UNIVERSITY OF AMSTERDAM
Informatics Institute

AMLAB
Amsterdam
Machine Learning Lab

Machine Learning 1

Lecture 9 - Unsupervised Learning

Latent Variable Models - K-means clustering - Lagrange Multipliers - Gaussian Mixture Models - Expectation Maximization

Erik Bekkers

Catch-up: NN training with error backpropagation

Image credit: Kirillm | Getty Images

University of Amsterdam
Informatics Institute

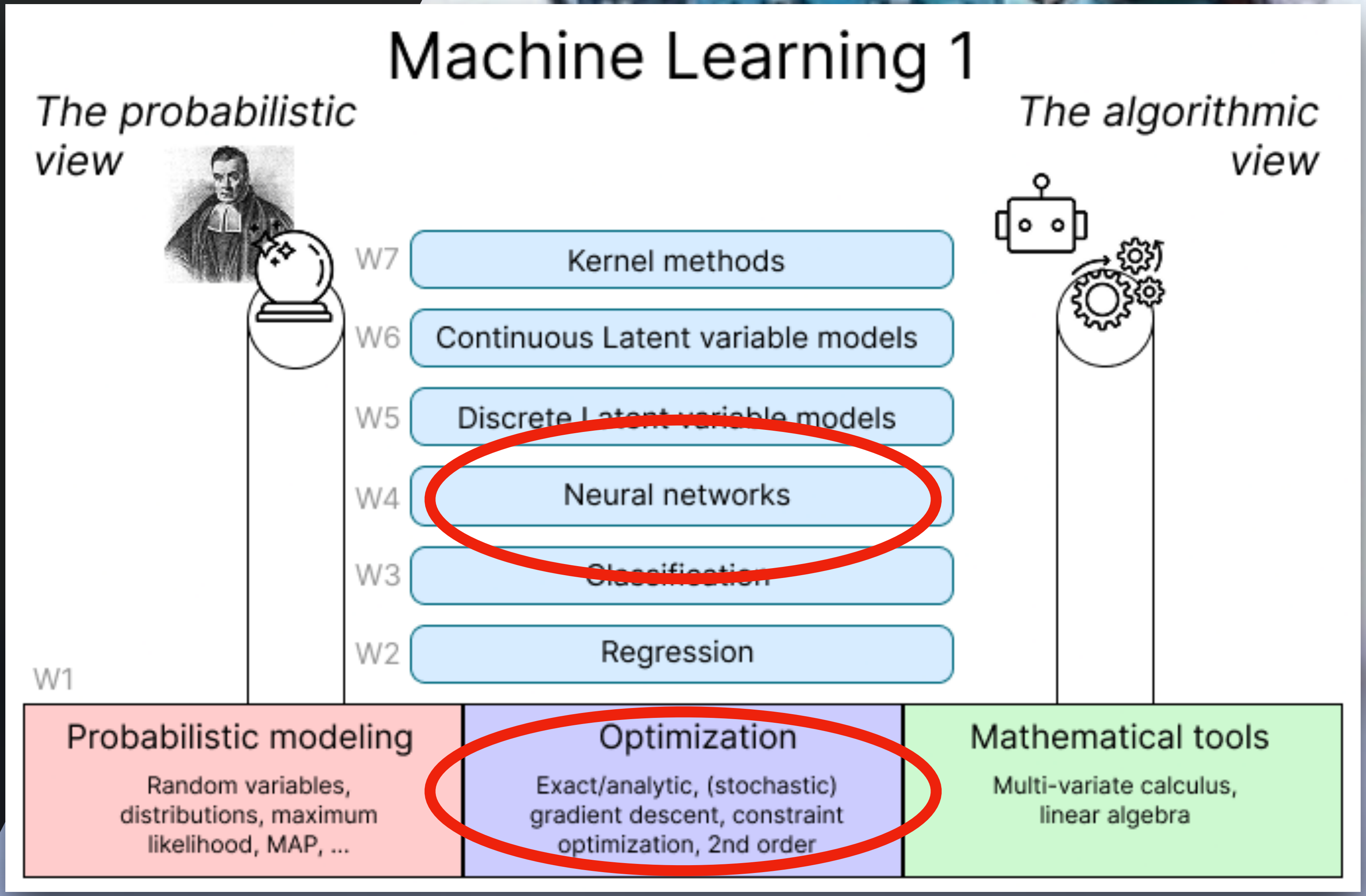
AMLAB Amsterdam Machine Learning Lab

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

Neural Networks: Parameter Optimization

- For each task a different loss function $E(\mathbf{w})$
- Optimal parameters $\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}} E(\mathbf{w})$
- Problem: $E(\mathbf{w})$ is not convex in $\mathbf{w}$, so several local minima can exist.
- How to reach the global minimum?

(practically) Never

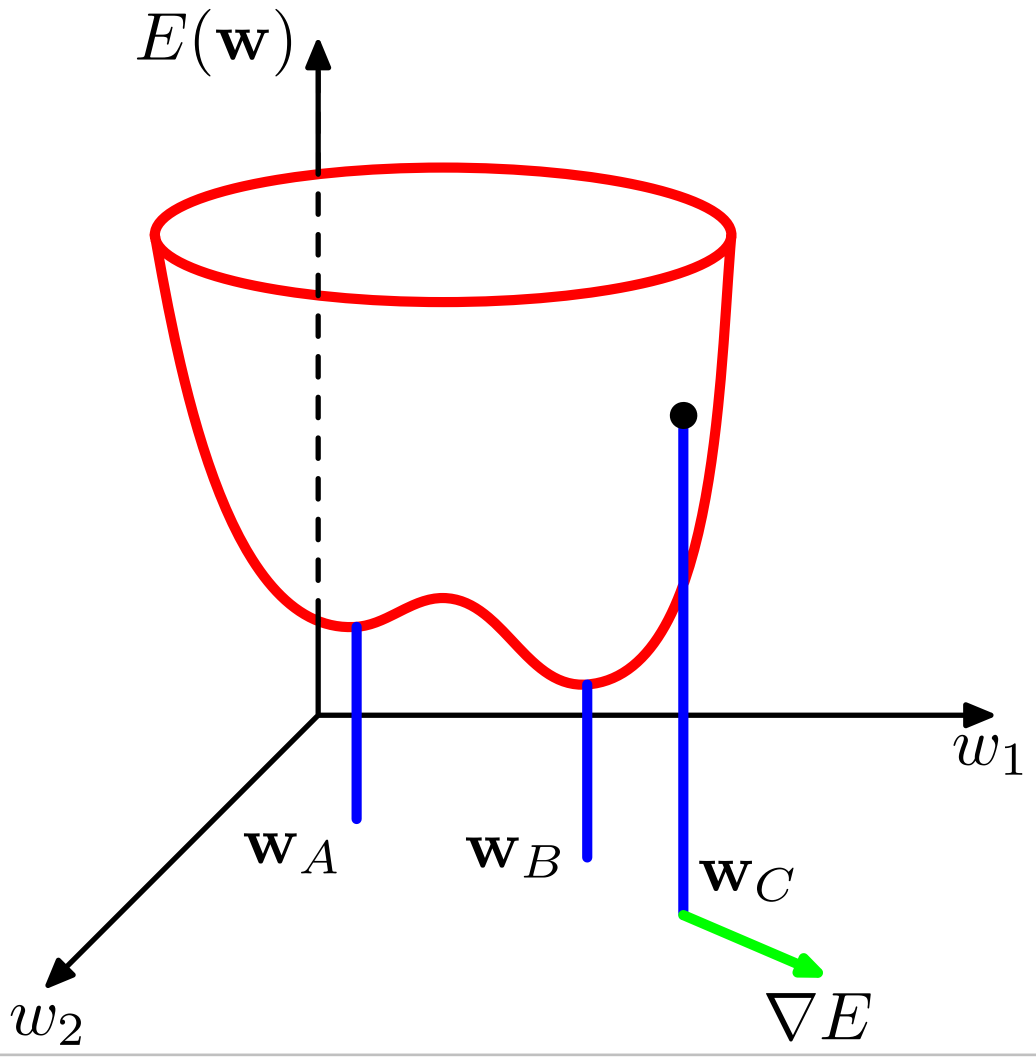


Figure: $E(\mathbf{w})$ as surface in weight space (Bishop 5.4)

Gradient Descent vs. Stochastic Gradient Descent

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

using all data

- Will easily get stuck in local minimum where $\nabla E(\mathbf{w}) = 0$

$D = \{x_1, x_2, \ldots, x_N\}$

- Use $E(\mathbf{w}) = \sum_{n=1}^{N} E_n(\mathbf{w}))$ to implement SGD:
  - Carefully choose learning rate $\eta > 0$
  - Randomly initialize $\mathbf{w}^{(0)}$
  - Randomly/sequentially choose $\mathbf{x}_n$ and update $\mathbf{w}$:

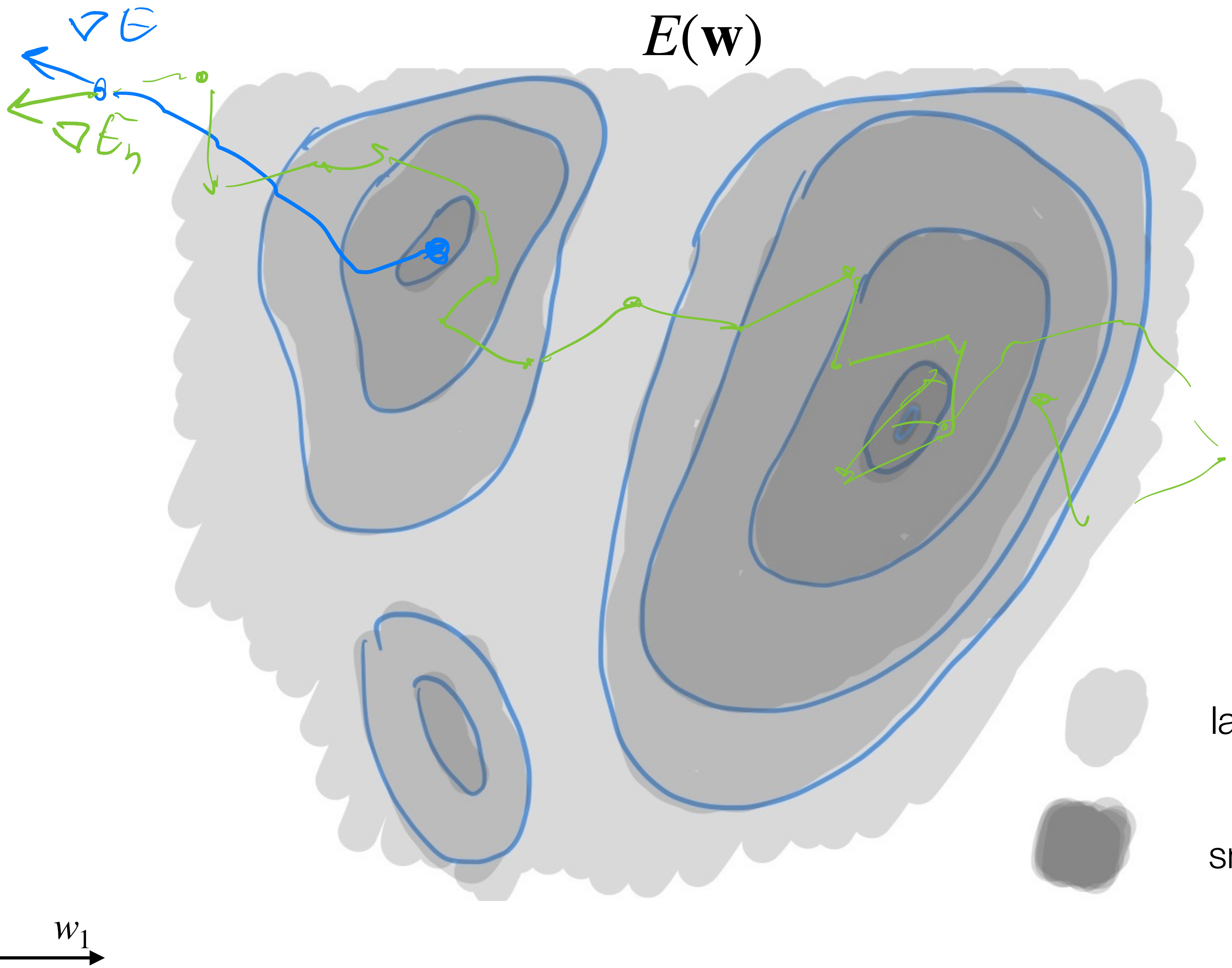
$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n(\mathbf{w}^{(\tau)})$$

estimate ∇E using only one sample

  - Convergence to area around local minimum

- Can also use minibatches $\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla \sum_{i=1}^{M} E_i(\mathbf{w}^{(\tau)})$

Gradient Descent vs. Stochastic Gradient Descent



Gradient Descent vs. Stochastic Gradient Descent

- About the learning rate:

  η

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

Example: Test Errors and Local Minima

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

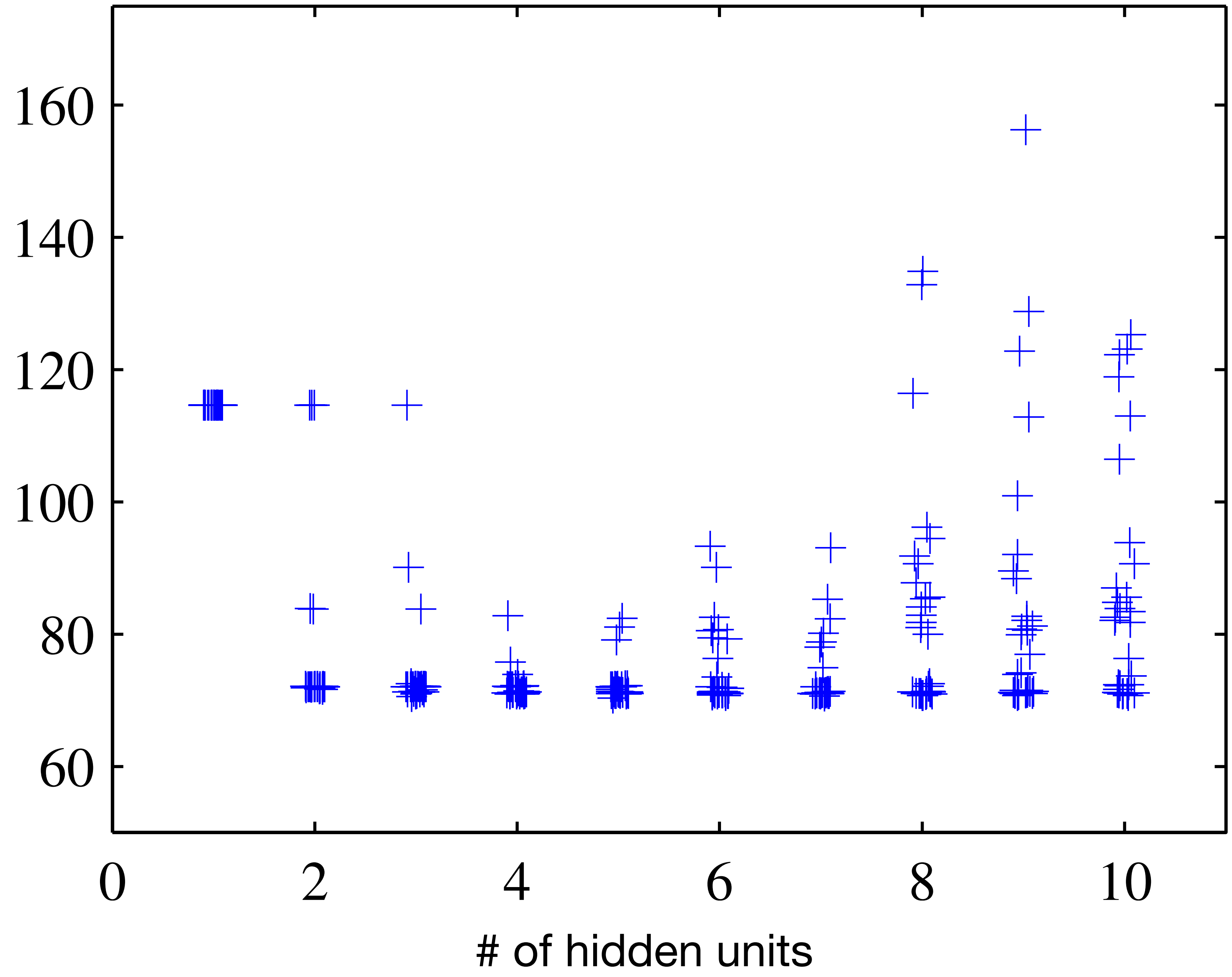
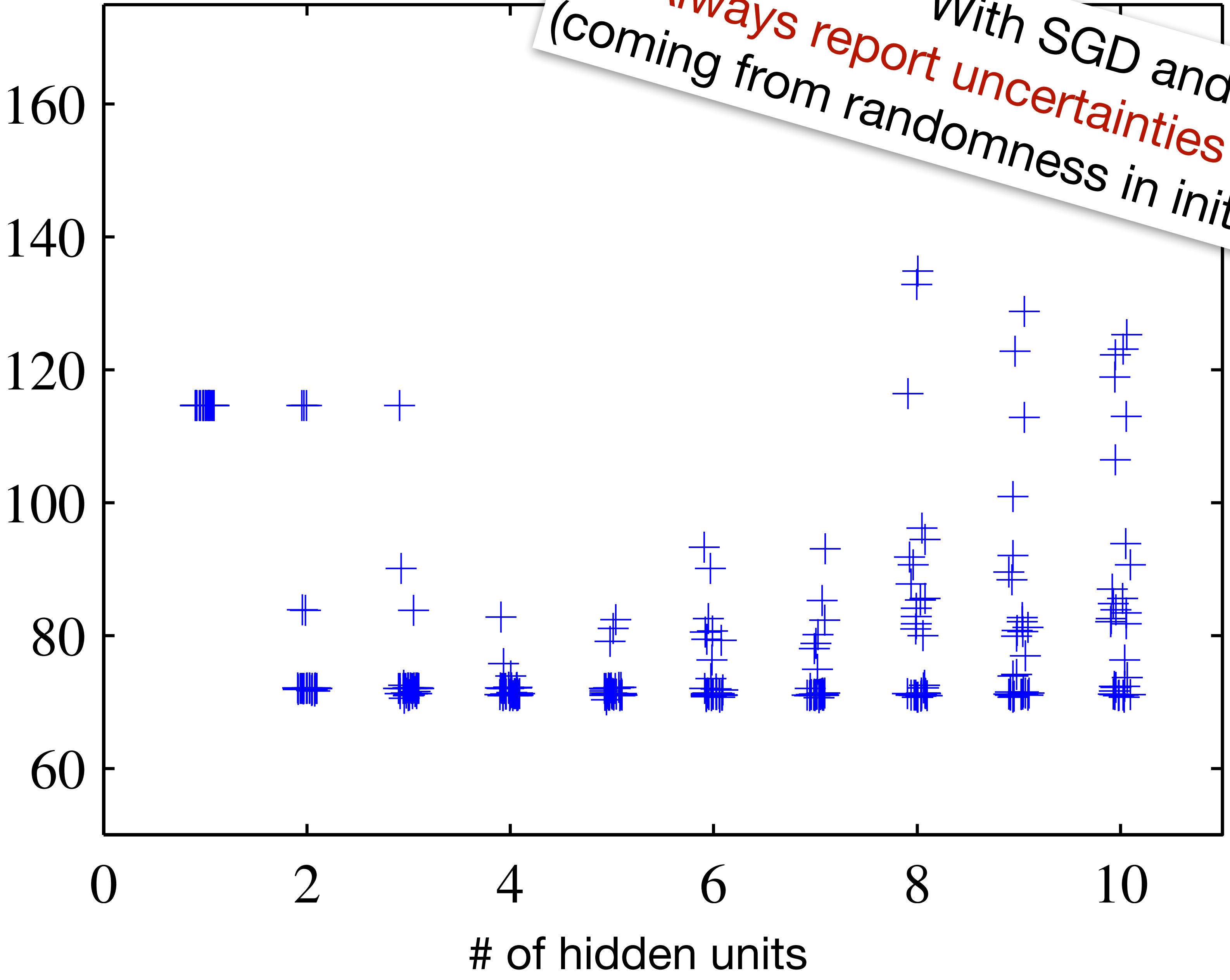


Figure: sum-of-squares test error vs. network size (# of hidden units) for 30 random starts each (Bishop 5.10)

Example: Test Errors and Local Minima

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



With SGD and NN:
Always report uncertainties on performance
(coming from randomness in initialization and SGD)

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

University of Amsterdam
Informatics Institute

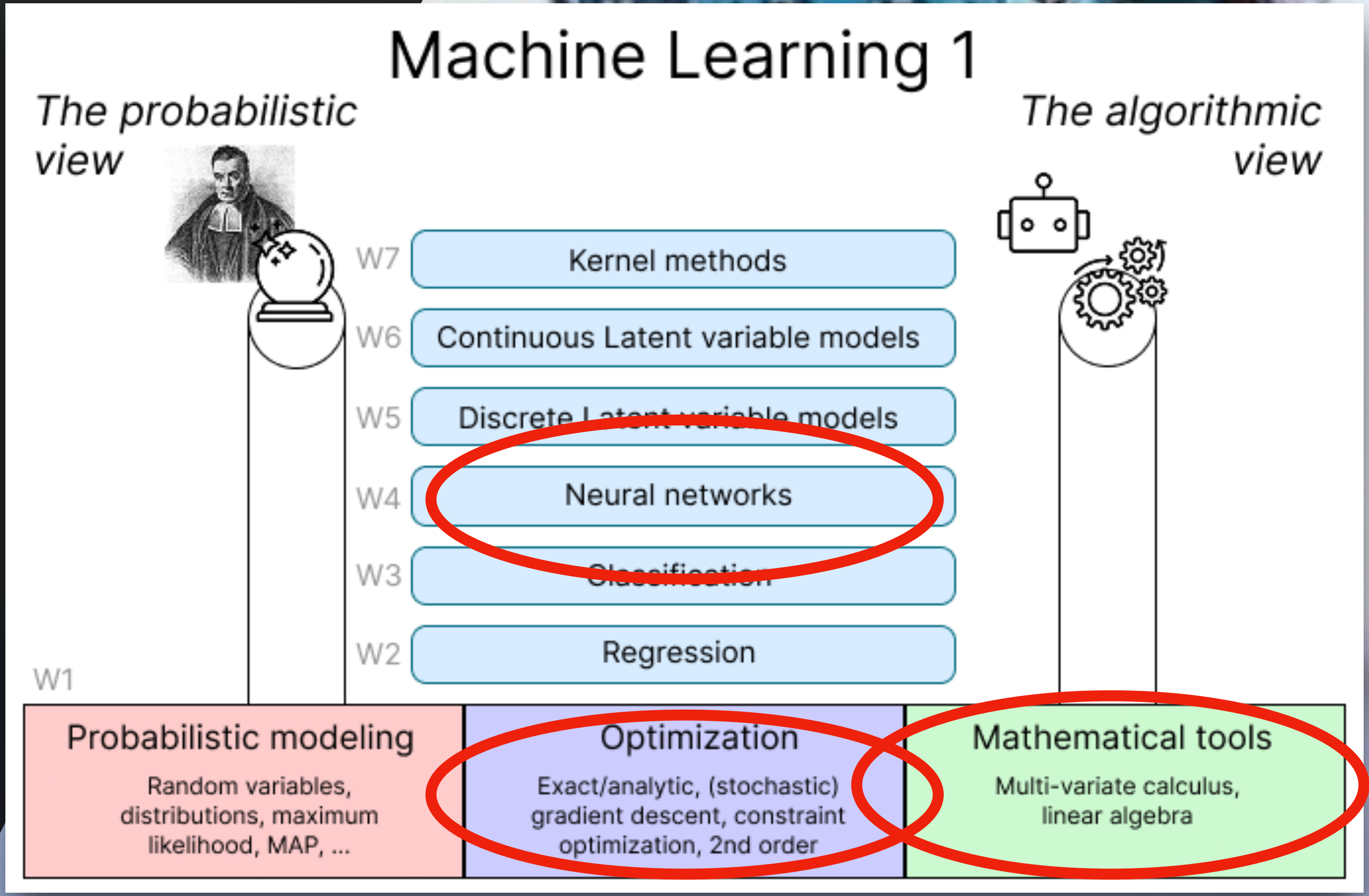
AMLAB
Amsterdam
Machine Learning Lab

Machine Learning 1

Lecture 8.5 - Supervised Learning
Neural Networks - Error Backpropagation

Erik Bekkers

(Bishop 5.3)



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, ...

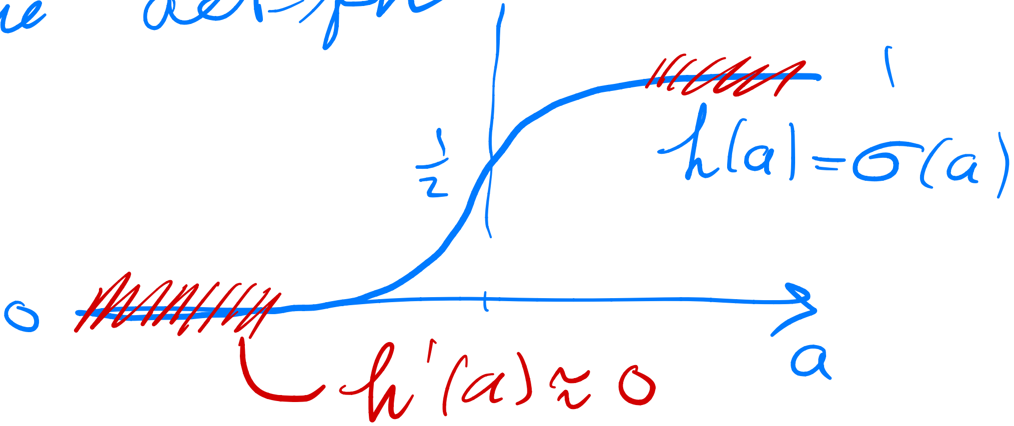
Optimization
Exact/analytic, (stochastic) gradient descent, constraint optimization, 2nd order

Mathematical tools
Multi-variate calculus, linear algebra

Slide credits: Patrick Forré and Rianne van den Berg

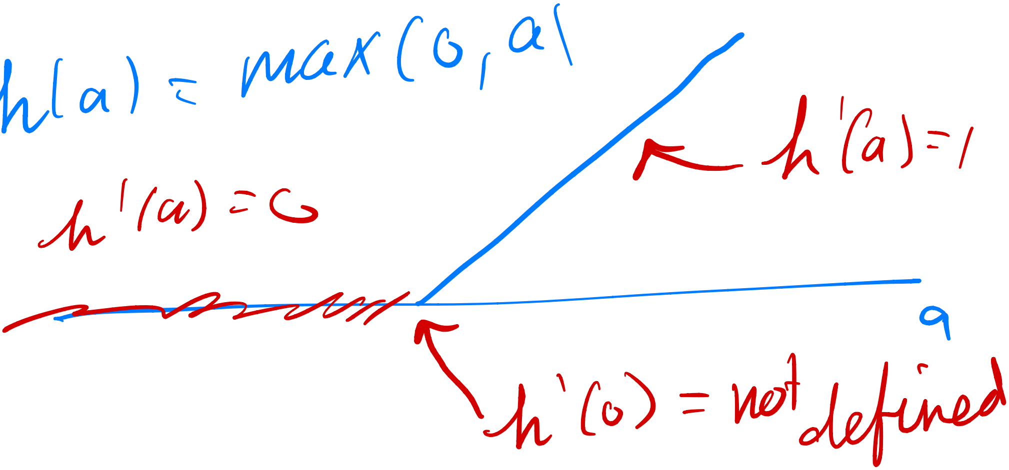
Image credit: Kirillm | Getty Images

Backprop depends on $h'(a)$, the derivative of the act fn



Better choice: ReLU

$h(a) = \max(0, a)$



Forward and Backward Propagation (with skip connection)

- Consider NN with skip connection (skipping at most 1 layer) with index notation:
    - i for the nodes that are used twice, once from i to j and once from i to k
    - j for the "regular" nodes
    - k for the "output nodes" of such a skip-connect layer
- The the forward pass of this layer

$$a_j = \sum_i h(a_i) w_{ji}$$

$$a_k = \sum_j h(a_j) w_{kj} + \sum_i h(a_i) w_{ki}$$

- Then we have the following update rules

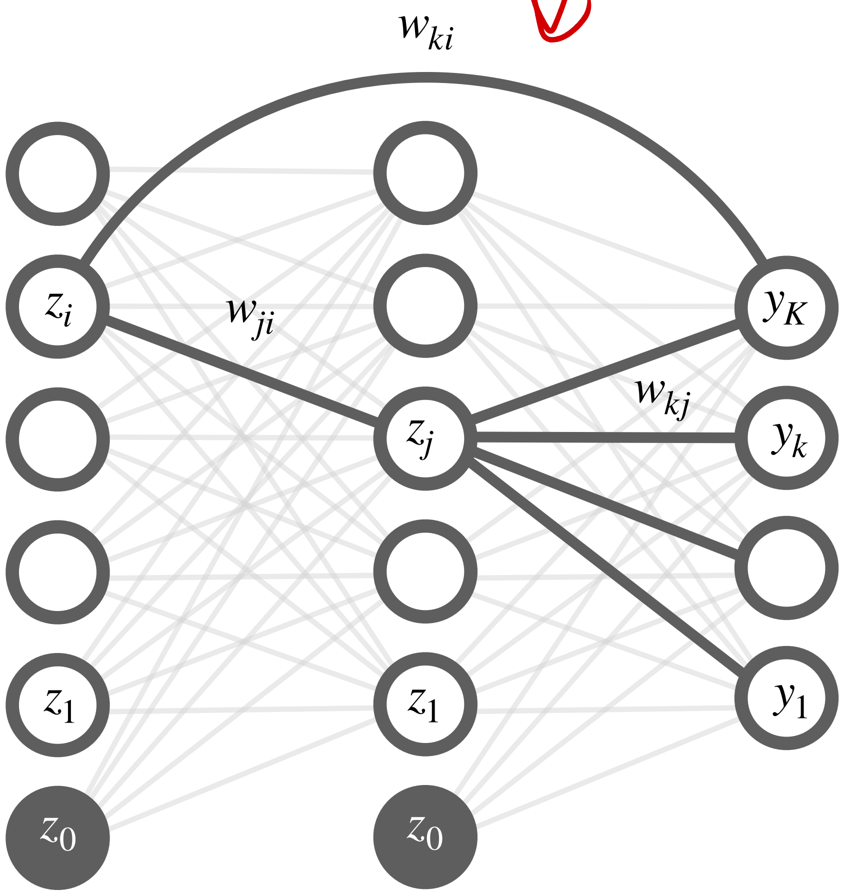
$$\delta_j = h'(a_j) \sum_k \delta_k w_{kj} \qquad \text{(the usual update rule)}$$

$$\delta_i = \delta_i^{(j)} + \delta_i^{(k)}, \quad \text{with} \quad \delta_i^{(j)} := h'(a_i) \sum_j \delta_j \, w_{ji}$$

$$\text{and} \quad \delta_i^{(k)} := h'(a_i) \sum_k \delta_k \, w_{ki}$$

$h(a) = a$

$h'(a) = 1$ $\downarrow$



Forward and Backward Propagation (with skip connection)

- Derivation

$$\delta_i = \frac{\partial E}{\partial a_i} = \sum_k \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial a_i}$$

$$= \sum_k \delta_k \left(\sum_j \frac{\partial a_k}{\partial a_j}\frac{\partial a_j}{\partial a_i} + \sum_{i'} \frac{\partial a_k}{\partial z_{i'}}\frac{\partial z_{i'}}{\partial a_i} \right)$$

$$= \sum_k \delta_k \left(\sum_j h'(a_j)\, w_{kj}\, h'(a_i)\, w_{ji} + w_{ki}\, h'(a_i) \right)$$

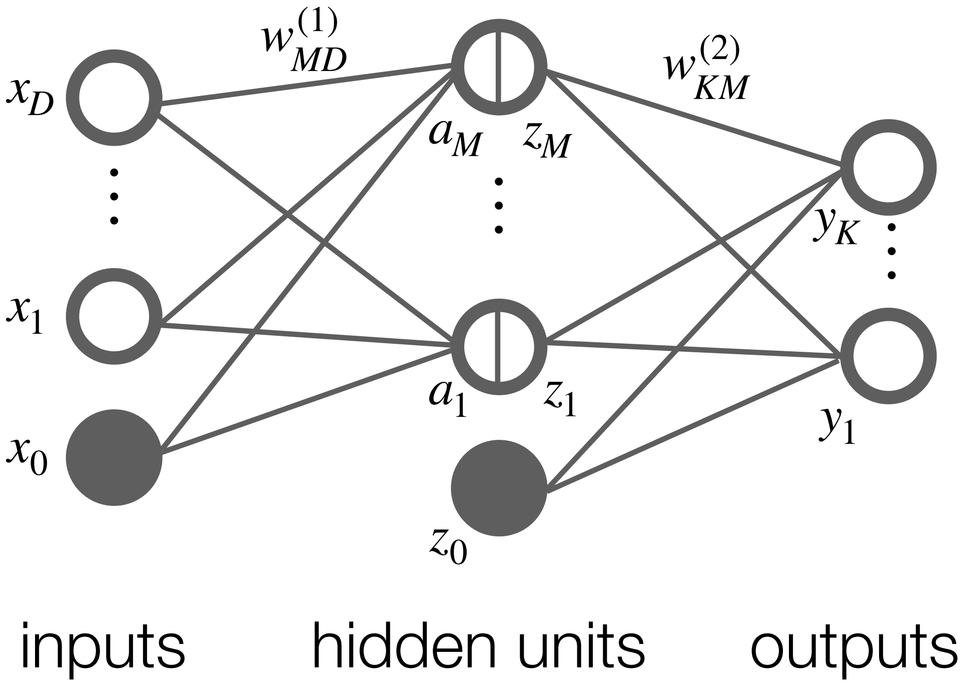
$$= h'(a_i) \sum_j \left(h'(a_j) \sum_k \delta_k\, w_{kj} \right) w_{ji} + h'(a_i) \sum_k \delta_k w_{ki}$$

$$= h'(a_i) \sum_j \delta_j\, w_{ji} + \delta_i^{(k)}$$

$$= \delta_i^{(j)} + \delta_i^{(k)}$$

Example: Backpropagation with tanh

- Two layer neural network:



- Regression with K outputs: $y_k = a_k^{(2)}$
- Hidden units: $z_j = h(a_j) = \tanh(a_j^{(1)})$
- Activation function $h(a) = \tanh(a) = \dfrac{e^a - e^{-a}}{e^a + e^{-a}}$
- Has derivative $h'(a) = 1 - h(a)^2$
- Error function $E = \dfrac{1}{2}\sum_k (y_k - t_k)^2$

- After forward propagation, compute:

$$\delta_k^{out} = y_k - t_k$$

- Backpropagate using:

$$\delta_j = (1 - z_j^2)\sum_{k=1}^{K} w_{kj}^{(2)}\delta_k^{out}$$

Update weights in first and second layer using:

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j x_i \quad \text{and} \quad \frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k^{out} z_j$$

University of Amsterdam
Informatics Institute

AMLAB
Amsterdam Machine Learning Lab

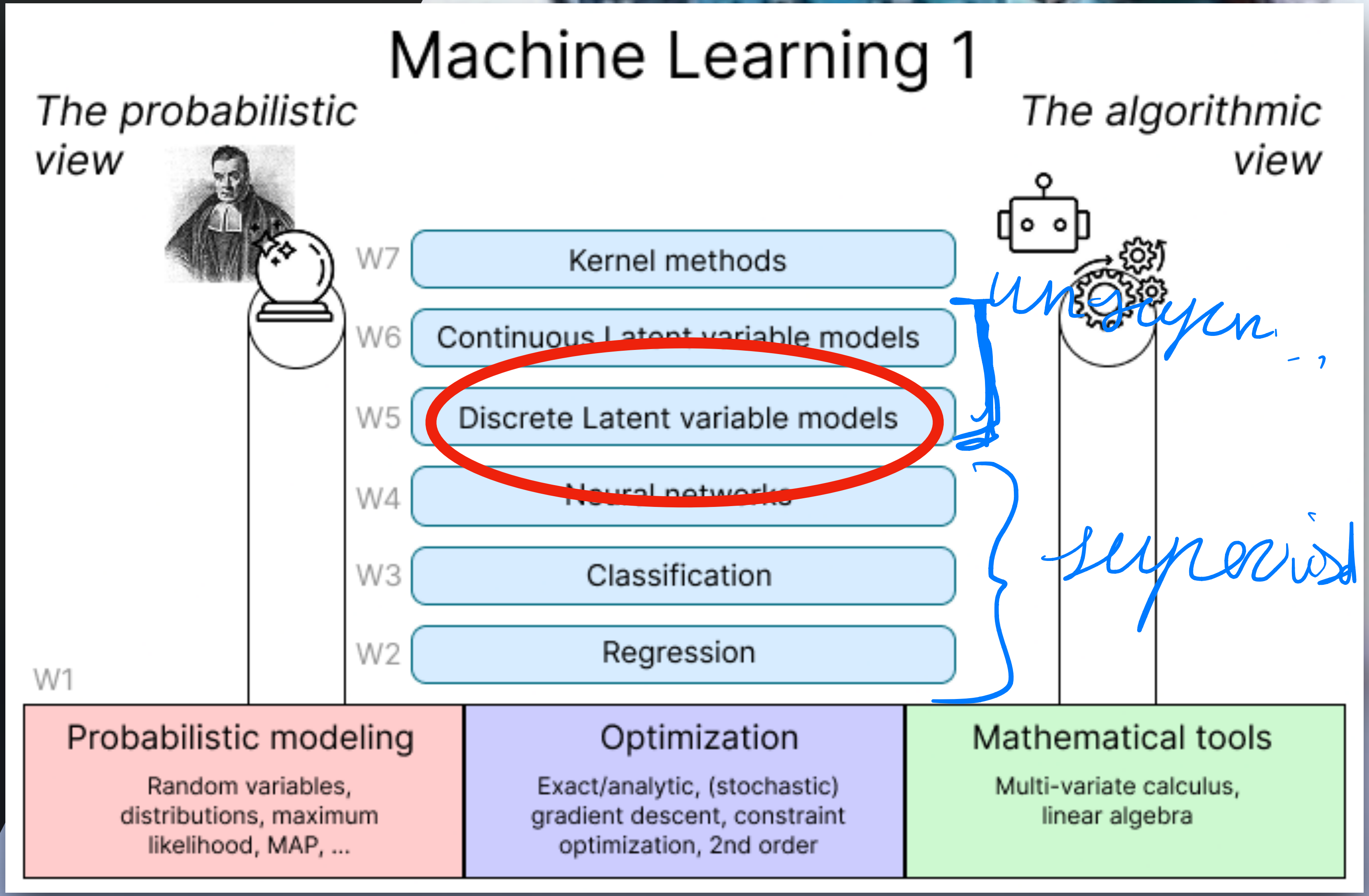
Machine Learning 1

Lecture 9.1 - Unsupervised Learning

Latent Variable Models

Erik Bekkers

(Bishop 9.0)



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

unsuperv...

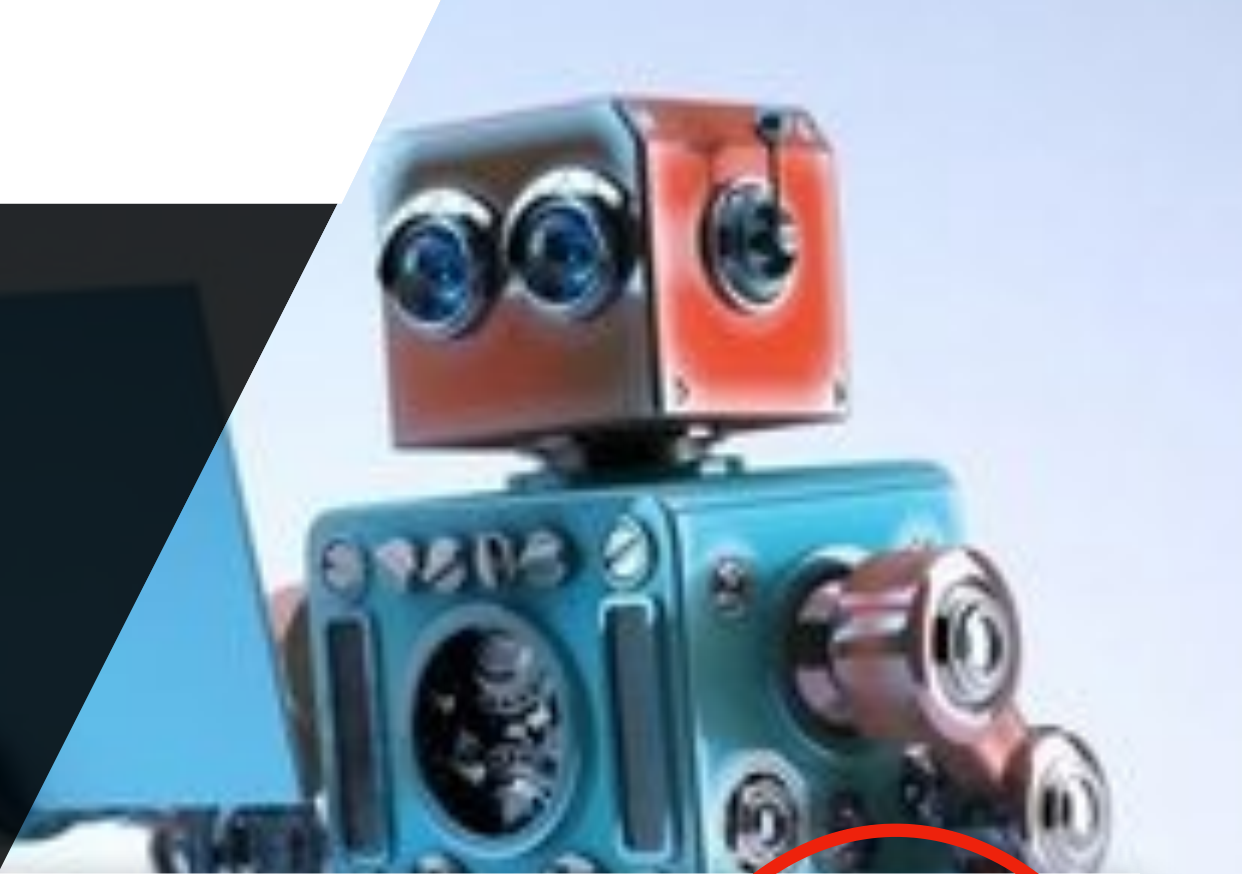
supervised

Slide credits: Patrick Forré and Rianne van den Berg

Image credit: Kirillm | Getty Images

UNIVERSITY OF AMSTERDAM
Informatics Institute

AMLAB Amsterdam Machine Learning Lab



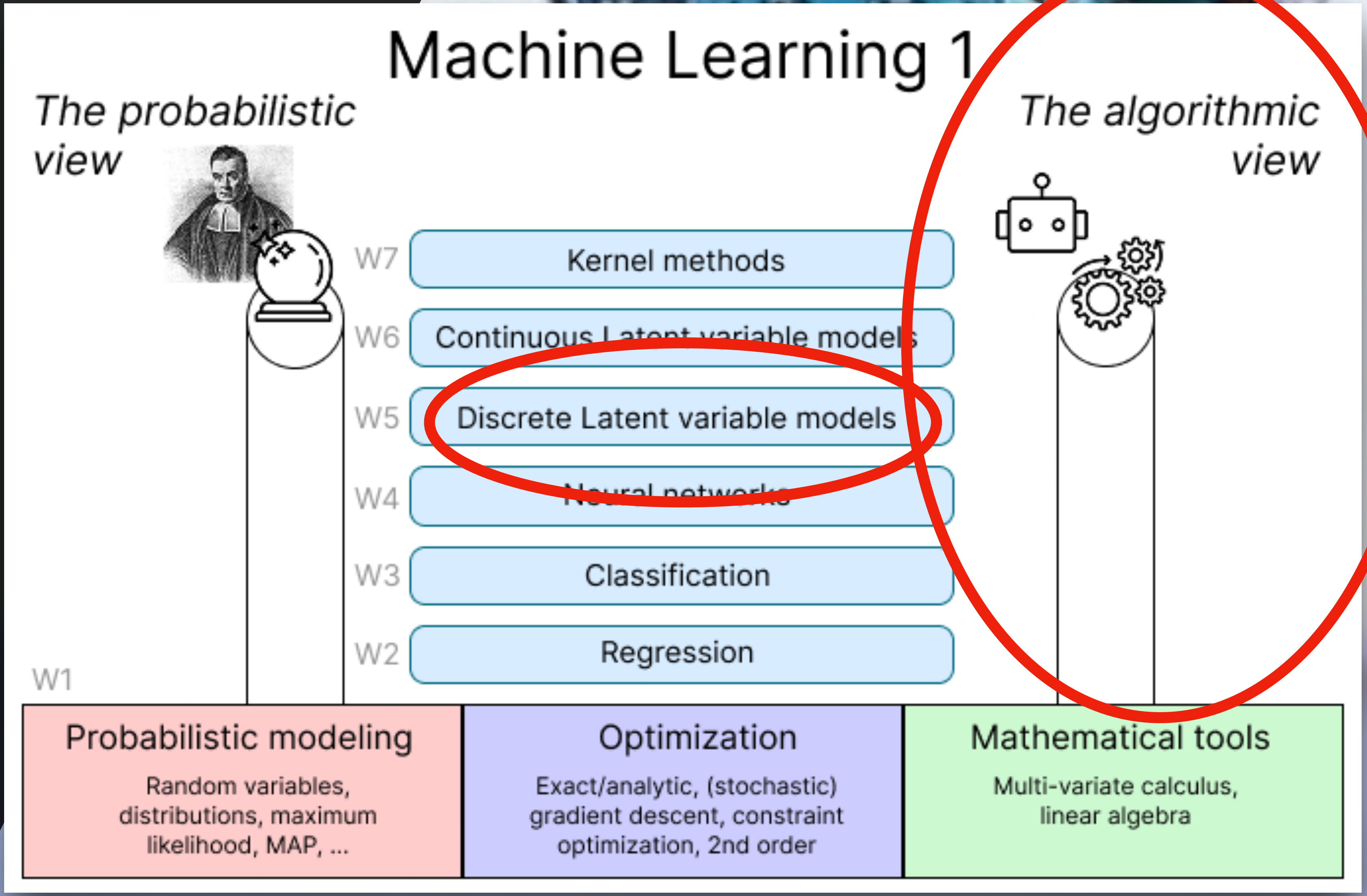
Machine Learning 1

Lecture 9.2 - Unsupervised Learning

K-Means Clustering

Erik Bekkers

(Bishop 9.1)



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

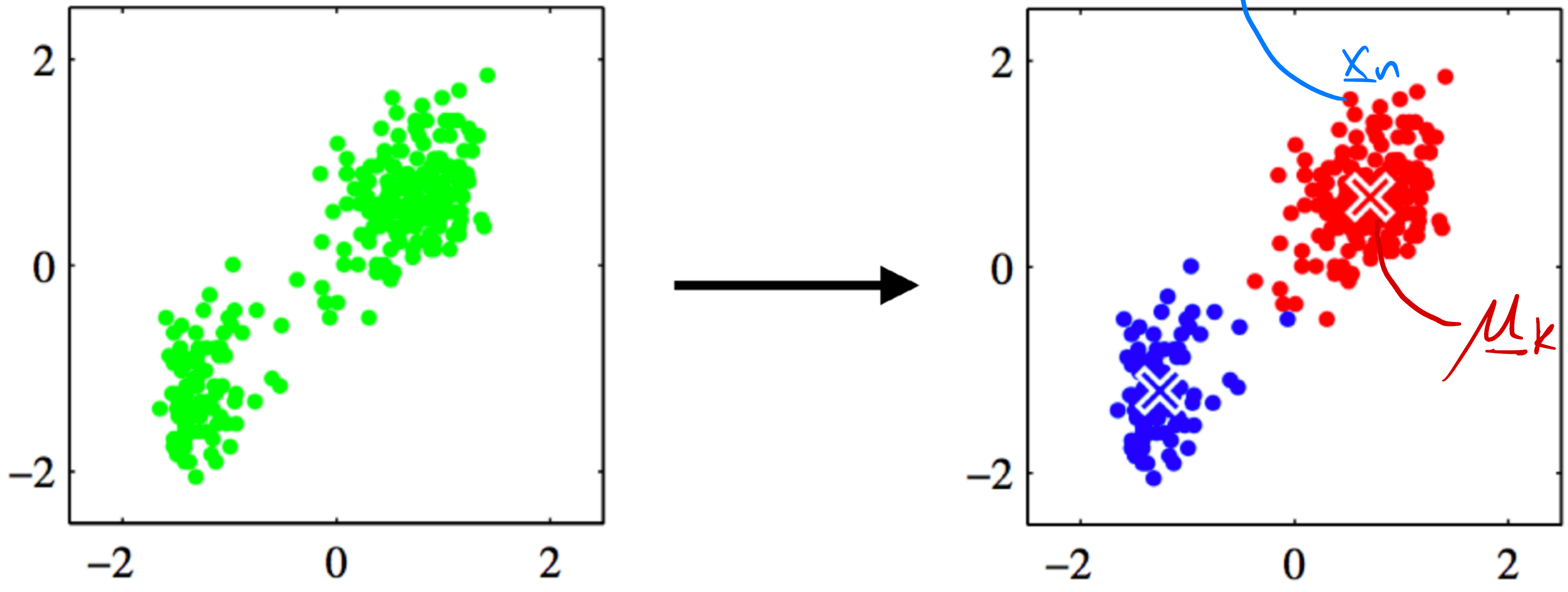
Image credit: Kirillm | Getty Images

Clustering with K-means

$z_{nk} = 1$ ← means n belongs to class k

- Data: a sample of points $\mathbf{x}$ (*without* a target)
- Goal: every single data point is assigned to a cluster – **a discrete latent variable**

$\underline{z}_n = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$



Dataset

Final clustering

Minimize J (EM algorithm)

Expectation step

"expected" class label

- Initialize with a random $\boldsymbol{\mu}_k \in \mathbb{R}^D$
- Repeat until convergence:
  - Find the assignment (fixed means) – **E-step**

$$z_{nk} = \begin{cases} 1 & \text{if } k = \underset{j}{\operatorname{argmin}} \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

hard assignment!

  - Find the means (fixed assignments) – **M-step**

$$\boldsymbol{\mu}_k = \frac{\sum_n z_{nk}\,\mathbf{x}_n}{\sum_n z_{nk}}$$

"Maximization step"

(actually minimization step in K-means)

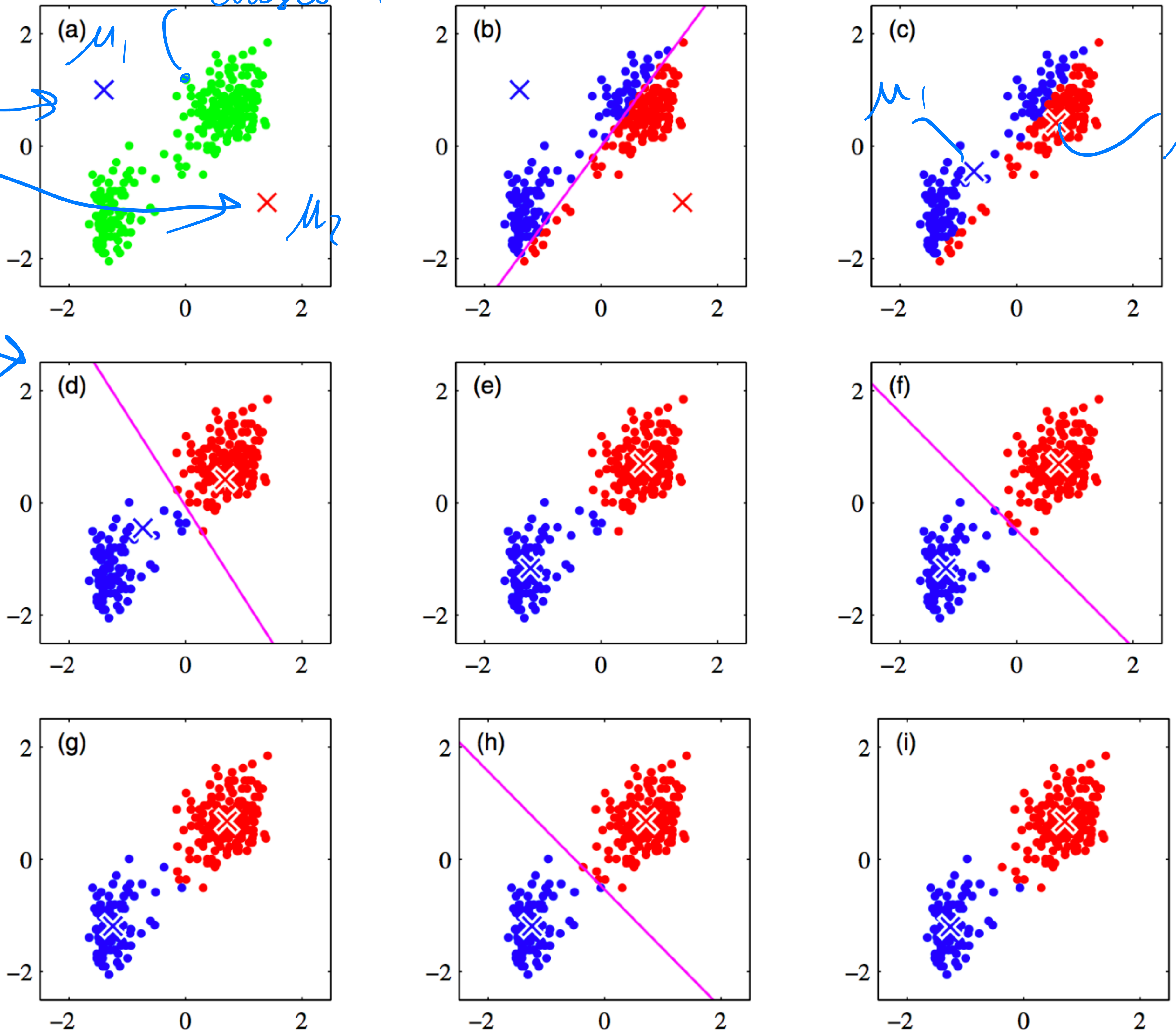
Example

E: assignment

M: maximization
min

random init

cluster 1



μ_1

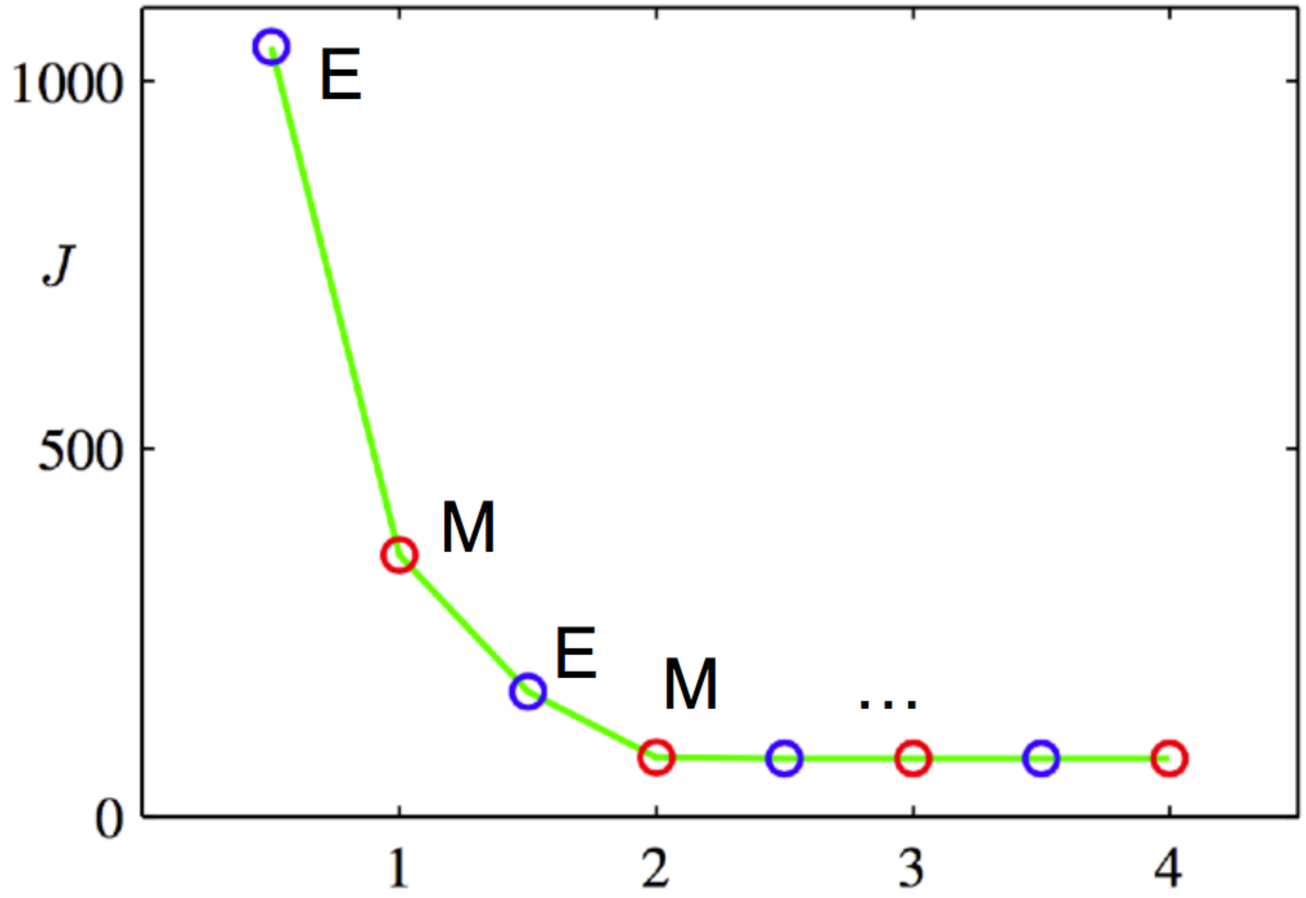
μ_2

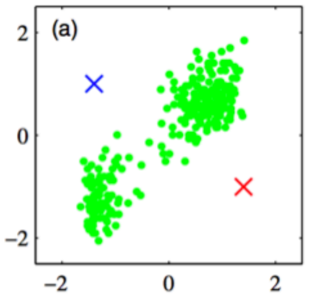
μ_1

μ_2

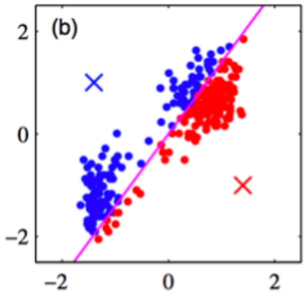
Convergence

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

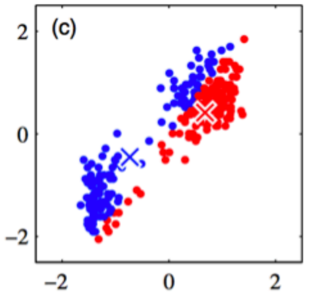




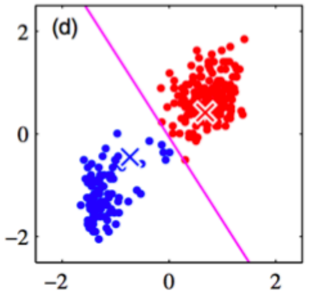
Initialize



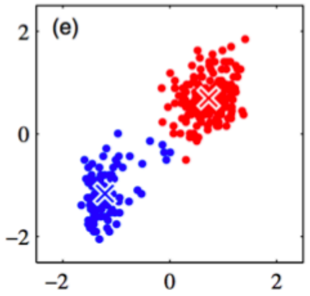
E-step



M-step



E-step

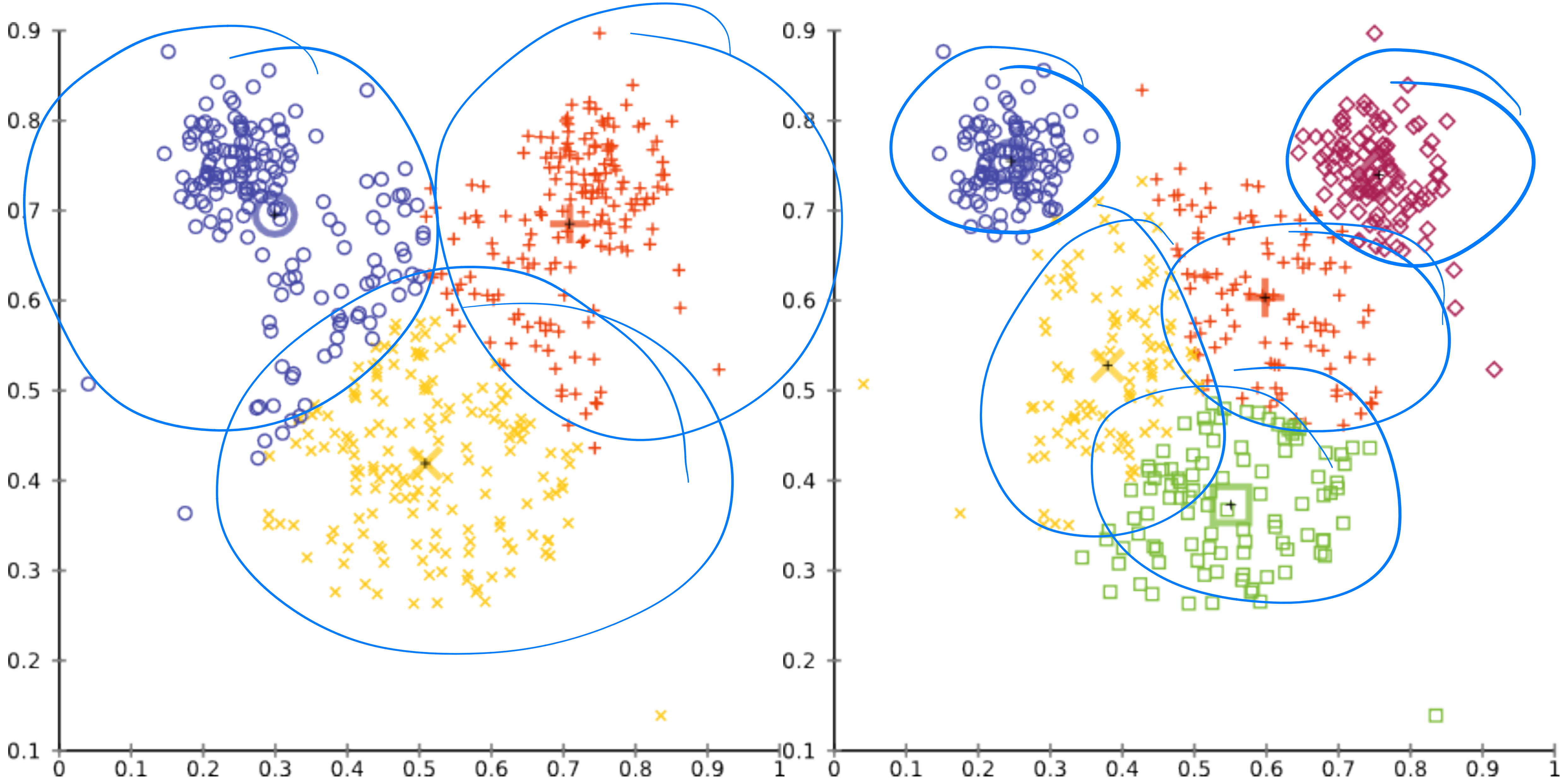


M-step

…

K-means cluster assume the same size for each cluster

Failures of K-means (the mouse data!)



K = 3

K = 5

Improvements

- Stochastic gradient for big data
  - For each datapoint, update nearest cluster mean:

$$\boldsymbol{\mu}_k = \boldsymbol{\mu}_k - \eta \left(\frac{\partial J}{\partial \boldsymbol{\mu}_k} \right)^T$$

$$= \underline{\boldsymbol{\mu}_k} + 2\eta(\boldsymbol{x}_n - \boldsymbol{\mu}_k)$$

instead of the M-step

- Other distances between points
  - Euclidean not always appropriate (discrete data), sensitive to outliers

$$\tilde{J} = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \mathscr{V}(\mathbf{x}_n, \boldsymbol{\mu}_k)$$

$\| x_n - \mu_k \|_2^2$

$\| x_n - \mu_k \|_1$ L_1 error less sensitive to outliers

general distance $d(x_n, \mu_k)$

University of Amsterdam
Informatics Institute

AMLAB
Amsterdam
Machine Learning Lab

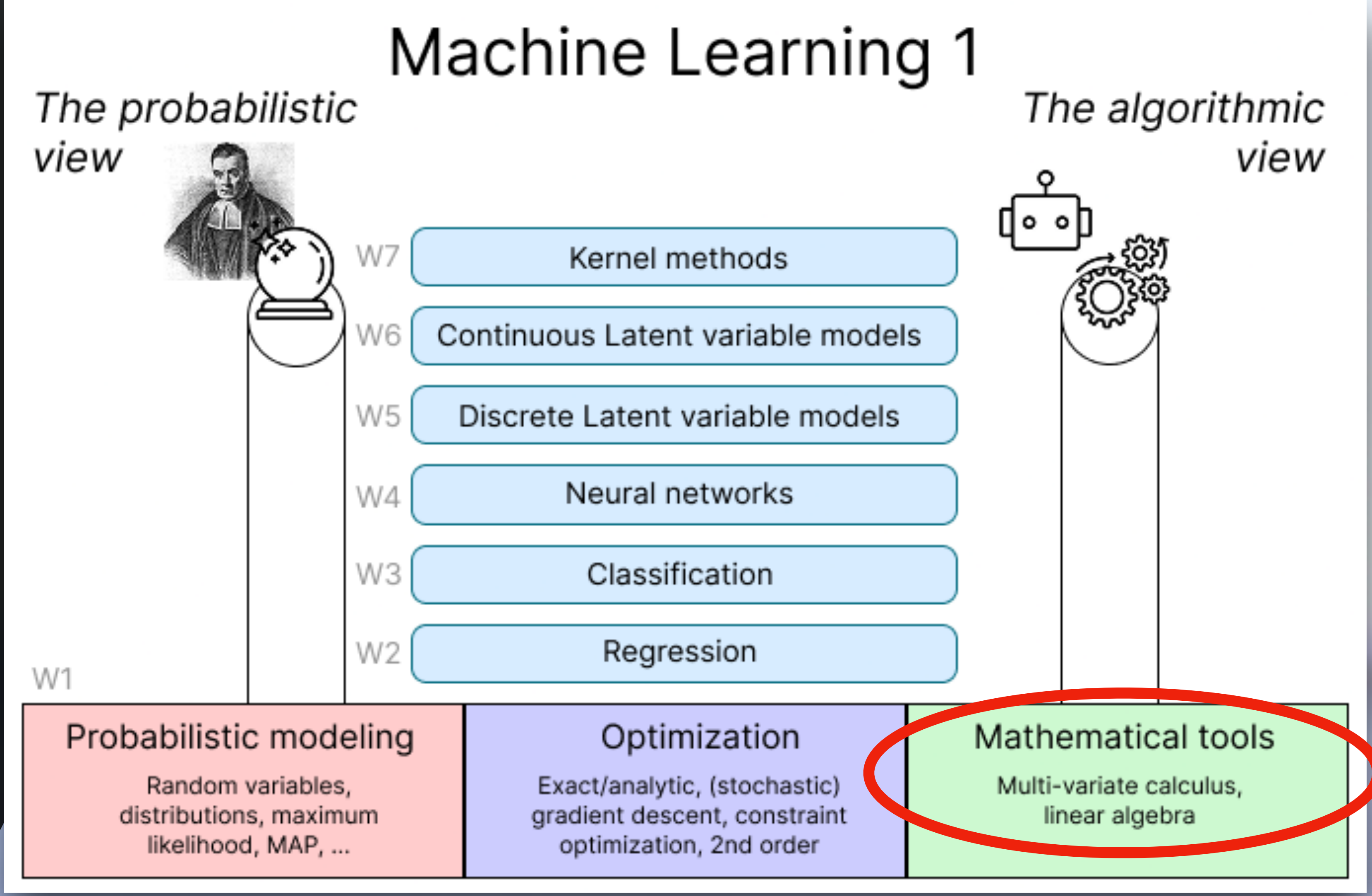
Machine Learning 1

Lecture 9.3 - Unsupervised Learning

Intermezzo: Lagrange Multipliers

Erik Bekkers

(Bishop Appendix E)



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

Constrained optimization

- So far we have been looking at **(unconstrained) optimization problems** of the form

$$\max_{\mathbf{w}} \quad f(\mathbf{w})$$

- *E.g., f could be an error function that depends on data and the model parameters:*

$$f(\mathbf{w}) = -E_D(\mathbf{w}) = -\sum_{(\mathbf{x}_n, t_n) \in D} \|y(\mathbf{x}_n, \mathbf{w}) - t_n\|^2$$

Constrained optimization

- So far we have been looking at **(unconstrained) optimization problems** of the form

$$\max_{\mathbf{w}} \quad f(\mathbf{w})$$

- *E.g., f could be an error function that depends on data and the model parameters:*

$$f(\mathbf{w}) = -E_D(\mathbf{w}) = -\sum_{(\mathbf{x}_n, t_n)\in D} \|y(\mathbf{x}_n, \mathbf{w}) - t_n\|^2$$

- If our solutions for $\mathbf{w}$ need to satisfy some constraint, we have to solve a **constrained optimization problem** of the form

$$\max_{\mathbf{w}} \quad f(\mathbf{w}) \qquad \text{subject to} \qquad g(\mathbf{w}) = c$$

- *E.g., g could enforce that the norm of $\mathbf{w}$ in a model $y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x}$ is equal to 1:*

$$g(\mathbf{w}) = \|\mathbf{w}\|^2 \qquad \text{and} \qquad c = 1$$

prior probs

$$\left.\{\pi_k\}_{k=1}^{K}\right\} \qquad \underbrace{\sum_{k=1}^{K} \pi_k} = \underbrace{1}$$

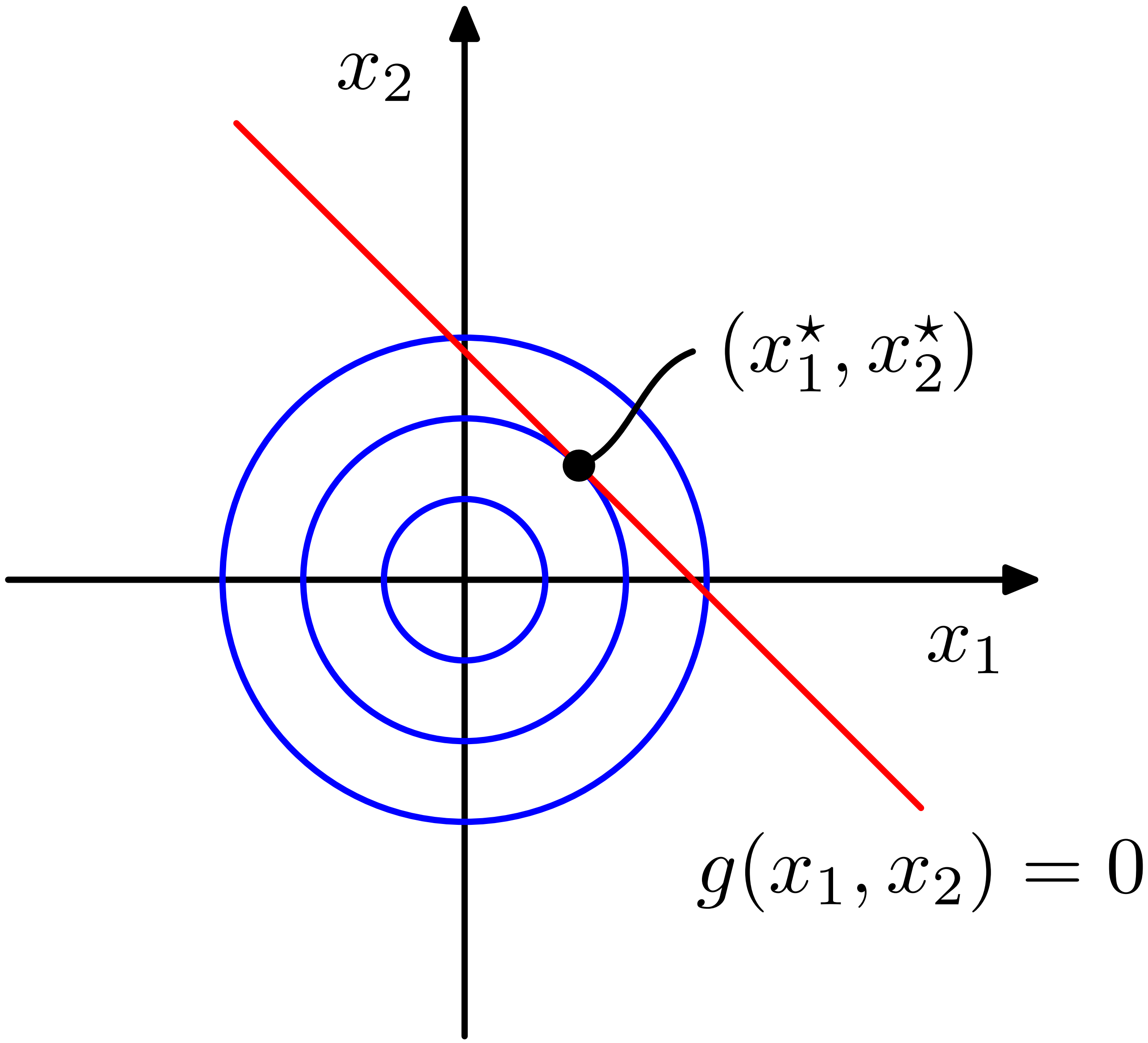
$$g(\{\pi_k\}) = c$$

Constrained optimization

- **Problem**:

$$\max_{x_1, x_2} f(x_1, x_2) \qquad \text{subject to} \quad g(x_1, x_2) = 0$$

with
$$f(x_1, x_2) = 1 - x^2 - x_2^2$$
$$g(x_1, x_2) = x_1 + x_2 - 1$$



- **Lagrangian**

$$L(x_1, x_2, \lambda) = 1 - x_1^2 - x_2^2 + \lambda(x_1 + x_2 - 1)$$

- **Stationary points satisfy**:

$$\frac{\partial}{\partial x_1} L(x_1, x_2, \lambda) = -2x_1 + \lambda = 0$$

$$\frac{\partial}{\partial x_2} L(x_1, x_2, \lambda) = -2x_2 + \lambda = 0$$

$$\frac{\partial}{\partial \lambda} L(x_1, x_2, \lambda) = x_1 + x_2 - 1 = 0$$