

# Syllabus

Fri. 27.10. (1) 0. Introduction

## A. Supervised Learning: Linear Models & Fundamentals

- Fri. 3.11. (2) A.1 Linear Regression
- Fri. 10.11. (3) A.2 Linear Classification
- Fri. 17.11. (4) A.3 Regularization
- Fri. 24.11. (5) A.4 High-dimensional Data

## B. Supervised Learning: Nonlinear Models

- Fri. 1.12. (6) B.1 Nearest-Neighbor Models
- Fri. 8.12. (7) B.4 Support Vector Machines
- Fri. 15.12. (8) B.3 Decision Trees
- Fri. 22.12. (9) B.2 Neural Networks
- *Christmas Break* —
- Fri. 12.1. (10) B.5 A First Look at Bayesian and Markov Networks

## C. Unsupervised Learning

- Fri. 19.1. (11) C.1 Clustering
- Fri. 26.1. (12) C.2 Dimensionality Reduction
- Fri. 2.2. (13) C.3 Frequent Pattern Mining
- Fri. 9.2. (14) Q&A

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

1 / 24

Machine Learning

# Outline

1. Network Topologies

2. Stochastic Gradient Descent (Backpropagation)

3. Regularization

# Outline

1. Network Topologies

2. Stochastic Gradient Descent (Backpropagation)

3. Regularization

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

Machine Learning

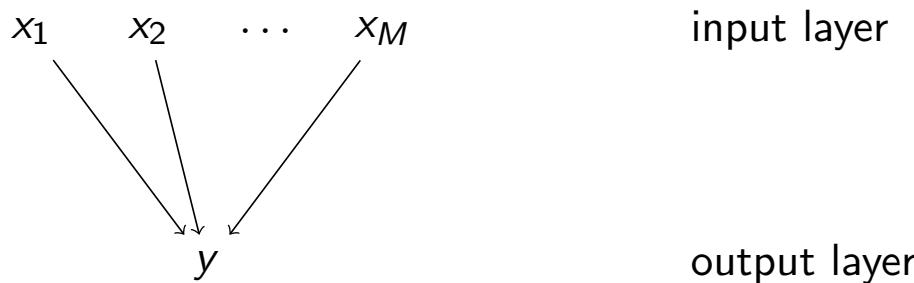
1 / 24

## Logistic Regression

logistic regression:

$$\hat{y}(x) := \hat{p}(y = 1 \mid x) = \text{logistic}(\beta^T x), \quad x \in \mathbb{R}^M$$

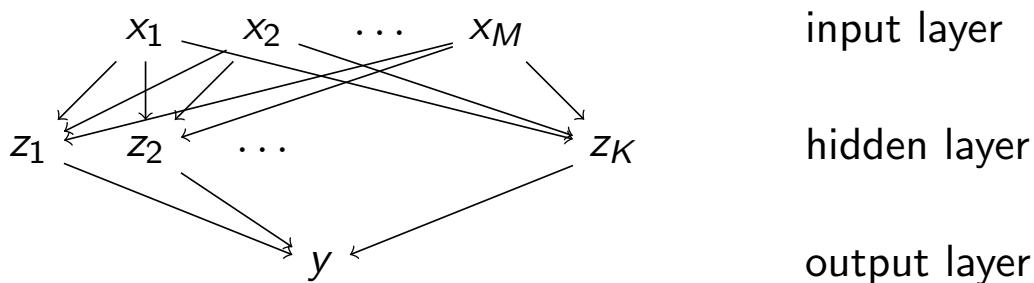
# Logistic Regression (0 hidden layers)



logistic regression:

$$\hat{y}(x) := \hat{p}(y = 1 \mid x) = \text{logistic}(\beta^T x), \quad x \in \mathbb{R}^M$$

# Feedforward Neural Network (1 hidden layer)

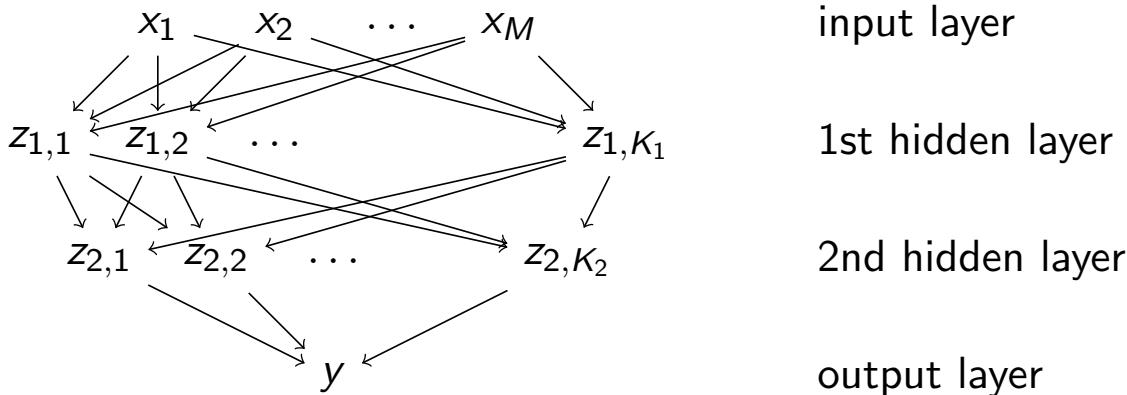


feedforward neural network (1 hidden layer):

$$\hat{y}(x) := \text{logistic}(\beta_2^T z(x))$$

$$z_k(x) := \text{logistic}(\beta_{1,k}^T x), \quad k = 1, \dots, K, x \in \mathbb{R}^M$$

# Feedforward Neural Network (2 hidden layers)



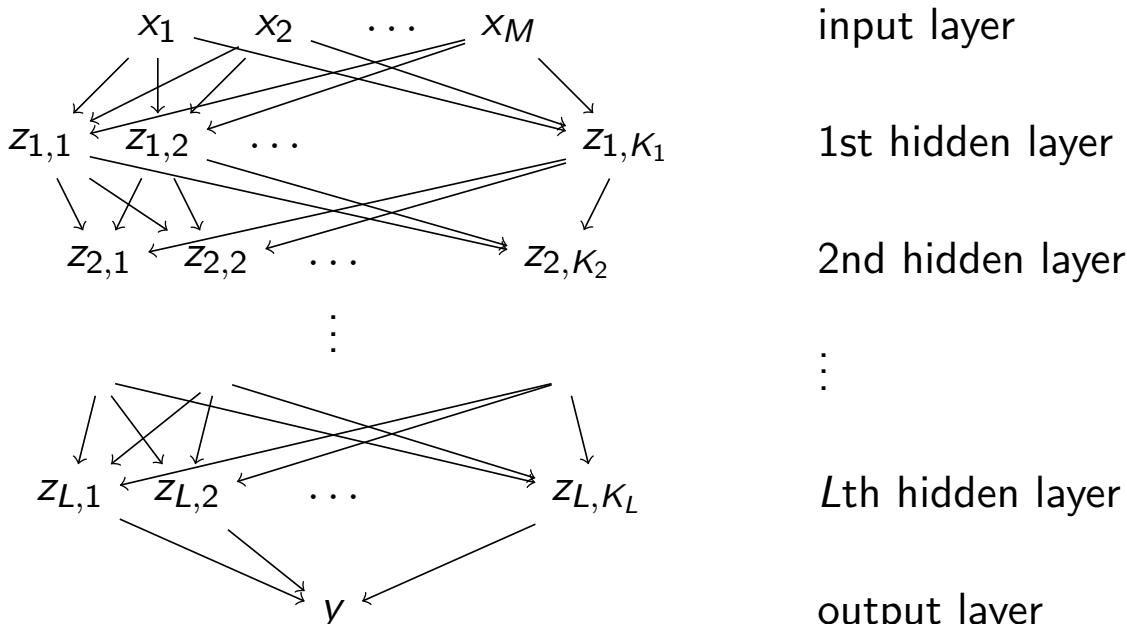
feedforward neural network (2 hidden layers):

$$\hat{y}(x) := \text{logistic}(\beta_3^T z_2(x))$$

$$z_{2,k}(x) := \text{logistic}(\beta_{2,k}^T z_1(x)), \quad k = 1, \dots, K_2$$

$$z_{1,k}(x) := \text{logistic}(\beta_{1,k}^T x), \quad k = 1, \dots, K_1, \quad x \in \mathbb{R}^M$$

# Feedforward Neural Network ( $L$ hidden layers)



$$\hat{y}(x) := \text{logistic}(\beta_{L+1}^T z_L(x))$$

$$z_{\ell,k}(x) := \text{logistic}(\beta_{\ell,k}^T z_{\ell-1}(x)), \quad \ell = 2, \dots, L, k = 1, \dots, K_\ell$$

$$z_{1,k}(x) := \text{logistic}(\beta_{1,k}^T x), \quad k = 1, \dots, K_1, \quad x \in \mathbb{R}^M$$

# Different Targets $y$

Binary classification:

$$\hat{y}(x) := \hat{p}(y = 1 | x) = \text{logistic}(\beta_{L+1}^T z_L(x))$$

Regression:

$$\hat{y}(x) := \beta_{L+1}^T z_L(x)$$

Regression with multiple outputs:

$$\hat{y}(x) := \beta_{L+1} z_L(x), \quad \beta \in \mathbb{R}^{T \times K_L} \text{ a matrix!}$$

Multi-class classification:

$$\hat{y}(x) := \hat{p}(y | x) = \text{softmax}(\beta_{L+1} z_L(x))$$

Notes:

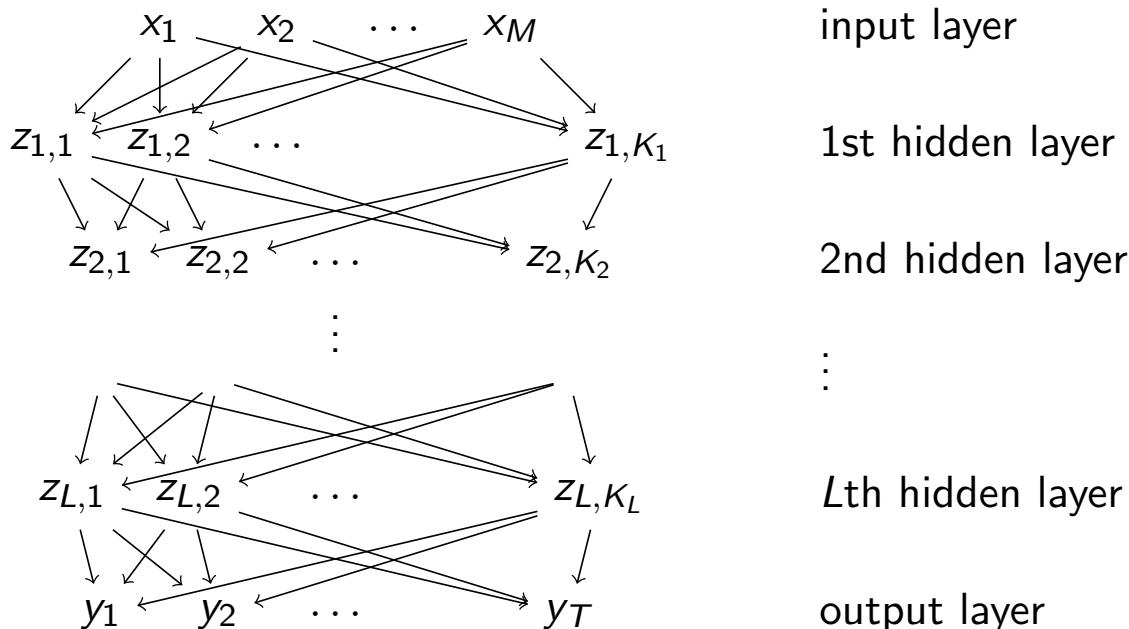
- ▶  $L$  hidden layers
- ▶ at hidden nodes always are logistic/sigmoid functions (**activation function, transfer function**).

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

5 / 24

Machine Learning

# Feedforward Neural Network ( $L$ hidden layers, $T$ outputs)



$$\hat{y}_k(x) := s(\beta_{L+1,k}^T z_L(x)), \quad k = 1, \dots, T$$

$$z_{\ell,k}(x) := s(\beta_{\ell,k}^T z_{\ell-1}(x)), \quad \ell = 2, \dots, L, \quad k = 1, \dots, K_\ell$$

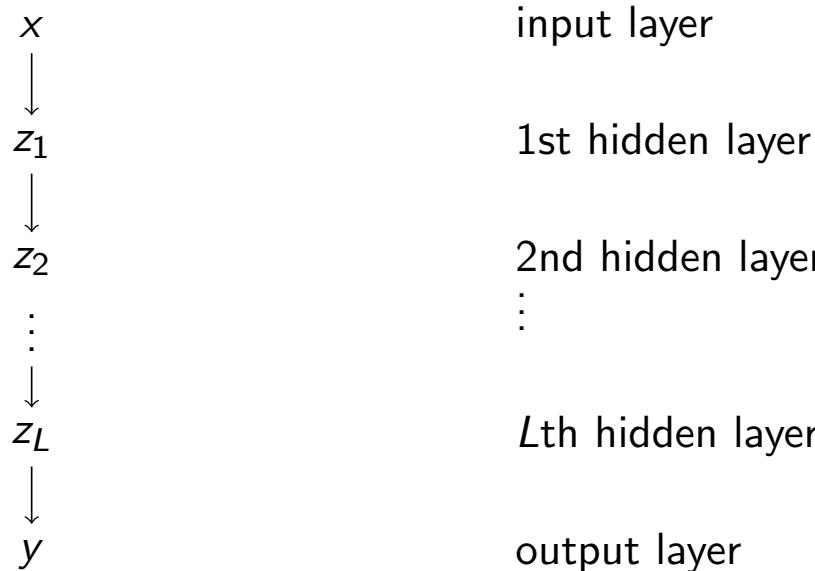
$$z_{1,k}(x) := s(\beta_{1,k}^T x), \quad k = 1, \dots, K_1, \quad x \in \mathbb{R}^M$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

6 / 24

# Feedforward Neural Network ( $L$ hidden layers, $T$ outputs)

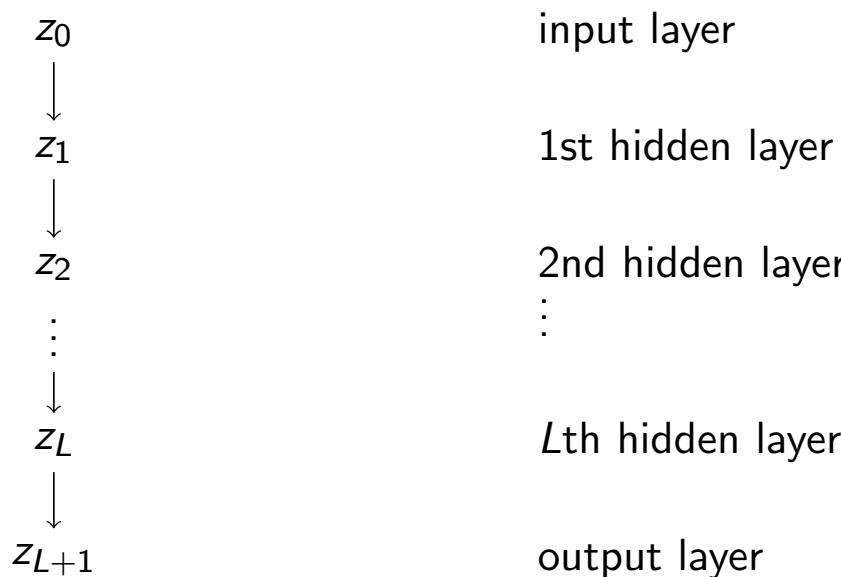
make it simple 1: each layer a vector



$$\begin{aligned}\hat{y}(x) &:= s^\circ(\beta_{L+1} z_L(x)) & \beta_\ell &\in \mathbb{R}^{T \times K_L} \\ z_\ell(x) &:= s^\circ(\beta_\ell z_{\ell-1}(x)), \quad \ell = 2, \dots, L, & \beta_\ell &\in \mathbb{R}^{K_\ell \times K_{\ell-1}} \\ z_1(x) &:= s^\circ(\beta_1 x) & \beta_\ell &\in \mathbb{R}^{K_1 \times M}\end{aligned}$$

# Feedforward Neural Network ( $L$ hidden layers, $T$ outputs)

make it simple 2: rename  $x$  and  $\hat{y}$  to  $z_0$  and  $z_{L+1}$



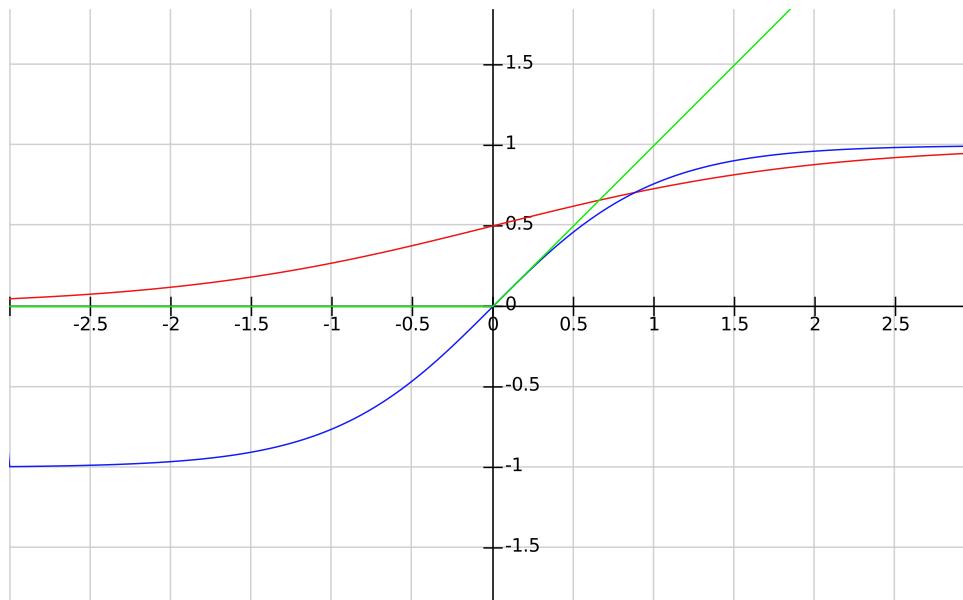
$$z_\ell(x) := s^\circ(\beta_\ell z_{\ell-1}(x)), \quad \ell = 1, \dots, L + 1 \quad \beta_\ell \in \mathbb{R}^{K_\ell \times K_{\ell-1}}$$

$$\text{with } z_0 := x, \quad \hat{y}(x) := z_{L+1}(x), \quad K_1 := M, \quad K_{L+1} := T$$

# Activation Functions

Nowadays, usually the **rectifier** is used as activation function  $s$  (such nodes are called **ReLU: rectified linear unit**):

$$\text{rect}(x) := \max(0, x)$$



red: logistic, blue: tanh, green: rect

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

7 / 24

Machine Learning

# Network Topologies

- ▶ **feedforward neural network** (aka **multilayer perceptron**, MLP)
  - ▶ often just a single hidden layer is used
    - ▶ NN with single hidden layer is already a **universal approximator**
  - ▶ **skip arcs** can be used to connect layers skipping a hidden layer
  - ▶ sometimes layers are not connected completely, but have **sparse connections**.
  - ▶ nodes & connections always form a DAG
- ▶ **recurrent neural network**
  - ▶ neural networks with backward connections / not a DAG.
  - ▶ used in language modeling
  - ▶ no simple probabilistic interpretation
- ▶ **Hopfield networks / associative memory:**
  - ▶ symmetric connections between hidden units
  - ▶ probabilistic counterpart: Boltzmann machine.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

8 / 24

# Outline

1. Network Topologies

2. Stochastic Gradient Descent (Backpropagation)

3. Regularization

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

Machine Learning

9 / 24

## Vector Calculus Refresh – Gradients & Jacobians

function with  $N$  inputs, single output:

$$f : \mathbb{R}^N \rightarrow \mathbb{R}$$

$$x \mapsto f(x_1, \dots, x_N)$$

**gradient (vector):**

$$\nabla f(x) := \left( \frac{\partial f}{\partial x_n}(x) \right)_{n=1:N}$$

function/map with  $N$  inputs,  $M$  outputs:

$$f : \mathbb{R}^N \rightarrow \mathbb{R}^M$$

$$x \mapsto (f_m(x_1, \dots, x_N))_{m=1:M}$$

**Jacobian** (matrix):

$$Df(x) := \left( \frac{\partial f_m}{\partial x_n}(x) \right)_{m=1:M, n=1:N}$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

9 / 24

# Vector Calculus Refresh – Chain Rule

function composition:

$$\begin{array}{ccccccc}
 X := \mathbb{R}^N & \xrightarrow{f} & Y := \mathbb{R}^M & \xrightarrow{g} & Z := \mathbb{R} \\
 x & \mapsto & f(x) & & & & \\
 & & y & \mapsto & g(y) & & \\
 & & x & \mapsto & g \circ f(x) & := g(f(x)) & 
 \end{array}$$

**chain rule:**

$$\nabla(g \circ f)(x) = Df(x)^T(\nabla g)(f(x))$$

# Vector Calculus Refresh – Elementwise Function Application

function with single input, single output:

$$f : \mathbb{R} \rightarrow \mathbb{R}$$

$$x \mapsto f(x)$$

**elementwise function application:**

$$\begin{aligned}
 f^\circ : \mathbb{R}^N &\rightarrow \mathbb{R}^N \\
 x \mapsto (f(x_n))_{n=1:N} &= \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{pmatrix}
 \end{aligned}$$

its Jacobian:

$$Df^\circ(x) = \begin{pmatrix} f'(x_1) & & & \\ & f'(x_2) & & \\ & & \ddots & \\ & & & f'(x_N) \end{pmatrix} = \text{diag}(f'^\circ(x))$$

# Vector Calculus Refresh – Partial Gradients & Jacobians

function with  $N$  inputs, single output:

$$f : \mathbb{R}^N \rightarrow \mathbb{R}$$

$$x \mapsto f(x_1, \dots, x_N)$$

**partial gradient (vector):**

$$\nabla_I f(x) := \left( \frac{\partial f}{\partial x_n}(x) \right)_{n \in I}, \quad I \subseteq \{1, \dots, N\}$$

function/map with  $N$  inputs,  $M$  outputs:

$$f : \mathbb{R}^N \rightarrow \mathbb{R}^M$$

$$x \mapsto (f_m(x_1, \dots, x_N))_{m=1:M}$$

**partial Jacobian (matrix):**

$$D_I f(x) := \left( \frac{\partial f_m}{\partial x_n}(x) \right)_{m=1:M, n \in I} \quad I \subseteq \{1, \dots, N\}$$

## Objective Function

feedforward neural network,  $L$  hidden layers with  $K_1, \dots, K_L$  nodes each:

$$z_\ell(x) := s^\circ(\beta_\ell z_{\ell-1}(x)), \quad \ell = 1, \dots, L+1, \quad \beta_\ell \in \mathbb{R}^{K_\ell \times K_{\ell-1}}$$

with  $z_0 := x$ ,  $\hat{y}(x) := z_{L+1}(x)$ ,  $K_1 := M$ ,  $K_{L+1} := T$

objective function:

$$f(\beta) := \frac{1}{N} \sum_{n=1}^N \mathcal{L}(y_n, \hat{y}(x_n)) + \frac{\lambda}{2} \|\beta\|^2 = \frac{1}{N} \sum_{n=1}^N \mathcal{L}(\beta; x_n, y_n) + \frac{\lambda}{2} \|\beta\|^2$$

loss for single sample:

$$\mathcal{L}(\beta; x, y) := \mathcal{L}(y, z_{L+1}(x))$$

# Objective Function

feedforward neural network,  $L$  hidden layers with  $K_1, \dots, K_L$  nodes each:

$$u_\ell(x) := \beta_\ell z_{\ell-1}(x), \quad \ell = 1, \dots, L+1, \quad \beta_\ell \in \mathbb{R}^{K_\ell \times K_{\ell-1}}$$

$$z_\ell(x) := s^\circ(u_\ell(x))$$

$$\text{with } z_0 := x, \quad \hat{y}(x) := z_{L+1}(x), \quad K_1 := M, \quad K_{L+1} := T$$

loss for single sample:

$$\mathcal{L}(\beta; x, y) := \mathcal{L}(y, z_{L+1}(x))$$

# Objective Function

feedforward neural network,  $L$  hidden layers with  $K_1, \dots, K_L$  nodes each:

$$u_\ell(z_{\ell-1}) := \beta_\ell z_{\ell-1}, \quad \ell = 1, \dots, L+1, \quad \beta_\ell \in \mathbb{R}^{K_\ell \times K_{\ell-1}}$$

$$z_\ell(u_\ell) := s^\circ(u_\ell)$$

$$\text{with } z_0 := x, \quad \hat{y}(x) := (z_{L+1} \circ u_{L+1} \circ z_L \circ u_L \circ \dots \circ z_1 \circ u_1)(x), \quad K_1 := M, \quad K_{L+1} := T$$

loss for single sample:

$$\mathcal{L}(\beta; x, y) := \mathcal{L}(y, z_{L+1}(x)) = (\mathcal{L}_y \circ z_{L+1} \circ u_{L+1} \circ \dots \circ z_\ell \circ u_\ell \circ \dots \circ z_1 \circ u_1)(x)$$

$$\text{with pair loss } \mathcal{L}_y(z_{L+1}) := \text{loss}(y, z_{L+1})$$

# Objective Function

feedforward neural network,  $L$  hidden layers with  $K_1, \dots, K_L$  nodes each:

$$u_\ell(z_{\ell-1}) := \beta_\ell z_{\ell-1}, \quad \ell = 1, \dots, L+1, \quad \beta_\ell \in \mathbb{R}^{K_\ell \times K_{\ell-1}}$$

$$z_\ell(u_\ell) := s^\circ(u_\ell)$$

$$\text{with } z_0 := x, \quad \hat{y}(x) := (z_{L+1} \circ u_{L+1} \circ z_L \circ u_L \circ \dots \circ z_1 \circ u_1)(x), \quad K_1 := M, \quad K_{L+1} := T$$

loss for single sample:

$$\mathcal{L}(\beta; x, y) := \mathcal{L}(y, z_{L+1}(x)) = (\mathcal{L}_y \circ z_{L+1} \circ u_{L+1} \circ \dots \circ z_\ell \circ u_\ell \circ \dots \circ z_1 \circ u_1)(x)$$

$$\text{with pair loss } \mathcal{L}_y(z_{L+1}) := \text{loss}(y, z_{L+1})$$

its gradients:

$$\nabla_{\beta_{\ell,k}} \mathcal{L}(\beta) = D_{\beta_{\ell,k}} u_\ell(z_{\ell-1})^T \nabla(\mathcal{L}_y \circ z_{L+1} \circ u_{L+1} \circ \dots \circ z_{\ell+1} \circ u_{\ell+1} \circ z_\ell)(u_\ell)$$

$$\nabla(\mathcal{L}_y \circ z_{L+1} \circ u_{L+1} \circ \dots \circ z_{\ell+1} \circ u_{\ell+1} \circ z_\ell)(u_\ell)$$

$$= Dz_\ell^T Du_{\ell+1}^T \nabla(\mathcal{L}_y \circ z_{L+1} \circ u_{L+1} \circ \dots \circ z_{\ell+2} \circ u_{\ell+2} \circ z_{\ell+1})(u_{\ell+1})$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

13 / 24

Machine Learning

## Gradients / Recursion Scheme

single sample loss gradients:

$$\nabla_{\beta_{\ell,k}} \mathcal{L}(\beta) = D_{\beta_{\ell,k}} u_\ell(z_{\ell-1})^T \nabla(\mathcal{L}_y \circ z_{L+1} \circ u_{L+1} \circ \dots \circ z_{\ell+1} \circ u_{\ell+1} \circ z_\ell)(u_\ell)$$

$$\nabla(\mathcal{L}_y \circ z_{L+1} \circ u_{L+1} \circ \dots \circ z_{\ell+1} \circ u_{\ell+1} \circ z_\ell)(u_\ell)$$

$$= Dz_\ell^T Du_{\ell+1}^T \nabla(\mathcal{L}_y \circ z_{L+1} \circ u_{L+1} \circ \dots \circ z_{\ell+2} \circ u_{\ell+2} \circ z_{\ell+1})(u_{\ell+1})$$

establishes a recursive computation scheme:

$$\nabla_{\beta_{\ell,k}} \mathcal{L}(\beta) = D_{\beta_{\ell,k}} u_\ell(z_{\ell-1})^T g_\ell(u_\ell)$$

$$g_\ell(u_\ell) := Dz_\ell^T Du_{\ell+1}^T g_{\ell+1}(u_{\ell+1})$$

$$g_{L+1}(u_{L+1}) := Dz_{L+1}^T \nabla \mathcal{L}_y(z_{L+1})$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

14 / 24

# Gradients / Components

$$u_\ell(z_{\ell-1}) := \beta_\ell z_{\ell-1}, \quad \ell = 1, \dots, L+1, \quad \beta_\ell \in \mathbb{R}^{K_\ell \times K_{\ell-1}}$$

$$z_\ell(u_\ell) := s^\circ(u_\ell)$$

single sample loss gradients:

$$\nabla_{\beta_{\ell,k}} \mathcal{L}(\beta) = D_{\beta_{\ell,k}} u_\ell(z_{\ell-1})^T g_\ell(u_\ell)$$

$$g_\ell(u_\ell) := Dz_\ell^T Du_{\ell+1}^T g_{\ell+1}(u_{\ell+1})$$

$$g_{L+1}(u_{L+1}) := Dz_{L+1}^T \nabla \mathcal{L}_y(z_{L+1})$$

components:

$$Du_\ell = \beta_\ell$$

$$Dz_\ell = \text{diag}(s'^\circ(u_\ell))$$

$$D_{\beta_{\ell,k}} u_\ell = e_k z_{\ell-1}^T \quad \rightsquigarrow \quad \begin{aligned} \nabla_{\beta_{\ell,k}} \mathcal{L}(\beta) &= z_{\ell-1} e_k^T g_\ell(u_\ell) \\ \nabla_{\beta_\ell} \mathcal{L}(\beta) &= g_\ell(u_\ell) z_{\ell-1}^T \end{aligned}$$

Note:  $e_k$  denotes the  $k$ -th unit vector:  $(e_k)_j := \mathbb{I}(k=j)$ .

$\beta_\ell$  is a parameter matrix, thus  $\nabla_{\beta_\ell} \mathcal{L}(\beta)$  is a matrix-shaped gradient!

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

15 / 24

Machine Learning

# Gradients / Sticking Everything Together

parameters:

$$\beta_\ell \in \mathbb{R}^{K_\ell \times K_{\ell-1}}, \quad \ell = 1 : L+1$$

feed forward:

$$z_0 := x$$

$$u_\ell := \beta_\ell z_{\ell-1}, \quad \ell = 1 : L+1$$

$$z_\ell := s^\circ(u_\ell)$$

back propagation:

$$g_{L+1}(u_{L+1}) := \text{diag}(s'^\circ(u_{L+1})) \nabla \mathcal{L}_y(z_{L+1})$$

$$\nabla_{\beta_\ell} \mathcal{L}(\beta) = g_\ell(u_\ell) z_{\ell-1}^T, \quad \ell = L+1 : 1 \text{ backwards}$$

$$\beta_\ell^{\text{next}} := \beta_\ell - \eta (\nabla_{\beta_\ell} \mathcal{L}(\beta) + \lambda \beta_\ell)$$

$$g_\ell(u_\ell) := \text{diag}(s'^\circ(u_\ell)) \beta_{\ell+1}^T g_{\ell+1}(u_{\ell+1})$$

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

16 / 24

# SGD / Backpropagation

```

1 learn-nn-sgd( $\mathcal{D}^{\text{train}} := \{(x_1, y_1), \dots, (x_N, y_N)\}, L, K, s, \nabla \mathcal{L}, \lambda, \eta, I$ ):
2   randomly initialize  $\beta_\ell \in \mathbb{R}^{K_\ell \times K_{\ell-1}}$ ,  $\ell = 1 : L + 1$ 
3   for  $i := 1, \dots, I$ :
4     for  $(x_n, y_n) \in \mathcal{D}^{\text{train}}$  in random order:
5        $z_0 := x_n$                                      [feed forward]
6       for  $\ell := 1 : L + 1$ :
7          $u_\ell := \beta_\ell z_{\ell-1}$ 
8          $z_\ell := s^\circ(u_\ell)$ 
9          $g_{L+1} := \text{diag}(s'^\circ(u_{L+1})) \nabla \mathcal{L}_{y_n}(z_{L+1})$       [back propagation]
10        for  $\ell := L + 1 : 2$  backwards:
11           $g_{\ell-1} := \text{diag}(s'^\circ(u_{\ell-1})) \beta_\ell^T g_\ell$ 
12           $\beta_\ell := \beta_\ell - \eta_i(g_\ell z_{\ell-1}^T + \lambda \beta_\ell)$ 
13           $\beta_1 := \beta_1 - \eta_i(g_1 z_0^T + \lambda \beta_1)$ 
14        if converged(...):
15          return  $\beta$ 
16        raise exception "not converged in  $I$  iterations"

```

where

- ▶  $L$  number of layers
- ▶  $K$  layer sizes
- ▶  $s$  activation function
- ▶  $\nabla \mathcal{L}$  loss gradient
- ▶  $\lambda$  regularization weight
- ▶  $\eta$  step length schedule
- ▶  $I$  number of iterations

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

17 / 24

Machine Learning

## Outline

1. Network Topologies

2. Stochastic Gradient Descent (Backpropagation)

3. Regularization

# Regularization of Neural Networks

generic, working with any model:

- ▶ L2 regularization
  - ▶ aka **weight decay**
  - ▶ most frequently used method
- ▶ L1 regularization
- ▶ **early stopping**

specific for neural networks:

- ▶ structural regularization:
  - ▶ sufficiently small number of layers and sizes of layers
  - ▶ compare number of parameters with sample size!
- ▶ **dropout** [Srivastava et al., 2014]
  - ▶ use random sample of input and hidden nodes for each instance during training
- ▶ **Batch normalization** [Ioffe and Szegedy, 2015]
  - ▶ standardize the values  $z_{\ell,k}$  for each layer (for a minibatch).
- ▶ **self-normalizing neural networks** [Klambauer et al., 2017]

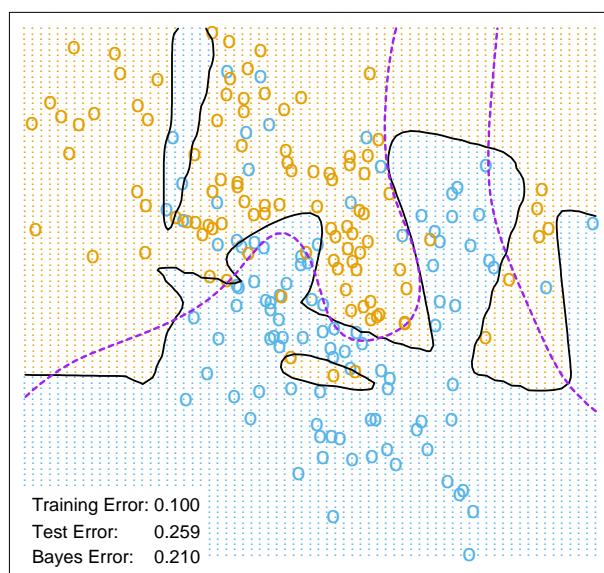
Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

18 / 24

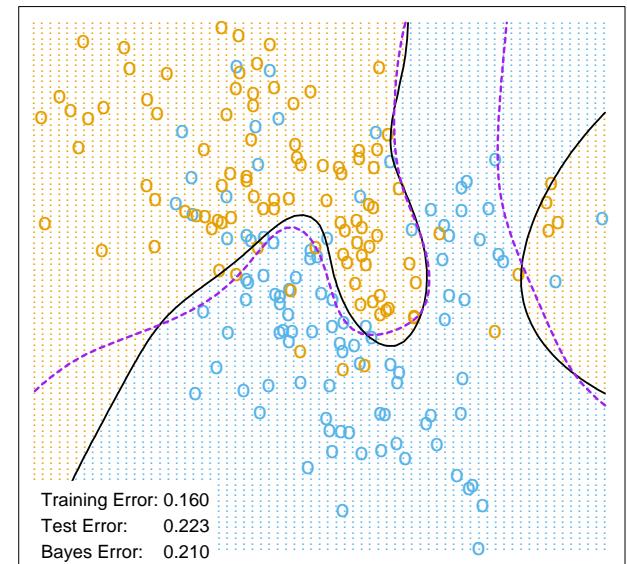
Machine Learning

## L2 regularization / Example

Neural Network - 10 Units, No Weight Decay



Neural Network - 10 Units, Weight Decay=0.02

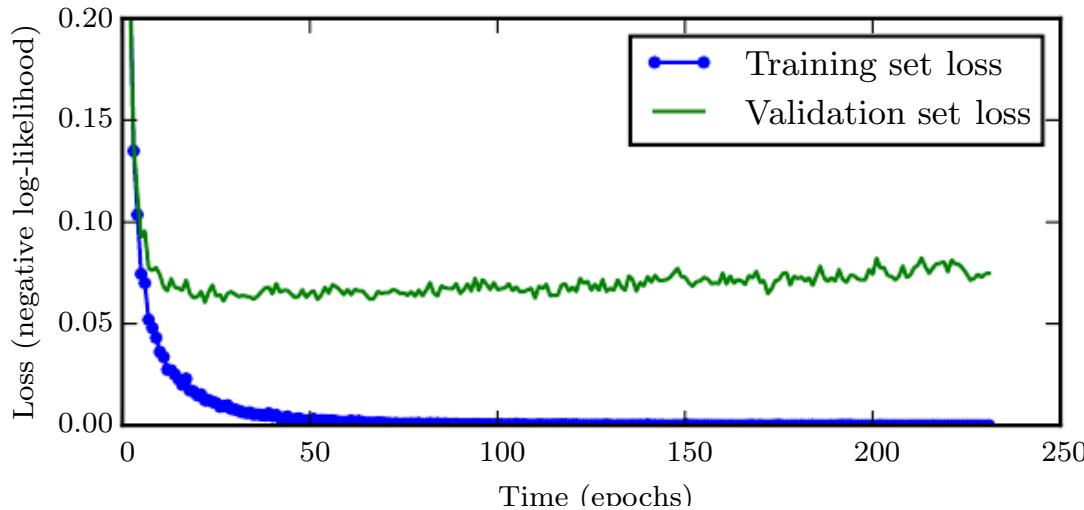


[Hastie et al., 2005, p. 39]

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

19 / 24

# Early Stopping



[source: Goodfellow et al. 2016, p. 239]

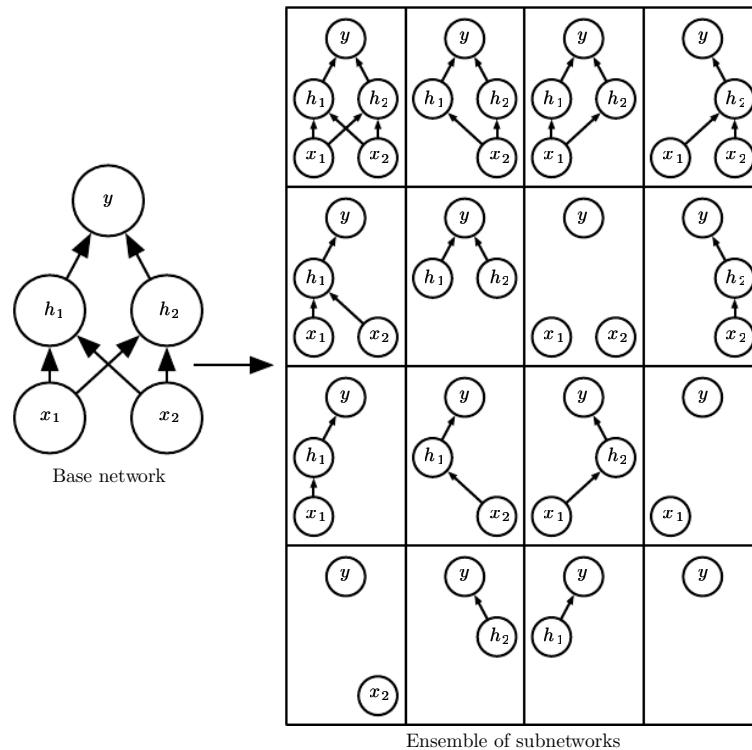
Early stopping works with any iterative learning algorithm.

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

20 / 24

Machine Learning

# Dropout



[source: Goodfellow et al. 2016, p. 252]

Lars Schmidt-Thieme, Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, Germany

21 / 24

## Summary (1/3)

- ▶ (Feedforward) Neural networks are supervised parametric models
  - ▶ arranged in several layers,
    - ▶ with the first layer being the inputs,
    - ▶ the last layer being the outputs,
    - ▶ intermediate/**hidden layers** representing subexpressions of the prediction function  
(not to be confused with latent variables!)
  - ▶ each layer composed of a **linear combination of the previous one**, with weights being parameters of the model,
  - ▶ and a **nonlinear activation function**,
    - ▶ usually the linear rectifier  $\max(0, x)$
    - ▶ or a sigmoid function (logistic, tanh)
- ▶ Neural networks are learnt through Stochastic Gradient Descent
  - ▶ computation of the gradients in reverse order of computations of predictions (**backpropagation**)
  - ▶ usually using **minibatches** for a few ten or hundred instances.

## Summary (2/3)

- ▶ As any other model, neural networks have to be regularized.
  - ▶ **structural regularization:**
    - ▶ number of nodes/layer and number of layers.
  - ▶ early stopping
  - ▶ L2 regularization (**weight decay**)
  - ▶ **dropout:** use a random sample of input and hidden nodes per example
- ▶ Neural networks can be extended in a rather straightforward way to work with sequential/time series, image data and any other kind of array data.
  - ▶ **convolutional neural networks**
  - ▶ these models belong to the most powerful models currently used in ML

# Summary (3/3)

- ▶ A neural network with a single hidden layer can already approximate any function arbitrarily well.
  - ▶ **universal approximator**
  - ▶ if one adds arbitrarily many hidden nodes in that layer as necessary
  - ▶ but deeper networks with more than one hidden layer have shown to generalize better
    - ▶ make better use of a given number of parameters
    - ▶ **deep learning**

## Further Readings

- ▶ See Murphy 2012, chapter 16.5 and Hastie et al. 2005, chapter 11.
- ▶ More detailed introduction in Goodfellow et al. 2016, chapter 6 and 7.

# References

- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The Mit Press, Cambridge, Massachusetts, November 2016. ISBN 978-0-262-03561-3.
- Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, volume 27. Springer, 2005.
- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.
- Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-Normalizing Neural Networks. *arXiv preprint arXiv:1706.02515*, 2017.
- Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.
- Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.