# Optimization and Data Science

## Lecture 18: Optimization in the Training of Artificial Neural Networks

Prof. Dr. Thomas Slawig

Kiel University - CAU Kiel
Dep. of Computer Science

Summer 2020

# Contents

# Optimization in the Training of Artificial Neural Networks

- What is it?

  Artificial Neural Networks (ANNs) try to mimic the human's brain

  ANNs can perform many tasks very efficiently

  ANNs have to be trained to work properly

  This training process is an optimization process
- Why are we studying this?

  Artificial neural networks (ANNs) are a main tool in machine learning, data science
- How does it work?

  The optimization problem is a least-squares or regression problem
- What if we can use it?

  Understand how training of ANNs works

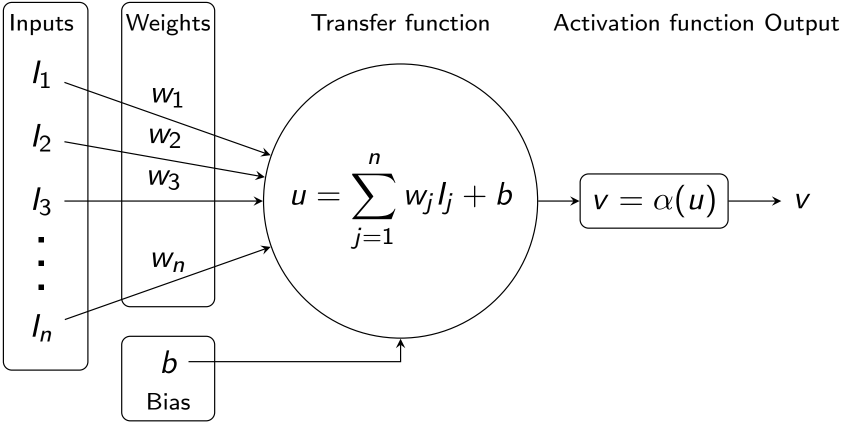
  Set parameters of ANN software properly

  Decide which algorithm might be beneficial

  Find right stopping criteria

# Contents

# Basis of Artificial Neural Networks (ANN): Artificial neuron




- This gives for one neuron with index $i$: $v_i = \alpha_i\left(\sum_{j=1}^{n} w_{ij} I_j + b_i\right)$.

# Artificial Neural Network (ANN) - Structure

- $\ell = 0$: input layer,
- $\ell = \ell_{out}$: output layer,
- $\ell = 1, \ldots, \ell_{out} - 1$: hidden layers.
- $N_\ell$: # neurons in layer $\ell$,
- Output of layer $\ell - 1$ is input of layer $\ell$:

$$v_{\ell i} = \alpha_{\ell i}\left( \sum_{j=1}^{N_{\ell-1}} w_{\ell ij} v_{\ell-1,j} + b_{\ell i} \right), i = 1, \ldots, N_\ell.$$

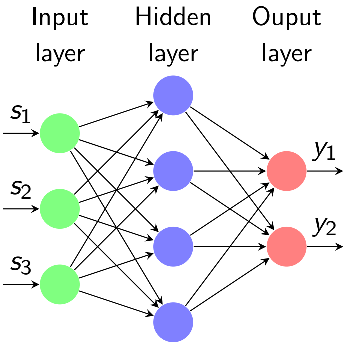
- Input of the ANN:

$$s_j = v_{0j}, j = 1, \ldots, N_0.$$

- Output of the ANN:

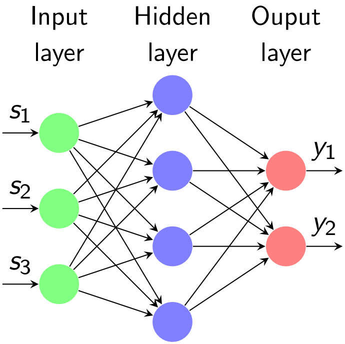
$$y_i = v_{\ell_{out} i}, i = 1, \ldots, N_{\ell_{out}}.$$

## Network structure

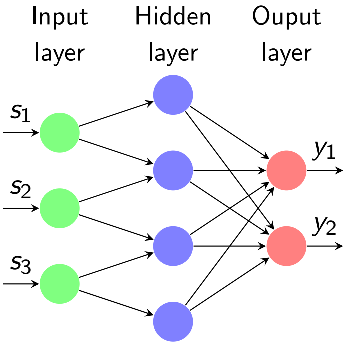


# Artificial Neural Network (ANN) - Structure

## Fully connected network




## Partially connected network




- $\rightsquigarrow$ sparsity of weight tensor $W = (w_{\ell ij})_{\ell ij}$, i.e., some of the weights $w_{\ell ij}$ are fixed as 0.

# Mathematical function representing the action of a single layer

## Layer $\ell$:

- matrix $W_\ell := (w_{\ell ij})_{ij} \in \mathbb{R}^{N_{\ell-1} \times N_\ell}$ of weights
- vector $b_\ell = (b_{\ell i})_i \in \mathbb{R}^{N_\ell}$ of biases
- activation functions $\alpha_\ell = (\alpha_{\ell i})_i$.

For neuron with index $i$:

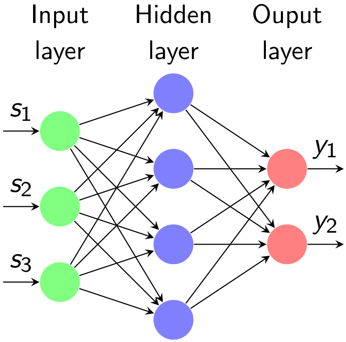
$$v_{\ell i} = \alpha_{\ell i}\left( \sum_{j=1}^{N_{\ell-1}} w_{\ell ij} v_{\ell-1,j} + b_{\ell i} \right).$$

Mapping of the whole layer can then be written as

$$v_\ell = \alpha_\ell(W_\ell v_{\ell-1} + b_\ell) =: F_\ell(v_{\ell-1}) \in \mathbb{R}^{N_\ell}.$$

$F_\ell$ depends on $W_\ell$, $b_\ell$, and (optionally) parameters of activation functions.

## Network structure



# Activation functions

... with parameters $\beta$:

$$\text{Identity (linear):} \quad v \mapsto v$$

$$\text{Heavyside function (nonlinear, not differentiable):} \quad v \mapsto \begin{cases} 0 & \text{if } v < 0 \\ 1, & \text{if } v \geq 0 \end{cases}$$

$$\text{Exponential linear unit (nonlinear):} \quad v \mapsto \begin{cases} \beta\,(e^v - 1)\,, & \text{if } v < 0 \\ v, & \text{if } v \geq 0 \end{cases}$$

$$\text{Hyperbolic tangens (nonlinear):} \quad v \mapsto \frac{1}{2}(1 + \tanh(\beta v))$$

$$\text{Logistic function (nonlinear):} \quad v \mapsto \frac{1}{1 + e^{-\beta v}}$$

$$\vdots$$

# Artificial neural network - Structure

**Given:**

- Number of inputs
- Number of outputs

**To be chosen:**

- Number of hidden layers
- Number of nodes per layer
- Fully/partially connected
- Constraints on weights (independent or common between some neurons) $\rightsquigarrow$ convolutional neural networks, ...
- Activation functions.

# Contents

# Mathematical function representing the whole ANN

- Mapping of one layer:

$$v_\ell = \alpha_\ell(W_\ell v_{\ell-1} + b_\ell) =: F_\ell(v_{\ell-1}), \quad \ell = 1, \ldots, \ell_{out}.$$

- Complete network:

$$s := v_0 \in \mathbb{R}^{N_0} : \text{ input},$$

$$y := v_{\ell_{out}} = F_{\ell_{out}} \circ \cdots \circ F_1(s) =: F(s,x) \in \mathbb{R}^{N_{\ell_{out}}} : \text{ output}.$$

  where in $x \in \mathbb{R}^n$ we summarize all parameters, i.e.,

$$x := \left\{ (W_\ell)_{\ell=1}^{\ell_{out}}, (b_\ell)_{\ell=1}^{\ell_{out}}, \text{ optionally parameters of activation functions} \right\}.$$

- $\rightsquigarrow$ Mathematical function representing an ANN for fixed parameters $x$:

$$F(\cdot, x) : \mathbb{R}^{N_0} \to \mathbb{R}^{N_{\ell_{out}}}, \quad s \mapsto y = F(s,x)$$

- seen as function of parameters $x$ and fixed input:

$$F(s, \cdot) : \mathbb{R}^n \to \mathbb{R}^{N_{\ell_{out}}}, \quad x \mapsto y = F(s,x)$$

# Training of an ANN

- Mathematical function representing an ANN for fixed parameters $x$:

$$F(\cdot, x) : \mathbb{R}^{N_0} \to \mathbb{R}^{N_{\ell_{out}}}, \quad s \mapsto y = F(s, x)$$

- seen as function of parameters $x$ and fixed input:

$$F(s, \cdot) : \mathbb{R}^{n} \to \mathbb{R}^{N_{\ell_{out}}}, \quad x \mapsto y = F(s, x)$$

- We define a **loss function**

$$L : \mathbb{R}^{n_{out}} \times \mathbb{R}^{n_{out}} \to \mathbb{R}$$

  that measures the misfit between model output $y = F(s, x)$ and data $z$, e.g.:

$$L(y, z) := \|y - z\|_2^2 \ \rightsquigarrow \ L(F(s, x), z) := \|F(s, x) - z\|_2^2$$

- Training: Given set $\{(s_j, z_j) : j = 1, \ldots, m\}$ of training data, we define the **empirical risk**

$$R_{emp}(x) := \frac{1}{m} \sum_{j=1}^{m} L(F(s_j, x), z_j).$$

# Training of an ANN

- Training procedure $\rightsquigarrow$ Optimization problem:

$$\min_{x \in \mathbb{R}^n} R_{emp}(x), \quad R_{emp}(x) := \frac{1}{m} \sum_{j=1}^{m} L(F(s_j, x), z_j)$$

- $\rightsquigarrow f = R_{emp}$ is our cost function.
- But: we want the ANN to be "good" for "arbitrary/random" data *after the training*.
- Here, we need a different quality measure, the **expected risk**:

$$R_{exp}(x) := \mathbb{E}\left(L(F(S, x), Z)\right).$$

- Now, $(S, Z)$ are considered as random variables.
- $\mathbb{E}$ is the expectation of the random variable, defined as

$$\mathbb{E}(X) := \int_{\mathbb{R}} x\, f_X(x) dx,$$

where $f_X$ is the probability density function of the random variable $X$.

# Gradient of the empirical risk

- Using the special structure of the empirical risk (= the cost function in the optimization to be performed in the training)

$$R_{emp}(x) = \frac{1}{m}\sum_{j=1}^{m} L(F(s_j, x), z_j),$$

- we get for the gradient (w.r.t. $x$!!!):

$$\nabla R_{emp}(x) = \frac{1}{m}\sum_{j=1}^{m} \nabla L(F(s_j, x), z_j),$$

- The gradient is the sum of the gradients of the loss terms.

# Contents

# Full/batch gradient vs. minibatch vs. stochastic gradient

- The gradient of the expected risk is the sum of the gradients of the loss terms:

$$\nabla R_{emp}(x) = \frac{1}{m}\sum_{j=1}^{m} \nabla L(F(s_j, x), z_j).$$

- We have now different options for the descent direction in a gradient method:

$$\textbf{Full/batch gradient: } d = -\frac{1}{m}\sum_{j=1}^{m} \nabla L(F(s_j, x), y_j),$$

$$\textbf{Stochastic gradient (SG): } d = -\nabla L(F(s_j, x), z_j), \quad j \text{ chosen randomly},$$

$$\textbf{Minibatch SG: } d = -\frac{1}{|J|}\sum_{j\in J} \nabla L(F(s_j, x), z_j),$$

$$J \subset \{1, \dots, m\} \text{ chosen randomly}.$$

- In the first and last case the single terms can be computed in parallel.

# Stochastic Gradient (SG) Method with fixed stepsize

**Algorithm (Stochastic Gradient Method, fixed stepsize)**:

- Input: Training data set $\{(s_j, z_j) : j = 1, \ldots, m\}$.

1. Choose initial guess $x_0 \in \mathbb{R}^n$.
2. Choose a fixed stepsize $\rho > 0$.
3. For $k = 0, 1, \ldots$ :
    1. Compute a random number $j \in \{1, \ldots, m\}$.
    2. Compute the negative gradient of the $j$-th loss term and take this as descent direction:
    $$d_k = -\nabla L(F(s_j, x), z_j).$$
    3. Set $x_{k+1} = x_k + \rho d_k$.

   until a stopping criterion is satisfied.

# Stochastic Gradient (SG) Method, decreasing stepsize

**Algorithm (Stochastic Gradient Method, decreasing stepsize)**:

- Input: Training data set $\{(s_j, z_j) : j = 1, \ldots, m\}$.

1. Choose initial guess $x_0 \in \mathbb{R}^n$.
2. Choose an initial stepsize $\rho_0 > 0$.
3. For $k = 0, 1, \ldots$ :
    1. Compute a random number $j \in \{1, \ldots, m\}$.
    2. Compute the negative gradient of the $j$-th loss term and take this as descent direction:
    $$d_k = -\nabla L(F(s_j, x), z_j).$$
    3. Set $x_{k+1} = x_k + \rho_k d_k$.
    4. Reduce the stepsize by "some formula" ($\rightarrow$ later), i.e., choose
    $$\rho_{k+1} < \rho_k.$$

until a stopping criterion is satisfied.

# Contents

# Gradient of the empiricial risk

- For the gradient of the empiricial risk,

$$\nabla R_{emp}(x) = \frac{1}{m} \sum_{j=1}^{m} \nabla L(F(s_j, x), z_j),$$

- we need the gradient (w.r.t. $x$) of the loss function with $y = F(s, x)$, i.e. the mapping:

$$x \mapsto L(F(s, x), z),$$

  where in $x \in \mathbb{R}^n$ we summarized all parameters, i.e.,

$$x := \left\{ (W_\ell)_{\ell=1}^{\ell_{out}}, (b_\ell)_{\ell=1}^{\ell_{out}}, \text{ optionally parameters of activation functions} \right\}.$$

- Example: Compute derivative w.r.t. one weight $w_{\ell ij}$ by the chain rule:

$$\frac{\partial}{\partial w_{\ell ij}} L(\underbrace{F(s, x)}_{=y}, z) = \sum_{k=1}^{N_{\ell_{out}}} \frac{\partial}{\partial y_k} L(y, z) \frac{\partial y_k}{\partial w_{\ell ij}} = \nabla_y L(y, z)^\top \frac{\partial y}{\partial w_{\ell ij}} \in \mathbb{R}.$$

# Gradient of the empiricial risk

- Derivative w.r.t. one weight $w_{\ell ij}$ by the chain rule:

$$\frac{\partial}{\partial w_{\ell ij}} L(F(s,x),z) = \sum_{k=1}^{N_{\ell_{out}}} \frac{\partial}{\partial y_k} L(y,z) \frac{\partial y_k}{\partial w_{\ell ij}} = \nabla_y L(y,z)^\top \frac{\partial y}{\partial w_{\ell ij}} \in \mathbb{R}.$$

- Outer derivative is easy (if loss function uses squared Euclidean norm):

$$\nabla_y L(y,z) = \nabla_y \left(\|y-z\|_2^2\right) = \nabla_y \left( \sum_{i=1}^{N_{\ell_{out}}} (y_i - z_i)^2 \right) = 2(y-z) \in \mathbb{R}^{N_{\ell_{out}}}.$$

- Inner derivative requires chain rule again since

$$y = (F_{\ell_{out}} \circ \dots \circ F_1)(s),$$

which gives

$$\mathbb{R}^{N_{\ell_{out}}} \ni \frac{\partial y}{\partial w_{\ell ij}} = \left( \frac{\partial y_k}{\partial w_{\ell ij}} \right)_{k=1}^{N_{\ell_{out}}} = F'_{\ell_{out}}(v_{\ell_{out}-1}) \cdots F'_{\ell+1}(v_\ell) \frac{\partial F_\ell}{\partial w_{\ell ij}}(v_{\ell-1})$$

because $F_{\ell-1}, \dots, F_1$ do not depend on $w_{\ell ij}$.

# Gradient of the empiricial risk

- For the derivative

$$\frac{\partial y}{\partial w_{\ell ij}} = F'_{\ell_{out}}(v_{\ell_{out}-1}) \cdots F'_{\ell+1}(v_\ell) \frac{\partial F_\ell}{\partial w_{\ell ij}}(v_{\ell-1}) \in \mathbb{R}^{N_{\ell_{out}}}$$

we need

$$\frac{\partial F_{\ell i}}{\partial v_k}(v) = \frac{\partial}{\partial v_k}\left(\alpha_{\ell i}\left(\sum_{j=1}^{N_{\ell-1}} w_{\ell ij} v_j + b_{\ell i}\right)\right) = \alpha'_{\ell i}(\cdots) \underbrace{\frac{\partial}{\partial v_k}\left(\sum_{j=1}^{N_{\ell-1}} w_{\ell ij} v_j + b_{\ell j}\right)}_{=w_{\ell ik}},$$

which gives

$$F'_\ell(v) := \mathrm{diag}\left(\alpha'_\ell(W_\ell v + b_\ell)\right) W_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}, \quad \ell = 1, \ldots, \ell_{out}.$$

# Gradient of the empiricial risk

- For the last term we get:

$$\frac{\partial F_\ell}{\partial w_{\ell ij}}(v_{\ell-1}) = \begin{pmatrix} 0 \\ \vdots \\ \alpha'_{\ell i}(w_{\ell ij} v_{\ell-1,j} + b_{\ell i}) v_{\ell-1,j} \\ \vdots \\ 0 \end{pmatrix} \quad \text{since } F_{\ell i}(v) = \alpha_{\ell i}\Big(\sum_{r=1}^{n} w_{\ell ir} v_{\ell-1,r} + b_{\ell i}\Big).$$

- With

$$F'_\ell(v) := \operatorname{diag}\left(\alpha'_\ell(W_\ell v + b_\ell)\right) W_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}, \quad \ell = 1, \dots, \ell_{out},$$

we get

$$\frac{\partial y_k}{\partial w_{\ell ij}} = F'_{\ell_{out}}(v_{\ell_{out}-1}) \cdots F'_{\ell+1}(v_\ell) \frac{\partial F_\ell}{\partial w_{\ell ij}}(v_{\ell-1}) = \left(\prod_{r=\ell}^{\ell_{out}} \operatorname{diag}\left(\alpha'_\ell(W_\ell v_{\ell-1} + b_\ell)\right) W_\ell\right)_{ki} v_{\ell-1,j}.$$

# Contents

# Random events in the ANN context

There are two points where random events play a role in the ANN context:

- Expected risk, the quantity we are interested in as quality measure of the trained ANN:

$$R_{exp}(x) := \mathbb{E}\left(L(F(S,x),Z)\right).$$

  - For "unknown" data, $(S,Z)$ are treated as random variables.
  - $\rightsquigarrow$ $F(S,x)$ is a random variable
  - $\rightsquigarrow$ $L(F(S,x),Z)$ is a random variable.
  - We want to minimize its expected value.
- Stochastic Descent Method:
  - Search direction is a random variable (here still written in lower case), if index subset $J \subset \{1,\ldots,m\}$ is randomly chosen:

$$d_k = -M_k\Bigg(\frac{1}{|J|}\sum_{j\in J}\nabla L(F(s_j,x),z_j)\Bigg).$$

  - $\rightsquigarrow$ iterates $x_k$ are random variables.
- $\rightsquigarrow$ Convergence results can be only formulated using expected values $\mathbb{E}(\ldots)$.

# Convergence Results

- Convergence results can be only formulated using expected values $\mathbb{E}(\ldots)$.
- They can be applied either to the empirical risk $f = R_{emp}$ ...
- ... or the expected risk $f = R_{exp}$.
- They can be formulated generally for the search direction $d_k$ being
  - stochastic/minibatch/full gradient

$$d_k = -\frac{1}{|J|}\sum_{j\in J} \nabla L(F(s_j, x), z_j).$$

  - Newton or Quasi-Newton method:

$$d_k = -M_k\Big(\frac{1}{|J|}\sum_{j\in J} \nabla L(F(s_j, x), z_j)\Big)$$

with $M_k$ positive-definite.

# Convergence result: Assumptions for $f$

### Theorem (Part 1)

*Let*

- $f$ *be bounded from below on an open set* $D \subset \mathbb{R}^n$ *with* $(x_k)_k \subset D$,
- $\nabla f$ *be Lipschitz-continuous with constant* $L > 0$, *i.e.*

$$\|\nabla f(x) - \nabla f(\tilde{x})\| \leq L\|x - \tilde{x}\| \text{ for all } x, \tilde{x} \in \mathbb{R}^n.$$

*(Satisfied if* $f$ *is twice continuously differentiable).*

# Convergence result: Assumptions for search direction

## Theorem (Part 2)

*Let the sequence of search directions satisfy:*

- *Directions are gradient-related (in expectation):*

$$\exists C_D > 0 : \quad -\frac{\nabla f(x_k)^\top \mathbb{E}(d_k)}{\|\nabla f(x_k)\|_2^2} \geq C_D > 0 \ \textit{ for all } k \in \mathbb{N}.$$

- *Expected value and variance are bounded by the gradient:*

$$\begin{aligned} \exists C_G \geq C_D : \quad & \|\mathbb{E}(d_k)\|_2 \leq C_G \|\nabla f(x_k)\|_2 \ \textit{ for all } k \in \mathbb{N}, \\ \exists M, M_V \geq 0 : \quad & \mathbb{V}(d_k) \leq M + M_V \|\nabla f(x_k)\|_2^2 \ \textit{ for all } k \in \mathbb{N}, \\ \textit{with the } \textbf{variance} \quad & \mathbb{V}(d_k) := \mathbb{E}\left((d_k - \mathbb{E}(d_k))^2\right) \end{aligned}$$

*which is a measure for the "noise" in $d_k$.*

# Convergence result

## Theorem (Part 3)

*Under the above assumptions we get:*

- *For a fixed stepsize that is "small enough":*

$$0 < \rho \le \frac{C_D}{LC_G} \implies \mathbb{E}\underbrace{\left(\frac{1}{k}\sum_{j=1}^{k}\|\nabla f(x_j)\|_2^2\right)}_{\text{mean gradient after } k \text{ steps}} \longrightarrow \frac{\rho LM}{C_D} \quad (k \to \infty).$$

  *where* $\mathbb{V}(d_k) \le M + M_V\|\nabla f(x_k)\|_2^2$.

- *For a decreasing stepsize: Similar results for the expected value of the gradient norm*

$$\mathbb{E}\left(\|\nabla f(x_k)\|_2\right)$$

  *as in the "normal" descent methods.*

# What is important

- The training of an artificial neural network (ANN) is an optimization problem.
- Parameters are the weights and the bias in the different neurons.
- The number of parameters depends on the number of layers and can be quite high ($\rightsquigarrow$ "deep learning").
- The cost function is a least-squares type function.
- We distinguish between the best fit for the given training data and the best fit for the (unknown) real application data.
- Efficient computation of the gradient is crucial.
- Stochastic methods are used, where in every iteration only a part of the data points are used for the gradient computation.
- The standard method is the stochastic gradient method.