

Machine Learning I

Lucas Fabian Naumann, David Stein, Bjoern Andres

Machine Learning for Computer Vision
TU Dresden



<https://mlcv.cs.tu-dresden.de/courses/25-winter/ml1/>

Winter Term 2025/2026

Learning of composite functions (deep learning)

Contents. This part of the course is about a special case of supervised learning: the supervised learning of **composite functions**, aka. **supervised deep learning**.

- ▶ We define a family of (composite) functions in terms a **compute graph**.
- ▶ We describe two algorithms for computing partial derivatives (if these exist) of such functions, **forward propagation** and **backward propagation**.
- ▶ In the exercises, we compare these algorithms.

Notation. Let $G = (V, E)$ a digraph.

- For any $v \in V$, let

$$P_v = \{u \in V \mid (u, v) \in E\} \quad \text{the set of **parents** of } v \quad (1)$$

$$C_v = \{w \in V \mid (v, w) \in E\} \quad \text{the set of **children** of } v. \quad (2)$$

- For any $u, v \in V$, let $\mathcal{P}(u, v)$ denote the set of all uv -paths of G . (Any path is a subgraph. For any node u , the uu -path $(\{u\}, \emptyset)$ exists.)

Let G be **acyclic**.

- For any $v \in V$, let

$$A_v = \{u \in V \mid \mathcal{P}(u, v) \neq \emptyset\} \setminus \{v\} \quad \text{the set of **ancestors** of } v \quad (3)$$

$$D_v = \{w \in V \mid \mathcal{P}(v, w) \neq \emptyset\} \setminus \{v\} \quad \text{the set of **descendants** of } v. \quad (4)$$

Definition. A tuple $(V, D, D', E, \Theta, \{g_{v\theta}: \mathbb{R}^{P_v} \rightarrow \mathbb{R}\}_{v \in (D \cup D') \setminus V, \theta \in \Theta})$ is called a **compute graph**, iff the following conditions hold:

- ▶ $G = (V \cup D \cup D', E)$ is an acyclic digraph.
- ▶ For any $v \in V$, called an **input node**, $P_v = \emptyset$.
- ▶ For any $v \in D'$, called an **output node**, $C_v = \emptyset$.
- ▶ For any $v \in D$, called a **hidden node**, $P_v \neq \emptyset$ and $C_v \neq \emptyset$.

Definition. For any compute graph

$(V, D, D', E, \Theta, \{g_{v\theta}: \mathbb{R}^{P_v} \rightarrow \mathbb{R}\}_{v \in (D \cup D') \setminus V, \theta \in \Theta})$, any $v \in V \cup D \cup D'$ and any $\theta \in \Theta$, let $\alpha_{v\theta}: \mathbb{R}^V \rightarrow \mathbb{R}$ such that for all $\hat{x} \in \mathbb{R}^V$:

$$\alpha_{v\theta}(\hat{x}) = \begin{cases} \hat{x}_v & \text{if } v \in V \\ g_{v\theta}(\alpha_{P_v\theta}(\hat{x})) & \text{otherwise} \end{cases} . \quad (5)$$

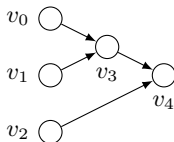
For any $\theta \in \Theta$ let $f_\theta: \mathbb{R}^V \rightarrow \mathbb{R}^{D'}$ such that $f_\theta = \alpha_{D'\theta}$.

We call $\alpha_{v\theta}(\hat{x})$ the **activation** of v for **input** $\hat{x}$ and **parameters** θ .

We call $f_\theta(\hat{x})$ the **output** of the compute graph for input $\hat{x}$ and parameters θ .

Learning of composite functions (deep learning)

Example. Consider $V = \{v_0, v_1, v_2\}$, $D = \{v_3\}$, $D' = \{v_4\}$ and the edge set E of the digraph depicted below.



Consider, in addition, $\Theta = \{\theta_0, \theta_1\}$ and

$$g_{v_3\theta}: \mathbb{R}^{\{v_0, v_1\}} \rightarrow \mathbb{R}: \quad x \mapsto x_{v_0} + \theta_0 x_{v_1} \quad (6)$$

$$g_{v_4\theta}: \mathbb{R}^{\{v_2, v_3\}} \rightarrow \mathbb{R}: \quad x \mapsto x_{v_2} + x_{v_3}^{\theta_1} . \quad (7)$$

The compute graph $(V, D, D', E, \Theta, \{g_{v_3\theta}, g_{v_4\theta}\})$ defines the function

$$f_\theta: \mathbb{R}^V \rightarrow \mathbb{R}^{D'}: \quad x \mapsto x_{v_2} + (x_{v_0} + \theta_0 x_{v_1})^{\theta_1} . \quad (8)$$

Definition. Let $(V, D, D', E, \Theta, \{g_{v\theta} : \mathbb{R}^{P_v} \rightarrow \mathbb{R}\}_{v \in (D \cup D') \setminus V, \theta \in \Theta})$ a compute graph with $|D'| = 1$ and $\Theta = \mathbb{R}^J$ for some finite set $J \neq \emptyset$. Let f be the family of functions defined by this compute graph. The **l_2 -regularized logistic regression problem** wrt. f , labeled data $T = (S, \mathbb{R}^V, x, y)$ and $\sigma \in \mathbb{R}^+$ has the form

$$\min_{\theta \in \mathbb{R}^J} \frac{1}{|S|} \sum_{s \in S} \left(-y_s f_\theta(x_s) + \log \left(1 + 2^{f_\theta(x)} \right) \right) + \frac{\log e}{2\sigma^2} \|\theta\|^2 . \quad (9)$$

Remark.

- ▶ (9) generalizes l_2 -regularized linear logistic regression
- ▶ (9) can be non-convex in case f is not linear in θ .
- ▶ If the partial derivative of f wrt. θ_j exists for all $j \in J$, we can search for a local minimum using a steepest descent algorithm.
- ▶ To do so, we describe two techniques for computing $\nabla_\theta f$, **forward propagation** and **backward propagation**.

Learning of composite functions (deep learning)

Lemma. Let $j \in J$. For any $v \in V$: $\frac{\partial \alpha_{v\theta}}{\partial \theta_j} = 0$. For any $v \in (D \cup D') \setminus V$:

$$\frac{\partial \alpha_{v\theta}}{\partial \theta_j} = \sum_{u \in (A_v \cup \{v\}) \setminus V} \frac{\partial g_{u\theta}}{\partial \theta_j} \Delta_{uv} \quad (10)$$

with

$$\Delta_{uv} := \sum_{(V', E') \in \mathcal{P}(u, v)} \prod_{(u', v') \in E'} \frac{\partial g_{v'\theta}}{\partial \alpha_{u'\theta}}. \quad (11)$$

Remark. For any node u : $\Delta_{uu} = 1$. For any u, v with $\mathcal{P}(u, v) = \emptyset$: $\Delta_{uv} = 0$.

Proof (idea).

$$\begin{aligned} \frac{\partial \alpha_{v\theta}}{\partial \theta_j} &= \frac{\partial g_{v\theta}}{\partial \theta_j} + \sum_{u \in P_v} \frac{\partial g_{v\theta}}{\partial \alpha_{u\theta}} \frac{\partial \alpha_{u\theta}}{\partial \theta_j} \quad (12) \\ &= \frac{\partial g_{v\theta}}{\partial \theta_j} + \sum_{u \in P_v} \frac{\partial g_{v\theta}}{\partial \alpha_{u\theta}} \frac{\partial g_{u\theta}}{\partial \theta_j} + \sum_{u \in P_v} \sum_{u' \in P_u} \frac{\partial g_{v\theta}}{\partial \alpha_{u\theta}} \frac{\partial g_{u\theta}}{\partial \alpha_{u'\theta}} \frac{\partial \alpha_{u'\theta}}{\partial \theta_j} \\ &= \text{repeated application (12)} \\ &= \sum_{u \in (A_v \cup \{v\}) \setminus V} \frac{\partial g_{u\theta}}{\partial \theta_j} \sum_{(V', E') \in \mathcal{P}(u, v)} \prod_{(u', v') \in E'} \frac{\partial g_{v'\theta}}{\partial \alpha_{u'\theta}} \end{aligned}$$

Lemma (forward propagation). For all nodes $u \neq w$ such that $\mathcal{P}(u, w) \neq \emptyset$:

$$\Delta_{uw} = \sum_{v \in P_w} \frac{\partial g_{w\theta}}{\partial \alpha_{v\theta}} \Delta_{uv} \quad (13)$$

Proof.

$$\begin{aligned} \Delta_{uw} &= \sum_{(V', E') \in \mathcal{P}(u, w)} \prod_{(u', v') \in E'} \frac{\partial g_{v'\theta}}{\partial \alpha_{u'\theta}} \\ &= \sum_{v \in P_w} \sum_{(V'', E'') \in \mathcal{P}(u, v)} \prod_{(u', v') \in E'' \cup \{v, w\}} \frac{\partial g_{v'\theta}}{\partial \alpha_{u'\theta}} \\ &= \sum_{v \in P_w} \frac{\partial g_{w\theta}}{\partial \alpha_{v\theta}} \sum_{(V'', E'') \in \mathcal{P}(u, v)} \prod_{(u', v') \in E''} \frac{\partial g_{v'\theta}}{\partial \alpha_{u'\theta}} \\ &= \sum_{v \in P_w} \frac{\partial g_{w\theta}}{\partial \alpha_{v\theta}} \Delta_{uv} \end{aligned}$$

□

Learning of composite functions (deep learning)

The **forward propagation algorithm** computes Δ_{uw} for one node u and all nodes w . It is defined wrt. an arbitrary partial order $<_P$ of the nodes such that

$$\forall w \in D \cup D' \quad \forall w' \in P_w: \quad w' <_P w . \quad (14)$$

Input:

Compute graph $(V, D, D', E, \Theta, \{g_{v\theta}: \mathbb{R}^{P_v} \rightarrow \mathbb{R}\}_{v \in (D \cup D') \setminus V, \theta \in \Theta})$

Node $u \in V \cup D \cup D'$

for w ordered by $<_P$ (14)

if $w = u$

$$\Delta_{uw} := 1$$

else if $\mathcal{P}(u, w) = \emptyset$

$$\Delta_{uw} := 0$$

else

$$\Delta_{uw} := \sum_{v \in P_w} \frac{\partial g_{w\theta}}{\partial \alpha_{v\theta}} \Delta_{uv} \quad (13)$$

Lemma (backward propagation). For all nodes $u \neq w$ such that $\mathcal{P}(u, w) \neq \emptyset$:

$$\Delta_{uw} = \sum_{v \in C_u} \frac{\partial g_{v\theta}}{\partial \alpha_{u\theta}} \Delta_{vw} \quad (15)$$

Proof.

$$\begin{aligned} \Delta_{uw} &= \sum_{(V', E') \in \mathcal{P}(u, w)} \prod_{(u', v') \in E'} \frac{\partial g_{v'\theta}}{\partial \alpha_{u'\theta}} \\ &= \sum_{v \in C_u} \sum_{(V'', E'') \in \mathcal{P}(v, w)} \prod_{(u', v') \in E'' \cup \{(u, v)\}} \frac{\partial g_{v'\theta}}{\partial \alpha_{u'\theta}} \\ &= \sum_{v \in C_u} \frac{\partial g_{v\theta}}{\partial \alpha_{u\theta}} \sum_{(V'', E'') \in \mathcal{P}(v, w)} \prod_{(u', v') \in E''} \frac{\partial g_{v'\theta}}{\partial \alpha_{u'\theta}} \\ &= \sum_{v \in C_u} \frac{\partial g_{v\theta}}{\partial \alpha_{u\theta}} \Delta_{vw} \end{aligned}$$

□

Learning of composite functions (deep learning)

The **backward propagation algorithm** computes Δ_{uw} for one node w and all nodes u . It is defined wrt. an arbitrary partial order $<_C$ of the nodes such that

$$\forall u \in V \cup D \quad \forall v \in C_u: \quad v <_C u . \quad (16)$$

Input:

Compute graph $(V, D, D', E, \Theta, \{g_{v\theta}: \mathbb{R}^{P_v} \rightarrow \mathbb{R}\}_{v \in (D \cup D') \setminus V, \theta \in \Theta})$

Node $w \in V \cup D \cup D'$

for u ordered by $<_C$ (16)

if $u = w$

$$\Delta_{uw} := 1$$

else if $\mathcal{P}(u, w) = \emptyset$

$$\Delta_{uw} := 0$$

else

$$\Delta_{uw} := \sum_{v \in C_u} \frac{\partial g_{v\theta}}{\partial \alpha_{u\theta}} \Delta_{vw} \quad (15)$$
