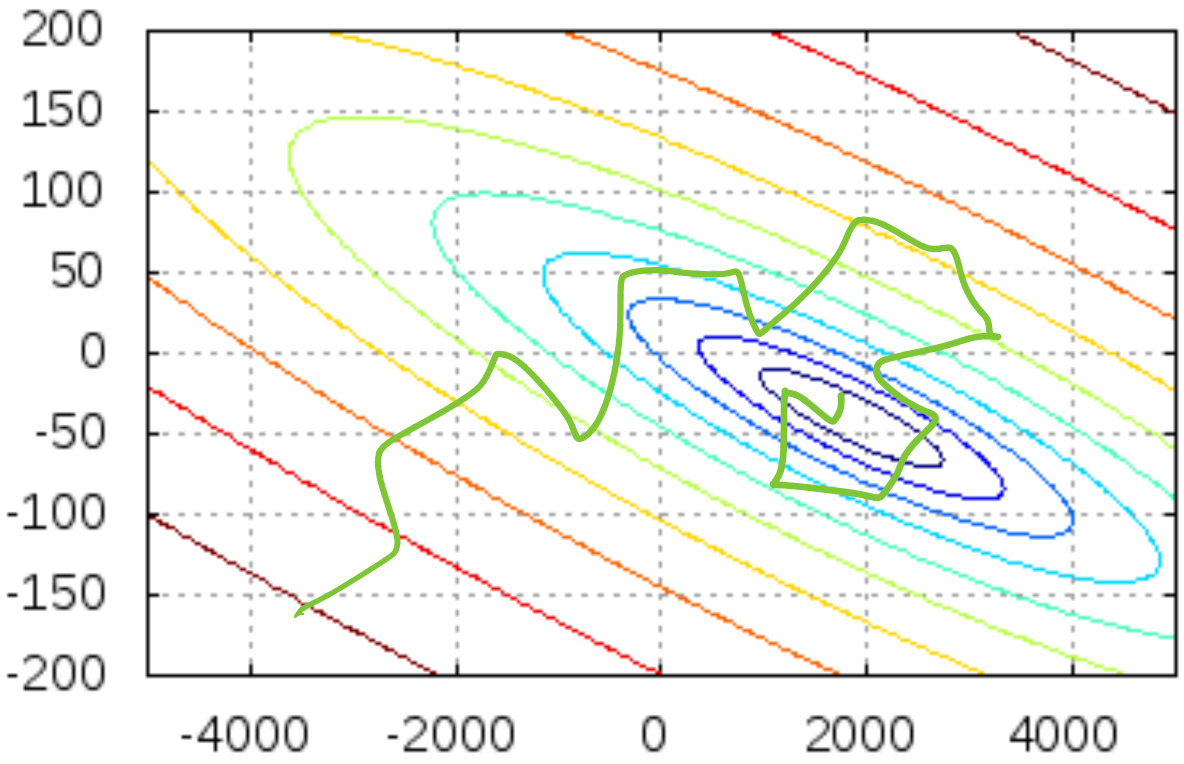
GD



$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

+ requires less iterations
- each iteration is slow to compute

SGD



$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)})$$

- requires more iterations
+ each iteration is fast to compute

Example: Test Errors and Local Minima

Restart for different random initial $\mathbf{w}^{(0)}$ to end up in different local minima

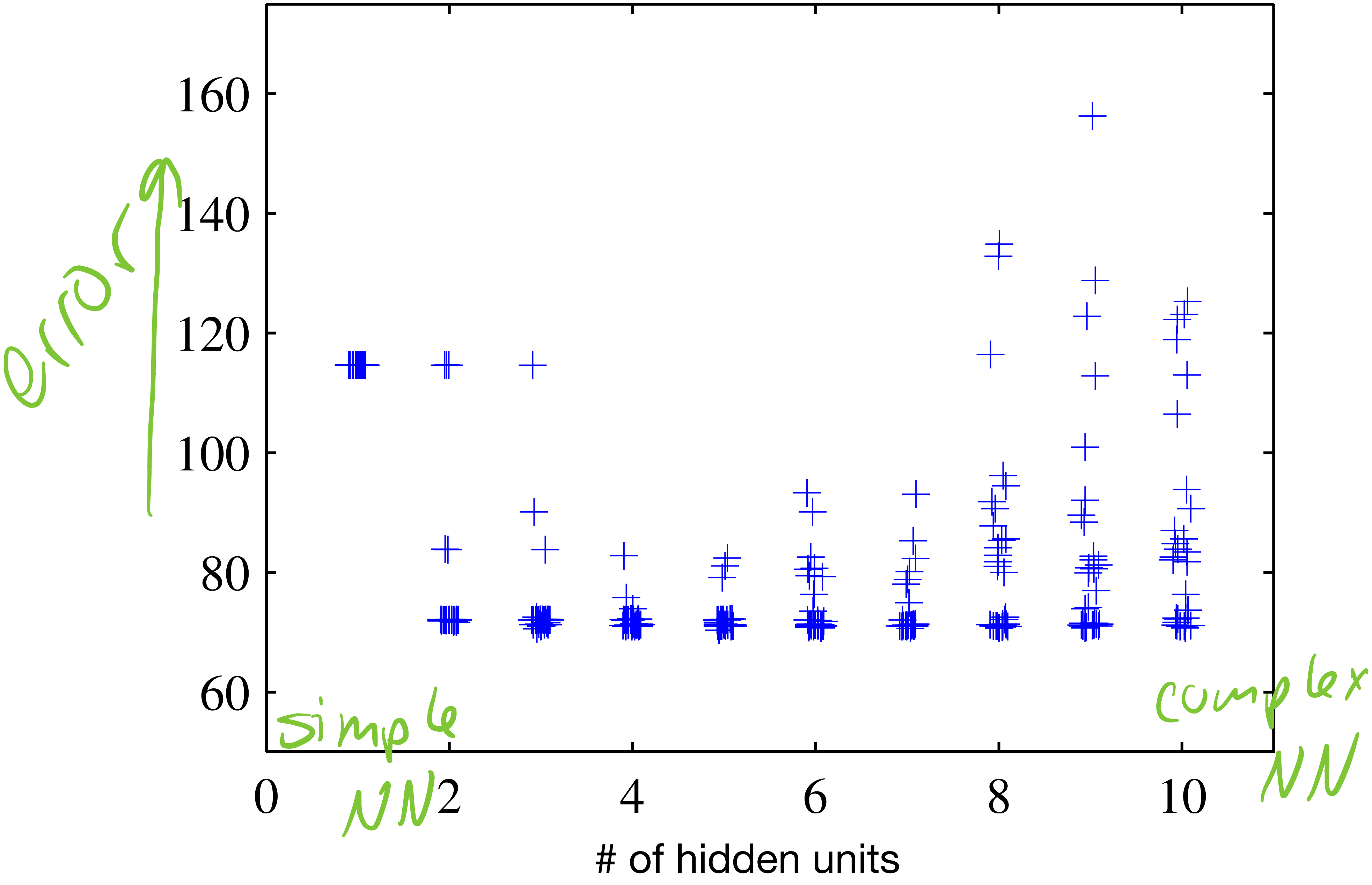


Figure: sum-of-squares test error vs. network size (# of hidden units) for 30 random starts each (Bishop 5.10)