Train artificial neural networks

ANDREAS LANGER, LUND UNIVERSITY

Components of a neural network

A neural network is specified by the following components:

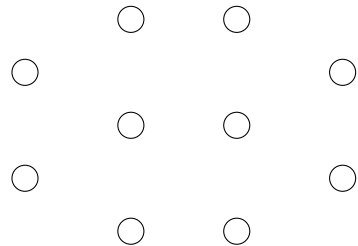
- a directed graph (V, E)

LUND UNIVERSITY

Components of a neural network

A neural network is specified by the following components:

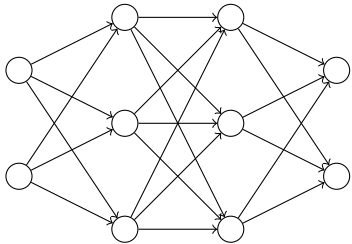
- a directed graph (V, E)
  - V .... set of knots/neurons



Components of a neural network

A neural network is specified by the following components:

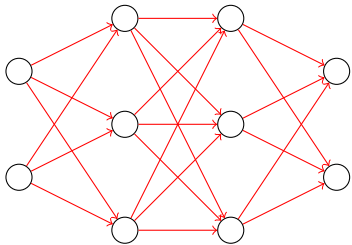
- a directed graph (V, E)
  - V .... set of knots/neurons
  - E .... set of (directed) edges, which connect the nodes



Components of a neural network

A neural network is specified by the following components:

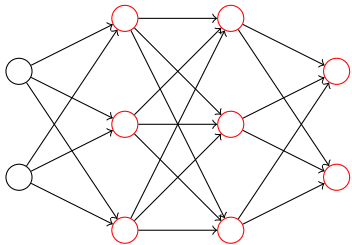
- a directed graph (V, E)
  - V .... set of knots/neurons
  - E .... set of (directed) edges, which connect the nodes
- weights $w : E \to \mathbb{R}$



Components of a neural network

A neural network is specified by the following components:

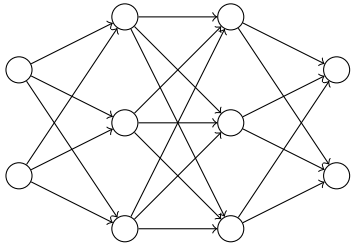
- a directed graph (V, E)
  - V .... set of knots/neurons
  - E .... set of (directed) edges, which connect the nodes
- weights $w : E \to \mathbb{R}$
- Family of activation functions $\sigma := \{\sigma_v : \mathbb{R} \to \mathbb{R},\ v \in V\}$.



Components of a neural network

A neural network is specified by the following components:

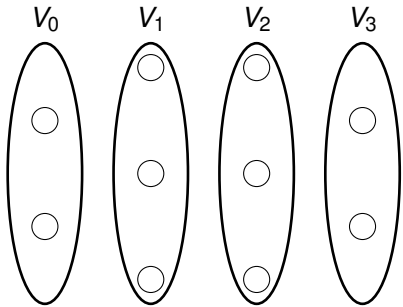
- a directed graph (V, E)
  - V .... set of knots/neurons
  - E .... set of (directed) edges, which connect the nodes
- weights $w : E \to \mathbb{R}$
- Family of activation functions $\sigma := \{\sigma_v : \mathbb{R} \to \mathbb{R},\ v \in V\}$.



$\Rightarrow$ Neural networks are characterized by the tuple (V, E, σ, w).

Requirements to the network

(i) The net is layered: Set of nodes V has decomposition into $T+1$, $T \in \mathbb{N}$, (non-empty) disjoint subsets $V_i \subset V$, $i = 0, \ldots, T$, i.e. $V = \bigcup_{i=0}^{T} V_i$.

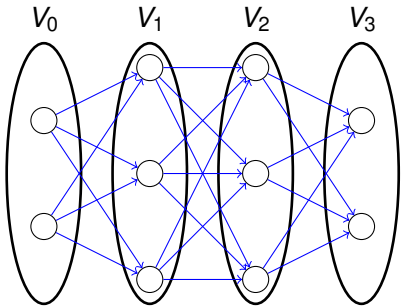


Requirements to the network

(i) The net is layered: Set of nodes V has decomposition into $T+1$, $T \in \mathbb{N}$, (non-empty) disjoint subsets $V_i \subset V$, $i = 0, \ldots, T$, i.e. $V = \bigcup_{i=0}^{T} V_i$.

(ii) Each edge in E connects some node in V_{t-1} with a node in V_t, $t = 1, \ldots, T$.

I.e. edges always run from one layer to the following one, $E \subseteq \bigcup_{t=1}^{T} V_{t-1} \times V_t$.



Requirements to the network

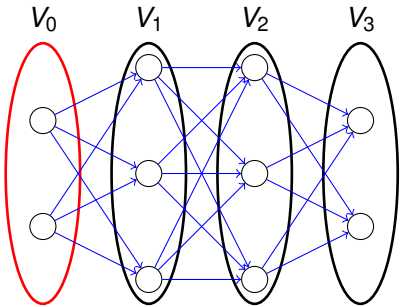
(i) The net is layered: Set of nodes V has decomposition into $T+1$, $T \in \mathbb{N}$, (non-empty) disjoint subsets $V_i \subset V$, $i = 0, \dots, T$, i.e. $V = \bigcup_{i=0}^{T} V_i$.

(ii) Each edge in E connects some node in V_{t-1} with a node in V_t, $t = 1, \dots, T$.

I.e. edges always run from one layer to the following one, $E \subseteq \bigcup_{t=1}^{T} V_{t-1} \times V_t$.

Notation:

- V_0 .... input layer. Let $n \in \mathbb{N}$ be the dimension of the input space, then $|V_0| = n + 1$.



Requirements to the network

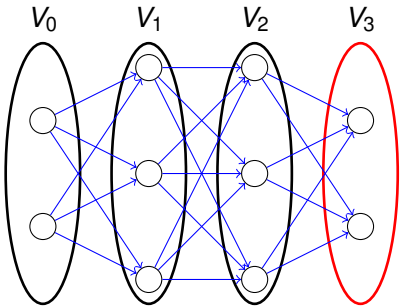
(i) The net is layered: Set of nodes V has decomposition into $T+1$, $T \in \mathbb{N}$, (non-empty) disjoint subsets $V_i \subset V$, $i = 0, \dots, T$, i.e. $V = \bigcup_{i=0}^{T} V_i$.

(ii) Each edge in E connects some node in V_{t-1} with a node in V_t, $t = 1, \dots, T$.

I.e. edges always run from one layer to the following one, $E \subseteq \bigcup_{t=1}^{T} V_{t-1} \times V_t$.

Notation:

- V_0 .... input layer. Let $n \in \mathbb{N}$ be the dimension of the input space, then $|V_0| = n + 1$.
- V_T .... output layer



Requirements to the network

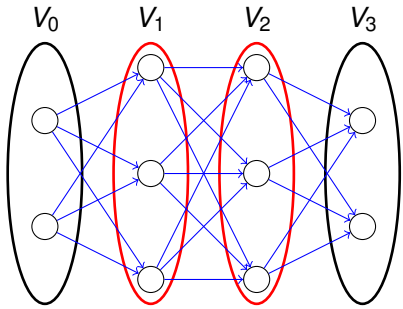
(i) The net is layered: Set of nodes V has decomposition into $T+1$, $T \in \mathbb{N}$, (non-empty) disjoint subsets $V_i \subset V$, $i = 0, \ldots, T$, i.e. $V = \bigcup_{i=0}^{T} V_i$.

(ii) Each edge in E connects some node in V_{t-1} with a node in V_t, $t = 1, \ldots, T$.

I.e. edges always run from one layer to the following one, $E \subseteq \bigcup_{t=1}^{T} V_{t-1} \times V_t$.

Notation:

- V_0 .... input layer. Let $n \in \mathbb{N}$ be the dimension of the input space, then $|V_0| = n + 1$.
- V_T .... output layer
- $V_1, \ldots, V_{T-1}$ .... hidden layers



Requirements to the network

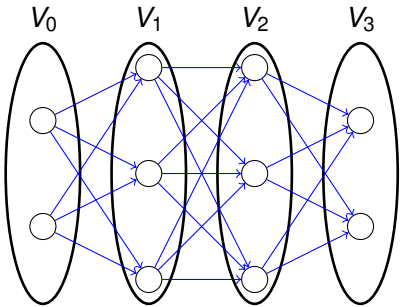
(i) The net is layered: Set of nodes V has decomposition into $T+1$, $T \in \mathbb{N}$, (non-empty) disjoint subsets $V_i \subset V$, $i = 0, \dots, T$, i.e. $V = \bigcup_{i=0}^{T} V_i$.

(ii) Each edge in E connects some node in V_{t-1} with a node in V_t, $t = 1, \dots, T$.

I.e. edges always run from one layer to the following one, $E \subseteq \bigcup_{t=1}^{T} V_{t-1} \times V_t$.

Notation:

- V_0 .... input layer. Let $n \in \mathbb{N}$ be the dimension of the input space, then $|V_0| = n + 1$.
- V_T .... output layer
- $V_1, \dots, V_{T-1}$ .... hidden layers
- $T \in \mathbb{N}$ .... depth of the network



Requirements to the network

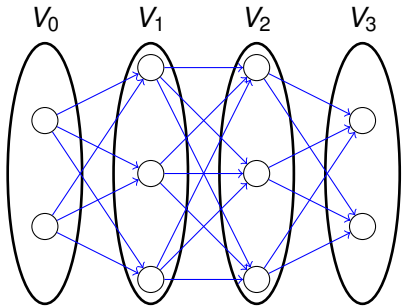
(i) The net is layered: Set of nodes V has decomposition into $T+1$, $T \in \mathbb{N}$, (non-empty) disjoint subsets $V_i \subset V$, $i = 0, \dots, T$, i.e. $V = \bigcup_{i=0}^{T} V_i$.

(ii) Each edge in E connects some node in V_{t-1} with a node in V_t, $t = 1, \dots, T$.

I.e. edges always run from one layer to the following one, $E \subseteq \bigcup_{t=1}^{T} V_{t-1} \times V_t$.

Notation:

- V_0 .... input layer. Let $n \in \mathbb{N}$ be the dimension of the input space, then $|V_0| = n + 1$.
- V_T .... output layer
- $V_1, \dots, V_{T-1}$ .... hidden layers
- $T \in \mathbb{N}$ .... depth of the network
- $\max_{t=0,\dots,T} |V_t|$ .... width of the network

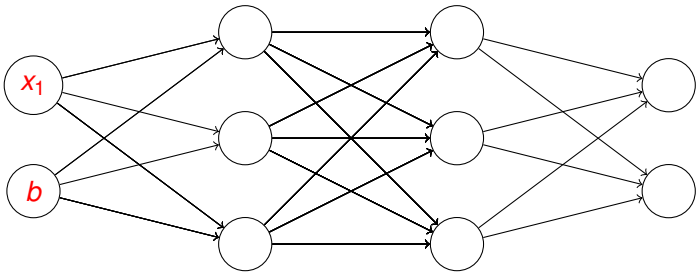


Input – Output

LUND UNIVERSITY

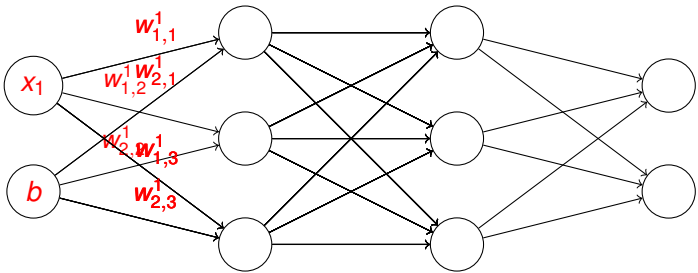
Input – Output

1. V_0 passes $x \in \mathbb{R}^n$ to V_1, i.e. $o_{v_i^0}(x) = x_i$, $i = 1, \ldots, n$, and $o_{v_{n+1}^0}(x) = b$.



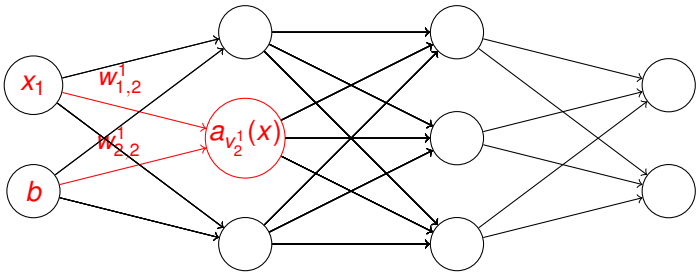
Input – Output

1. V_0 passes $x \in \mathbb{R}^n$ to V_1, i.e. $o_{v_i^0}(x) = x_i$, $i = 1, \ldots, n$, and $o_{v_{n+1}^0}(x) = b$.
2. For $t = 1, \ldots, T$ we have: Let $w_{i,j}^t \in \mathbb{R}$ weight of edge v_i^{t-1} to v_j^t.



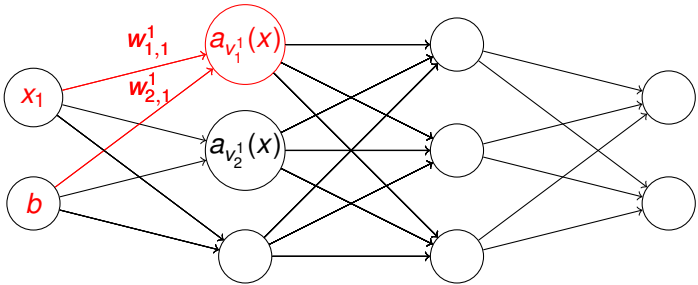
Input – Output

1. V_0 passes $x \in \mathbb{R}^n$ to V_1, i.e. $o_{v_i^0}(x) = x_i$, $i = 1, \ldots, n$, and $o_{v_{n+1}^0}(x) = b$.
2. For $t = 1, \ldots, T$ we have: Let $w_{i,j}^t \in \mathbb{R}$ weight of edge v_i^{t-1} to v_j^t.
   Input for v_2^1: $a_{v_2^1}(x) = w_{1,2}^1 x_1 + w_{2,2}^1 b = \sum_{i=1}^{|V_0|} w_{i,2}^1 o_{v_i^0}(x)$



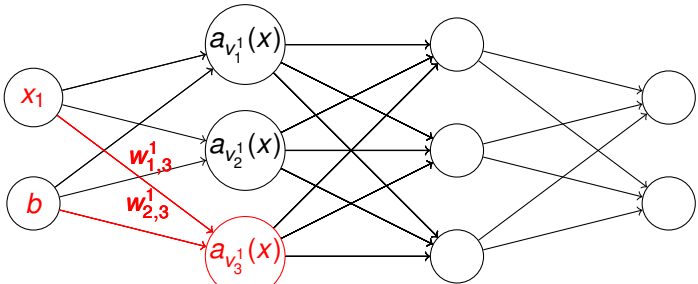
Input – Output

1. V_0 passes $x \in \mathbb{R}^n$ to V_1, i.e. $o_{v_i^0}(x) = x_i$, $i = 1, \ldots, n$, and $o_{v_{n+1}^0}(x) = b$.
2. For $t = 1, \ldots, T$ we have: Let $w_{i,j}^t \in \mathbb{R}$ weight of edge v_i^{t-1} to v_j^t.
   Input for v_1^1: $a_{v_1^1}(x) = w_{1,1}^1 x_1 + w_{2,1}^1 b = \sum_{i=1}^{|V_0|} w_{i,1}^1 o_{v_i^0}(x)$



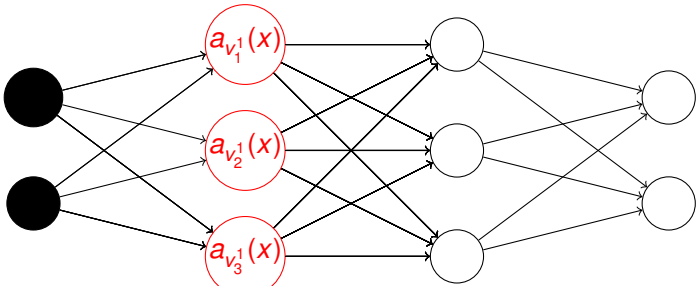
Input – Output

1. V_0 passes $x \in \mathbb{R}^n$ to V_1, i.e. $o_{v_i^0}(x) = x_i$, $i = 1, \ldots, n$, and $o_{v_{n+1}^0}(x) = b$.
2. For $t = 1, \ldots, T$ we have: Let $w_{i,j}^t \in \mathbb{R}$ weight of edge v_i^{t-1} to v_j^t.
   Input for v_3^1: $a_{v_3^1}(x) = w_{1,3}^1 x_1 + w_{2,3}^1 b = \sum_{i=1}^{|V_0|} w_{i,3}^1 o_{v_i^0}(x)$



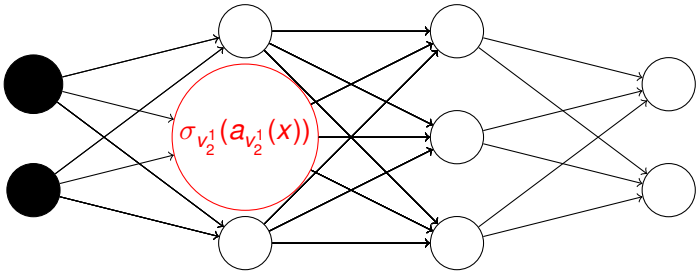
Input – Output

1. V_0 passes $x \in \mathbb{R}^n$ to V_1, i.e. $o_{v_i^0}(x) = x_i$, $i = 1, \ldots, n$, and $o_{v_{n+1}^0}(x) = b$.
2. For $t = 1, \ldots, T$ we have: Let $w_{i,j}^t \in \mathbb{R}$ weight of edge v_i^{t-1} to v_j^t.
   Input for v_j^t: $a_{v_j^t}(x) = \sum_{i=1}^{|V_{t-1}|} w_{i,j}^t o_{v_i^{t-1}}(x)$



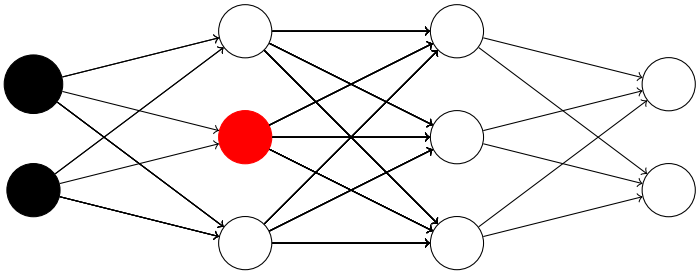
Input – Output

1. V_0 passes $x \in \mathbb{R}^n$ to V_1, i.e. $o_{v_i^0}(x) = x_i$, $i = 1, \ldots, n$, and $o_{v_{n+1}^0}(x) = b$.
2. For $t = 1, \ldots, T$ we have: Let $w_{i,j}^t \in \mathbb{R}$ weight of edge v_i^{t-1} to v_j^t.
   Input for v_j^t: $a_{v_j^t}(x) = \sum_{i=1}^{|V_{t-1}|} w_{i,j}^t o_{v_i^{t-1}}(x)$
   Output: $o_{v_2^1}(x) = \sigma_{v_2^1}(a_{v_2^1}(x))$



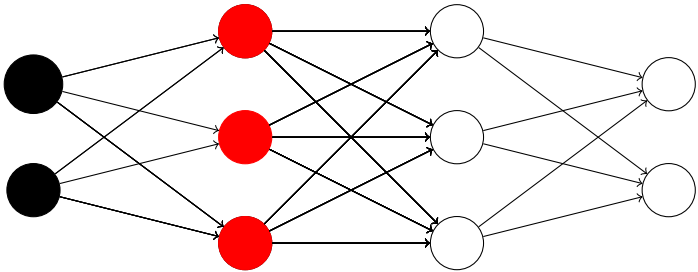
Input – Output

1. V_0 passes $x \in \mathbb{R}^n$ to V_1, i.e. $o_{v_i^0}(x) = x_i$, $i = 1, \ldots, n$, and $o_{v_{n+1}^0}(x) = b$.
2. For $t = 1, \ldots, T$ we have: Let $w_{i,j}^t \in \mathbb{R}$ weight of edge v_i^{t-1} to v_j^t.
   Input for v_j^t: $a_{v_j^t}(x) = \sum_{i=1}^{|V_{t-1}|} w_{i,j}^t o_{v_i^{t-1}}(x)$
   Output: $o_{v_i^t}(x) = \sigma_{v_i^t}(a_{v_i^t}(x))$



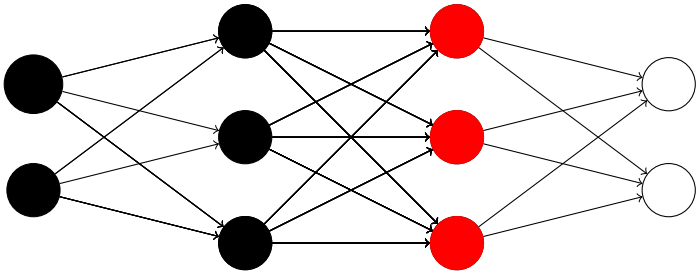
Input – Output

1. V_0 passes $x \in \mathbb{R}^n$ to V_1, i.e. $o_{v_i^0}(x) = x_i$, $i = 1, \ldots, n$, and $o_{v_{n+1}^0}(x) = b$.
2. For $t = 1, \ldots, T$ we have: Let $w_{i,j}^t \in \mathbb{R}$ weight of edge v_i^{t-1} to v_j^t.
   Input for v_j^t: $a_{v_j^t}(x) = \sum_{i=1}^{|V_{t-1}|} w_{i,j}^t o_{v_i^{t-1}}(x)$
   Output: $o_{v_i^t}(x) = \sigma_{v_i^t}(a_{v_i^t}(x))$



Input – Output

1. V_0 passes $x \in \mathbb{R}^n$ to V_1, i.e. $o_{v_i^0}(x) = x_i$, $i = 1, \ldots, n$, and $o_{v_{n+1}^0}(x) = b$.
2. For $t = 1, \ldots, T$ we have: Let $w_{i,j}^t \in \mathbb{R}$ weight of edge v_i^{t-1} to v_j^t.
   Input for v_j^t: $a_{v_j^t}(x) = \sum_{i=1}^{|V_{t-1}|} w_{i,j}^t o_{v_i^{t-1}}(x)$
   Output: $o_{v_i^t}(x) = \sigma_{v_i^t}(a_{v_i^t}(x))$



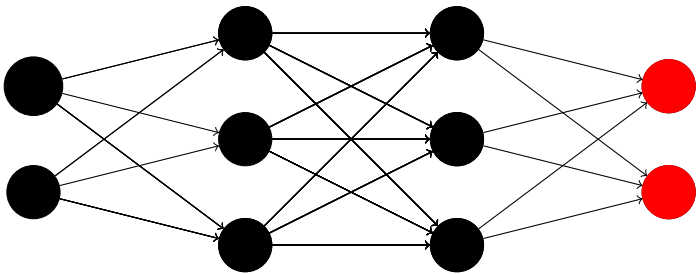
Input – Output

1. V_0 passes $x \in \mathbb{R}^n$ to V_1, i.e. $o_{v_i^0}(x) = x_i$, $i = 1, \dots, n$, and $o_{v_{n+1}^0}(x) = b$.
2. For $t = 1, \dots, T$ we have: Let $w_{i,j}^t \in \mathbb{R}$ weight of edge v_i^{t-1} to v_j^t.

   Input for v_j^t: $a_{v_j^t}(x) = \sum_{i=1}^{|V_{t-1}|} w_{i,j}^t o_{v_i^{t-1}}(x)$

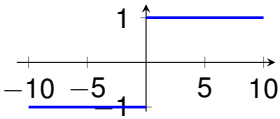
   Output: $o_{v_i^t}(x) = \sigma_{v_i^t}(a_{v_i^t}(x))$

   Output of the net: $h(x) := (o_{v_i^T}(x))_{i=1}^{|V_T|} \in \mathbb{R}^{|V_T|}$



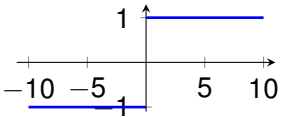
Examples of activation functions

- Sign function: $\sigma_{\text{sign}}(x) := \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$

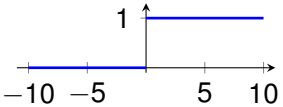


Examples of activation functions

- Sign function: $\sigma_{\text{sign}}(x) := \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$



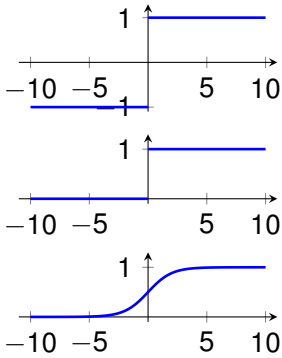
- Unit step function: $\sigma_{\text{step}}(x) := \begin{cases} +1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$



LUND UNIVERSITY

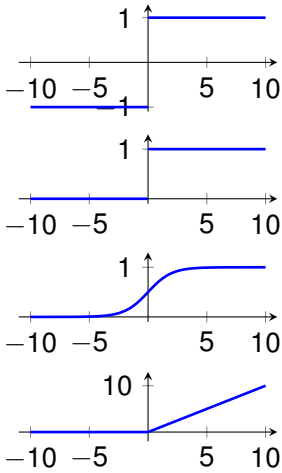
Examples of activation functions

- Sign function: $\sigma_{\text{sign}}(x) := \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$
- Unit step function: $\sigma_{\text{step}}(x) := \begin{cases} +1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$
- Sigmoid funktion: $\sigma_{\text{sig}}(x) = \frac{1}{1+e^{-x}}$

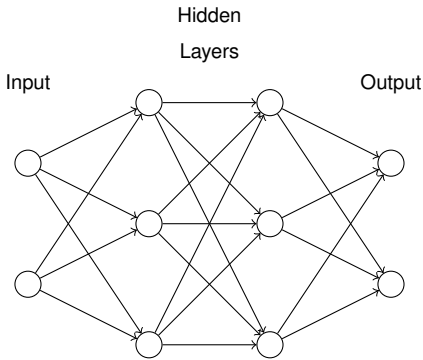


Examples of activation functions

- Sign function: $\sigma_{\text{sign}}(x) := \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$
- Unit step function: $\sigma_{\text{step}}(x) := \begin{cases} +1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$
- Sigmoid funktion: $\sigma_{\text{sig}}(x) = \frac{1}{1+e^{-x}}$
- Rectified linear unit (ReLU): $\sigma_{\text{ReLU}}(x) := \max\{0, x\}$

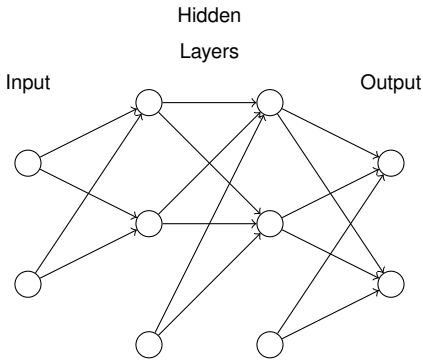


Neural Nets



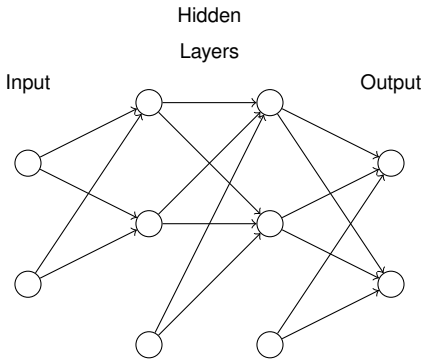
Feedforward
neural network (FNN)

Neural Nets

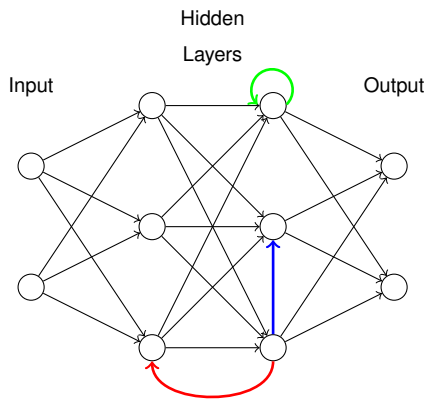


Feedforward
neural network (FNN)

Neural Nets



Feedforward
neural network (FNN)



Recurrent neural network (RNN)

What are artificial neural networks used for?

LUND UNIVERSITY

What are artificial neural networks used for?

The goal of machine learning: Use data to develop automated complex models or algorithms.

LUND UNIVERSITY

What are artificial neural networks used for?

The goal of machine learning: Use data to develop automated complex models or algorithms.

Application of machine learning:

- Task too complex to be “directly” programmed.
  - Tasks performed by humans: Driving a car, interpreting images, ...
  - Tasks beyond human capabilities: Analysis of large and complex data sets (transforming medical data into medical knowledge, ...).

LUND UNIVERSITY

What are artificial neural networks used for?

The goal of machine learning: Use data to develop automated complex models or algorithms.

Application of machine learning:

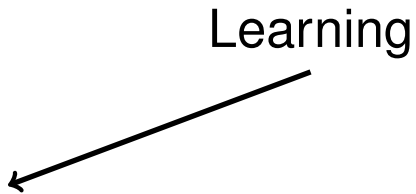
- Task too complex to be “directly” programmed.
  - Tasks performed by humans: Driving a car, interpreting images, ...
  - Tasks beyond human capabilities: Analysis of large and complex data sets (transforming medical data into medical knowledge, ...).
- Adaptivity: Machine learning tools adapt to the input data. Example:
  - Decoding of handwriting, where the program can adapt to handwriting of different users.
  - Spam filter: automatic adaptation to the change of spam e-mails.

LUND UNIVERSITY

Learning

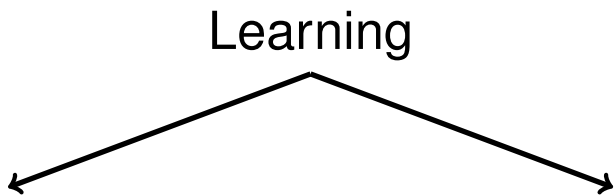
Supervised/unsupervised learning

LUND UNIVERSITY



Supervised learning: Learning based on training data $\Rightarrow$ Mapping rule that can predict classification of unknown data.
Example: Emails classified as spam or not spam $\rightarrow$ Spam-Filter.

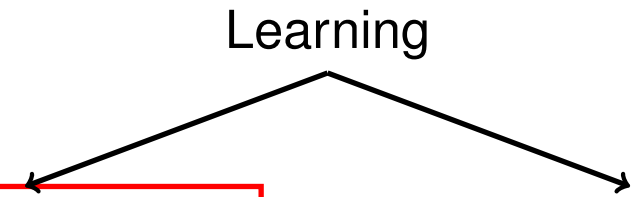
Supervised/unsupervised learning



Supervised learning: Learning based on training data $\Rightarrow$ Mapping rule that can predict classification of unknown data.
Example: Emails classified as spam or not spam $\rightarrow$ Spam-Filter.

Unsupervised learning: No training data.
Task: Recognize patterns in the data and/or model data.
Example: Data compression.

Supervised/unsupervised learning



Supervised learning: Learning based on training data $\Rightarrow$ Mapping rule that can predict classification of unknown data.
Example: Emails classified as spam or not spam $\rightarrow$ Spam-Filter.

Unsupervised learning: No training data.
Task: Recognize patterns in the data and/or model data.
Example: Data compression.

LUND UNIVERSITY

Components of supervised learning

- Domain (input space) X: Consists of elements, whose features we would like to learn.

LUND UNIVERSITY

Components of supervised learning

- Domain (input space) X: Consists of elements, whose features we would like to learn.
- Output space Y: Consists of possible features (labels) of the input $x \in X$.

LUND UNIVERSITY

Components of supervised learning

- Domain (input space) X: Consists of elements, whose features we would like to learn.
- Output space Y: Consists of possible features (labels) of the input $x \in X$.
- Training data $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (X \times Y)^m$ correctly labeled. Assumption: (x_i, y_i) are independently identically distributed (i.i.d.) w.r.t. a probability distribution P. We write $S \sim P^m$.

LUND UNIVERSITY

Components of supervised learning

- Domain (input space) X: Consists of elements, whose features we would like to learn.
- Output space Y: Consists of possible features (labels) of the input $x \in X$.
- Training data $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (X \times Y)^m$ correctly labeled. Assumption: (x_i, y_i) are independently identically distributed (i.i.d.) w.r.t. a probability distribution P. We write $S \sim P^m$.
- Hypothesis space $H \subset Y^X$ set of possible predictors $h : X \to Y$. E.g.: $H = \{h : X \to Y \mid h \text{ polynomial of order } \leq d \in \mathbb{N}\}$.

LUND UNIVERSITY

Components of supervised learning

- Domain (input space) X: Consists of elements, whose features we would like to learn.
- Output space Y: Consists of possible features (labels) of the input $x \in X$.
- Training data $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (X \times Y)^m$ correctly labeled. Assumption: (x_i, y_i) are independently identically distributed (i.i.d.) w.r.t. a probability distribution P. We write $S \sim P^m$.
- Hypothesis space $H \subset Y^X$ set of possible predictors $h : X \to Y$. E.g.: $H = \{h : X \to Y \mid \ h \text{ polynomial of order } \leq d \in \mathbb{N}\}$.
- Learning algorithm $A : \bigcup_{m \in \mathbb{N}} (X \times Y)^m \to H$, $S \mapsto h_S$: The task of the learning algorithm is to link a data point $x \in X$ with a correct label.

LUND UNIVERSITY

Risk

- Loss function $\ell : Y \times Y \to [0, \infty)$:
  Example:
  1. 0-1 loss: $\ell_{0-1}(y, h(x)) := \begin{cases} 0 & \text{falls } h(x) = y \\ 1 & \text{falls } h(x) \neq y \end{cases}$
  2. Square loss: $\ell_2(y, h(x)) := (h(x) - y)^2$

LUND UNIVERSITY

Risk

- Loss function $\ell : Y \times Y \to [0, \infty)$:
  Example:
  1. 0-1 loss: $\ell_{0-1}(y, h(x)) := \begin{cases} 0 & \text{falls } h(x) = y \\ 1 & \text{falls } h(x) \neq y \end{cases}$
  2. Square loss: $\ell_2(y, h(x)) := (h(x) - y)^2$
- Risk $\mathcal{R}_P : H \to \mathbb{R}^+$ with $\mathcal{R}_P(h) = \mathop{\mathbb{E}}_{(x,y)\sim P}(\ell(y, h(x))) = \int_{X \times Y} \ell(y, h(x)) \, \mathrm{d}P(x, y)$

LUND UNIVERSITY

Risk

- Loss function $\ell : Y \times Y \to [0, \infty)$:
  Example:
  1. 0-1 loss: $\ell_{0-1}(y, h(x)) := \begin{cases} 0 & \text{falls } h(x) = y \\ 1 & \text{falls } h(x) \neq y \end{cases}$
  2. Square loss: $\ell_2(y, h(x)) := (h(x) - y)^2$
- Risk $\mathcal{R}_P : H \to \mathbb{R}^+$ with $\mathcal{R}_P(h) = \mathop{\mathbb{E}}_{(x,y)\sim P}(\ell(y, h(x))) = \int_{X \times Y} \ell(y, h(x)) \, \mathrm{d}P(x, y)$

GOAL: Find hypotheses $h^* : X \to Y$ that minimizes the risk:

$$h^* \in \arg\min_{h \in H} \mathcal{R}_P(h)$$

LUND UNIVERSITY

Risk

- Loss function $\ell : Y \times Y \to [0, \infty)$:
  Example:
  1. 0-1 loss: $\ell_{0-1}(y, h(x)) := \begin{cases} 0 & \text{falls } h(x) = y \\ 1 & \text{falls } h(x) \neq y \end{cases}$
  2. Square loss: $\ell_2(y, h(x)) := (h(x) - y)^2$
- Risk $\mathcal{R}_P : H \to \mathbb{R}^+$ with $\mathcal{R}_P(h) = \mathop{\mathbb{E}}_{(x,y)\sim P}(\ell(y, h(x))) = \int_{X \times Y} \ell(y, h(x)) \, \mathrm{d}P(x, y)$

GOAL: Find hypotheses $h^* : X \to Y$ that minimizes the risk:

$$h^* \in \arg\min_{h \in H} \mathcal{R}_P(h)$$

Attention: The probability distribution P is not known!

LUND UNIVERSITY

Types of learning problems

Regression:

- Output space Y is continuous; e.g. $Y = \mathbb{R}$
- Loss function measures continuous distance between points; e.g. $\ell = \ell_2$
  $\Rightarrow \mathcal{R}_P(h) = \mathop{\mathbb{E}}_{(x,y)\sim P}(\ell(y, h(x))) = \mathop{\mathbb{E}}_{(x,y)\sim P}[|h(x) - y|^2]$

LUND UNIVERSITY

Types of learning problems

Regression:

- Output space Y is continuous; e.g. $Y = \mathbb{R}$
- Loss function measures continuous distance between points; e.g. $\ell = \ell_2$
  $\Rightarrow \mathcal{R}_P(h) = \mathop{\mathbb{E}}_{(x,y)\sim P}(\ell(y, h(x))) = \mathop{\mathbb{E}}_{(x,y)\sim P}[|h(x) - y|^2]$

Classification:

- Output space Y is discrete; e.g. $Y = \{0, 1\}$ (binary classification)
- Usual loss function is $\ell = \ell_{0-1}$
  $\Rightarrow \mathcal{R}_P(h) = \mathop{\mathbb{E}}_{(x,y)\sim P}(\ell(y, h(x))) = \mathop{\mathbb{P}}_{(x,y)\sim P}[h(x) \neq y]$

LUND UNIVERSITY

FNN - learning problem

How does the FNN fit into the statistical learning problem?

LUND UNIVERSITY

FNN - learning problem

How does the FNN fit into the statistical learning problem?

- FNN maps an element $x \in X = \mathbb{R}^{|V_0|-1}$ into the output space $Y = \mathbb{R}^{|V_T|}$. I.e. let (V, E, σ, w) fixed, then

$$h_{V,E,\sigma,w} : X \to Y.$$

FNN forms a hypothesis class

$$H = \{h_{V,E,\sigma,w} : X \to Y \mid \ V, E, \sigma, w\}.$$

LUND UNIVERSITY

FNN - learning problem

How does the FNN fit into the statistical learning problem?

- FNN maps an element $x \in X = \mathbb{R}^{|V_0|-1}$ into the output space $Y = \mathbb{R}^{|V_T|}$. I.e. let (V, E, σ, w) fixed, then

$$h_{V,E,\sigma,w} : X \to Y.$$

FNN forms a hypothesis class

$$H = \{h_{V,E,\sigma,w} : X \to Y \mid \ V, E, \sigma, w\}.$$

- We assume: Architecture (V, E, σ) given

$$\Rightarrow \ H_{V,E,\sigma} := \{h_{V,E,\sigma,w} : X \to Y \mid \ w : E \to \mathbb{R}\}.$$

- Learning means, choose $h_{V,E,\sigma,w} \in H_{V,E,\sigma}$ by fitting the weights w to the training data S.

LUND UNIVERSITY

Empirical risk

How can a learning algorithm try to minimize the risk via the hypothesis class without knowing the probability distribution?

LUND UNIVERSITY

Empirical risk

How can a learning algorithm try to minimize the risk via the hypothesis class without knowing the probability distribution?

Remedy:

- Training data: We can compute the empirical risk

$$\mathcal{R}_S(h) := \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, h(x_i)).$$

$\Rightarrow$ empirical risk minimization (ERM)

$$\inf_{h \in H} \mathcal{R}_S(h)$$

If $|H| < \infty \Rightarrow$ existence of a minimum is guaranteed.

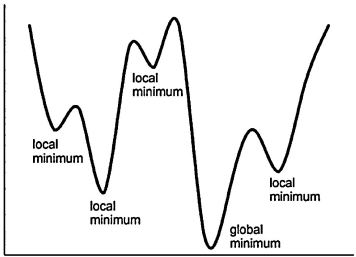
LUND UNIVERSITY

Characterization of a Solution

For a function $f : H \to \mathbb{R}$ a candidate $w^* \in H$ is called

- global minimizer of f in H, if

$$f(w^*) \leq f(w) \quad \text{for all } w \in H$$



Characterization of a Solution

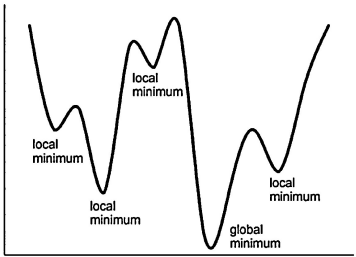
For a function $f : H \to \mathbb{R}$ a candidate $w^* \in H$ is called

(i) global minimizer of f in H, if

$$f(w^*) \leq f(w) \quad \text{for all } w \in H$$

(ii) strict global minimizer of f in H, if

$$f(w^*) < f(w) \quad \text{for all } w \in H \setminus \{w^*\}.$$



Characterization of a Solution

For a function $f : H \to \mathbb{R}$ a candidate $w^* \in H$ is called

(i) global minimizer of f in H, if

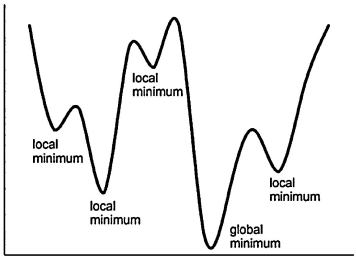
$$f(w^*) \leq f(w) \quad \text{for all } w \in H$$

(ii) strict global minimizer of f in H, if

$$f(w^*) < f(w) \quad \text{for all } w \in H \setminus \{w^*\}.$$

(iii) local minimizer of f in H, if there exists a neighborhood of w^* denoted by $N(w^*)$ such that

$$f(w^*) \leq f(w) \quad \text{for all } w \in H \cap N(w^*).$$



Characterization of a Solution

For a function $f : H \to \mathbb{R}$ a candidate $w^* \in H$ is called

(i) global minimizer of f in H, if

$$f(w^*) \leq f(w) \quad \text{for all } w \in H$$

(ii) strict global minimizer of f in H, if

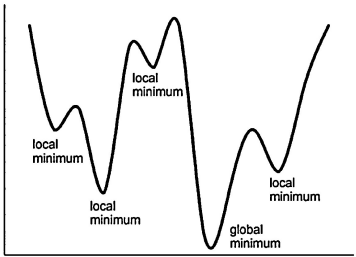
$$f(w^*) < f(w) \quad \text{for all } w \in H \setminus \{w^*\}.$$

(iii) local minimizer of f in H, if there exists a neighborhood of w^* denoted by $N(w^*)$ such that

$$f(w^*) \leq f(w) \quad \text{for all } w \in H \cap N(w^*).$$

(iv) strict local minimizer of f in H, if there exists $N(w^*)$ such that

$$f(w^*) < f(w) \quad \text{for all } w \in (H \cap N(x^*)) \setminus \{w^*\}.$$



Existence of minimizer

Under which conditions has

$$\min_{w \in H} f(w)$$

a solution?

LUND UNIVERSITY

Existence of minimizer

Under which conditions has

$$\min_{w \in H} f(w)$$

a solution?

- If $|H| < \infty$.
- If there exists an $w_0 \in H$ such that the level set

$$\mathcal{L}_f(w_0) := \{w \in H \mid f(w) \leq f(w_0)\}$$

is compact and f lower semi-continuous (Theorem of Weierstraß).

Definition

A set H is compact, if every sequence in H has a convergent subsequence whose limit belongs to H.

LUND UNIVERSITY

Training

- Find hypotheses in $H_{V,E,\sigma}$ which has minimal risk $\rightarrow$ adjust edge weights w.
- Risk $\mathcal{R}_P(w) := \int_{X \times Y} \ell(y, h_w) \, \mathrm{d}P(x, y)$ cannot be directly minimized.
- ERM: $\min_{w \in \mathbb{R}^{|E|}} \{\mathcal{R}_S(w) := \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, h_w(x_i))\}$

LUND UNIVERSITY

Training

- Find hypotheses in $H_{V,E,\sigma}$ which has minimal risk $\rightarrow$ adjust edge weights w.
- Risk $\mathcal{R}_P(w) := \int_{X \times Y} \ell(y, h_w) \, \mathrm{d}P(x, y)$ cannot be directly minimized.
- ERM: $\min_{w \in \mathbb{R}^{|E|}} \{\mathcal{R}_S(w) := \frac{1}{m} \sum_{i=1}^{m} \ell(y_i, h_w(x_i))\}$
- Gradient descent method:

Algorithmus Gradient descent method

Initialize: $w^0 \in \mathbb{R}^{|E|}$
for $j = 0, 1, 2, \ldots$ **do**
  Choose step size s^j
  $w^{j+1} = w^j - s^j \nabla_w \mathcal{R}_S(w^j)$
end for

LUND UNIVERSITY

Stochastic gradient method

- For a lot of training data S, the calculation of the gradient is computationally very intensive, since a gradient has to be calculated for each data item $(x, y) \in S$.

LUND UNIVERSITY

Stochastic gradient method

- For a lot of training data S, the calculation of the gradient is computationally very intensive, since a gradient has to be calculated for each data item $(x, y) \in S$.
- Remedy: Stochastic gradient method (SGD)

Idea of SGD:

- Random (uniformly distributed) selection of $\tilde{m} \ll m$ training data from S
- $w^{j+1} = w^j - s^j \frac{1}{\tilde{m}} \sum_{i=1}^{\tilde{m}} \nabla_w \ell(y_{j_i}, h_{w^j}(x_{j_i}))$ $\tilde{m}$ is called mini-batch size
- Often $\tilde{m} = 1$

Stochastic gradient method (cont.)

Algorithmus Stochastic gradient method

Initialize: $w^0 \in \mathbb{R}^{|E|}$
for $j = 0, 1, 2, \ldots$ **do**
  Choose randomly $(x, y) \in S$
  Set $d^j = -\nabla_w \ell(y, h_{w^j}(x))$
  Choose step-size s^j (e.g. $s^j = \frac{1}{c(j+1)}$, $c > 0$)
  $w^{j+1} = w^j + s^j d^j$
end for

LUND UNIVERSITY

Stochastic gradient method (cont.)

Algorithmus Stochastic gradient method

Initialize: $w^0 \in \mathbb{R}^{|E|}$
for $j = 0, 1, 2, \ldots$ **do**
  Choose randomly $(x, y) \in S$
  Set $d^j = -\nabla_w \ell(y, h_{w^j}(x))$
  Choose step-size s^j (e.g. $s^j = \frac{1}{c(j+1)}$, $c > 0$)
  $w^{j+1} = w^j + s^j d^j$
end for

Note: Let $w = w^j$ and (x_i, y_i) be randomly chosen from S

$$\mathop{\mathbb{E}}_{(x_i,y_i)\sim U}[\nabla_w \ell(y_i, h_w(x_i))] = \sum_{i=1}^{m} \nabla_w \ell(y_i, h_w(x_i)) \cdot \mathbb{P}[Z = (x_i, y_i)] = \frac{1}{m} \sum_{i=1}^{m} \nabla_w \ell(y_i, h_w(x_i))$$
$$= \nabla_w \mathcal{R}_S(w)$$

Descent method

Consider $\min_{w \in \mathbb{R}^n} f(w)$ with $f : \mathbb{R}^n \to \mathbb{R}$ continuously differentiable.

Algorithmus Descent method

Initialize: $w^0 \in R^n$
for $j = 0, 1, 2, \ldots$ **do**
  Compute a descent direction d^j
  Compute a step-size s^j,such that $f(w^j + s^j d^j) < f(w^j)$
  Set $w^{j+1} = w^j + s^j d^j$
end for

LUND UNIVERSITY

Descent direction - Criterion

Definition (Descent direction)

A direction $d \in \mathbb{R}^n \setminus \{0\}$ is called descent direction of the function f in $x \in \mathbb{R}^n$, if there exists an $\bar{s} > 0$ with

$$f(x + sd) < f(x) \qquad \forall s \in (0, \bar{s}).$$

Lemma

Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable in $x \in \mathbb{R}^n$. If $d \in \mathbb{R}^n$ fulfills the inequality

$$\nabla f(x)^T d < 0,$$

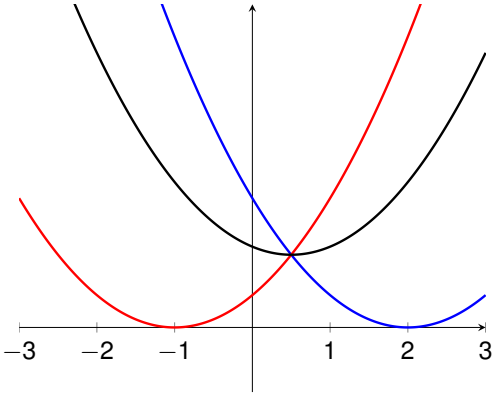
then d is a descent direction of f in x.

LUND UNIVERSITY

SGD is not a descent method

Example:

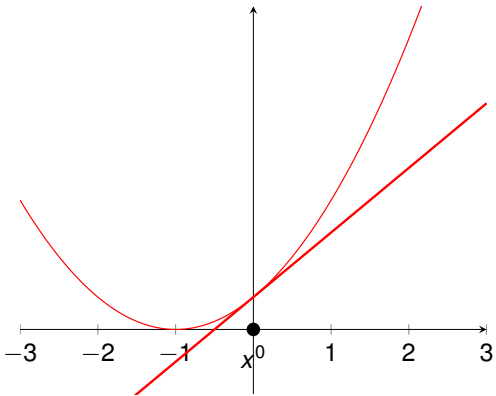
- $f(x) = \frac{1}{2}(f_1(x) + f_2(x))$ with $f_1(x) = \frac{1}{2}(x-2)^2$ and $f_2(x) = \frac{1}{2}(x+1)^2$.



SGD is not a descent method

Example:

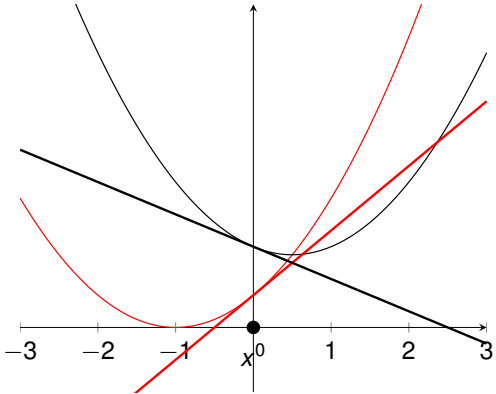
- $f(x) = \frac{1}{2}(f_1(x) + f_2(x))$ with $f_1(x) = \frac{1}{2}(x-2)^2$ and $f_2(x) = \frac{1}{2}(x+1)^2$.
- Set $x^0 = 0$ and randomly choose $i = 2 \Rightarrow d^0 = -\nabla f_2(x^0) = -1$.



SGD is not a descent method

Example:

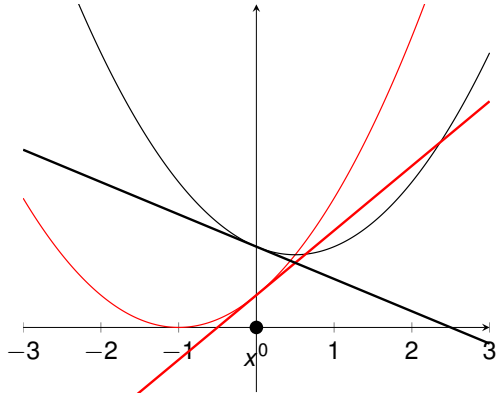
- $f(x) = \frac{1}{2}(f_1(x) + f_2(x))$ with $f_1(x) = \frac{1}{2}(x-2)^2$ and $f_2(x) = \frac{1}{2}(x+1)^2$.
- Set $x^0 = 0$ and randomly choose $i = 2 \Rightarrow d^0 = -\nabla f_2(x^0) = -1$.
- $\nabla f(x) = \frac{1}{2}(2x-1) \Rightarrow \nabla f(x^0) = -\frac{1}{2}$



SGD is not a descent method

Example:

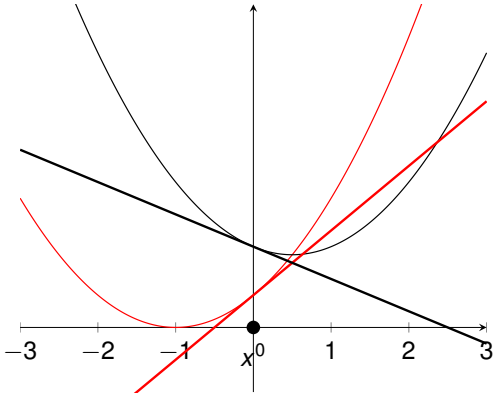
- $f(x) = \frac{1}{2}(f_1(x) + f_2(x))$ with $f_1(x) = \frac{1}{2}(x-2)^2$ and $f_2(x) = \frac{1}{2}(x+1)^2$.
- Set $x^0 = 0$ and randomly choose $i = 2 \Rightarrow d^0 = -\nabla f_2(x^0) = -1$.
- $\nabla f(x) = \frac{1}{2}(2x-1) \Rightarrow \nabla f(x^0) = -\frac{1}{2}$
- $\Rightarrow \nabla f(x^0)d^0 = \frac{1}{2} \not< 0$



SGD is not a descent method

Example:

- $f(x) = \frac{1}{2}(f_1(x) + f_2(x))$ with $f_1(x) = \frac{1}{2}(x-2)^2$ and $f_2(x) = \frac{1}{2}(x+1)^2$.
- Set $x^0 = 0$ and randomly choose $i = 2 \Rightarrow d^0 = -\nabla f_2(x^0) = -1$.
- $\nabla f(x) = \frac{1}{2}(2x-1) \Rightarrow \nabla f(x^0) = -\frac{1}{2}$
- $\Rightarrow \nabla f(x^0) d^0 = \frac{1}{2} \not< 0$



$\Rightarrow$ stochastic gradient method is not a descent method in general.

SGD for FNN

Algorithmus stochastic gradient method for FNN

Parameter: Stopping criteria
step-sizes $(s^1, s^2, \ldots)$
Input: Layered FNN-architecture (V, E)
Differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$
Initialize: Choose $w^0 \in \mathbb{R}^{|E|}$ randomly
for $j = 0, 1, 2, \ldots$ **do**
Choose randomly $(x, y) \in S$
Compute the gradient $g^j = \nabla_w \ell(y, h_{w^j}(x))$ (using backpropagation)
Set $w^{j+1} = w^j - s^j g^j$
end for

LUND UNIVERSITY

Backpropagation for FNN

Notation:

- Loss function $\ell(y_i, h_w(x_i)) =: \ell^i(w) = \frac{1}{2}\sum_{a=1}^{|V_T|}(h_a - y^a)^2$
- For simplicity let $\ell(w) = \ell^i(w)$
- $h_j = o_{T,j}$, $j = 1, \ldots, |V_T|$
- $o_{t,j} = \sigma(a_{t,j})$, $j = 1, \ldots, |V_t|$, $t = 1, \ldots, T$
- $a_{t,j} = \sum_{r=1}^{|V_{t-1}|} w_{rj}^t o_{t-1,r}$, $j = 1, \ldots, |V_t|$, $t = 1, \ldots, T$
- w_{rj}^t is the weight from V_{t-1} to V_t

Derivative of ℓ w.r.t. w_{rj}^t:

$$\frac{\partial \ell}{\partial w_{rj}^t} = \frac{\partial \ell}{\partial a_{t,j}} \frac{\partial a_{t,j}}{\partial w_{rj}^t} = \underbrace{\frac{\partial \ell}{\partial o_{t,j}}}_{=:\delta_{t,j}} \sigma'(a_{t,j}) o_{t-1,r} = \delta_{t,j} \sigma'(a_{t,j}) o_{t-1,r}$$

LUND UNIVERSITY

Computation of $\delta_{t,j}$

- Case 1: $t = T$

$$\frac{\partial \ell}{\partial w_{rj}^T} = \underbrace{\frac{\partial \ell}{\partial o_{T,j}}}_{=\delta_{T,j}} \sigma'(a_{T,j}) o_{T-1,r} = (h_j - y^j)\sigma'(a_{T,j}) o_{T-1,r} = \delta_{T,j}\sigma'(a_{T,j}) o_{T-1,r}$$

- Case 2: $t = T - 1$

$$\frac{\partial \ell}{\partial w_{rj}^{T-1}} = \frac{\partial \ell}{\partial o_{T-1,j}} \frac{\partial o_{T-1,j}}{\partial w_{rj}^{T-1}} = \sum_{i=1}^{|V_T|} \frac{\partial \ell}{\partial a_{T,i}} \frac{\partial a_{T,i}}{\partial o_{T-1,j}} \frac{\partial o_{T-1,j}}{\partial w_{rj}^{T-1}}$$

$$= \underbrace{\sum_{i=1}^{|V_T|} \delta_{T,i}\sigma'(a_{T,i}) w_{ji}^T}_{\delta_{T-1,j}} \sigma'(a_{T-1,j}) o_{T-2,r}$$

$$\Rightarrow \delta_{t,j} = \begin{cases} h_j - y^j & \text{if } t = T \\ \sum_{i=1}^{|V_{t+1}|} \delta_{t+1,i}\sigma'(a_{t+1,i}) w_{ji}^{t+1} & \text{else} \end{cases}$$

LUND UNIVERSITY

Backpropagation as algorithm

1. **Input:** Input of x, determine the output of the input layer.
2. **Feedforward:** Compute a_{t_j} and $o_{t,j}$ for all t, j.
3. **Output Error:** Compute $\delta_{T,j}$
4. **Backpropagate the Error:** Determine recursively $\delta_{t,j}$ for $t = T-1, \ldots, 1$
5. **Output:** $\frac{\partial \ell}{\partial w_{rj}^t} = \delta_{t,j}\sigma'(a_{t,j})o_{t-1,r}$ for all r, j, t.

LUND UNIVERSITY

Formulation of an attack

Given:

- Neural network (V, E, σ, w) and hence $h_w : X(= \mathbb{R}^n) \to Y(= \mathbb{R}^{|V_T|})$
- Loss function $\ell : Y \times Y \to \mathbb{R}$
- $x \in X = \mathbb{R}^n$ (can also be from $S_{|x}$) with $h_w(x) = y \in Y$
- Target classification $\tilde{y} \in Y$ (not always used)

LUND UNIVERSITY

Formulation of an attack

Given:

- Neural network (V, E, σ, w) and hence $h_w : X(= \mathbb{R}^n) \to Y(= \mathbb{R}^{|V_T|})$
- Loss function $\ell : Y \times Y \to \mathbb{R}$
- $x \in X = \mathbb{R}^n$ (can also be from $S_{|x}$) with $h_w(x) = y \in Y$
- Target classification $\tilde{y} \in Y$ (not always used)

Formulation of the attack problem: Solve

$$\inf_{\tilde{x} \in X} \|x - \tilde{x}\| \qquad \text{subject to} \quad h_w(\tilde{x}) = \tilde{y}$$

- This approach minimizes the change (disturbance) while the constraint requires the classification $\tilde{y}$.
- If $h_w(x) = \tilde{y}$, then the minimization problem is trivial, as the solution is $\tilde{x}^* = x$.

LUND UNIVERSITY

(A) The probably first attack

For simplicity let $\mathfrak{L}(x, y) := \ell(y, h_w(x))$.

Approach of [Szegedy, Zaremba, Sutskever, Bruna, Erhan, Goodfellow, Fergus 2014]

- Formulate the following unrestricted optimization problem

$$\min_{\tilde{x} \in X} \alpha \|x - \tilde{x}\|_2 + \mathfrak{L}(\tilde{x}, \tilde{y})$$

  with $\alpha > 0$.
- Solution method: L-BFGS

LUND UNIVERSITY

(B) Gradient methods

- Approach of [Goodfellow, Shlens, Szegedy 2015]
  - Linearization: $\mathfrak{L}(x+\delta, y) \approx \mathfrak{L}(x, y) + \delta^T \nabla_x \mathfrak{L}(x, y)$
  - We want to choose $\delta \in \mathbb{R}^n$ such that $\delta^T \nabla_x \mathfrak{L}(x, y)$ is maximal, where $\|\delta\|_\infty \leq \epsilon$ is sufficiently small.
  - Choose $\delta = \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y)) \Rightarrow \delta^T \nabla_x \mathfrak{L}(x, y) = \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y))^T \nabla_x \mathfrak{L}(x, y) \geq 0$
  - $\epsilon > 0$ specifies the size of the change (step size)
  - $\tilde{x} = x + \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y))$
  - δ is not a descent direction. BUT $-\delta$ is a descent direction!

LUND UNIVERSITY

(B) Gradient methods

(i) Approach of [Goodfellow, Shlens, Szegedy 2015]
- Linearization: $\mathfrak{L}(x+\delta, y) \approx \mathfrak{L}(x, y) + \delta^T \nabla_x \mathfrak{L}(x, y)$
- We want to choose $\delta \in \mathbb{R}^n$ such that $\delta^T \nabla_x \mathfrak{L}(x, y)$ is maximal, where $\|\delta\|_\infty \leq \epsilon$ is sufficiently small.
- Choose $\delta = \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y)) \Rightarrow \delta^T \nabla_x \mathfrak{L}(x, y) = \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y))^T \nabla_x \mathfrak{L}(x, y) \geq 0$
- $\epsilon > 0$ specifies the size of the change (step size)
- $\tilde{x} = x + \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y))$
- δ is not a descent direction. BUT $-\delta$ is a descent direction!

(ii) [Rozsa, Rudd, Boult 2016]: $\tilde{x} = x + \epsilon \nabla_x \mathfrak{L}(x, y)$

LUND UNIVERSITY

(B) Gradient methods

(i) Approach of [Goodfellow, Shlens, Szegedy 2015]
- Linearization: $\mathfrak{L}(x+\delta, y) \approx \mathfrak{L}(x, y) + \delta^T \nabla_x \mathfrak{L}(x, y)$
- We want to choose $\delta \in \mathbb{R}^n$ such that $\delta^T \nabla_x \mathfrak{L}(x, y)$ is maximal, where $\|\delta\|_\infty \leq \epsilon$ is sufficiently small.
- Choose $\delta = \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y)) \Rightarrow \delta^T \nabla_x \mathfrak{L}(x, y) = \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y))^T \nabla_x \mathfrak{L}(x, y) \geq 0$
- $\epsilon > 0$ specifies the size of the change (step size)
- $\tilde{x} = x + \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y))$
- δ is not a descent direction. BUT $-\delta$ is a descent direction!

(ii) [Rozsa, Rudd, Boult 2016]: $\tilde{x} = x + \epsilon \nabla_x \mathfrak{L}(x, y)$

(iii) [Dong et al. 2017]: $x^{k+1} = x^k + \epsilon \operatorname{sign} g^{k+1}$ mit $g^{k+1} = \mu g^k + \frac{\nabla_x \mathfrak{L}(x^k, y)}{\|\nabla_x \mathfrak{L}(x^k, y)\|}$

LUND UNIVERSITY

(B) Gradient methods

(i) Approach of [Goodfellow, Shlens, Szegedy 2015]
- Linearization: $\mathfrak{L}(x+\delta, y) \approx \mathfrak{L}(x, y) + \delta^T \nabla_x \mathfrak{L}(x, y)$
- We want to choose $\delta \in \mathbb{R}^n$ such that $\delta^T \nabla_x \mathfrak{L}(x, y)$ is maximal, where $\|\delta\|_\infty \leq \epsilon$ is sufficiently small.
- Choose $\delta = \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y)) \Rightarrow \delta^T \nabla_x \mathfrak{L}(x, y) = \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y))^T \nabla_x \mathfrak{L}(x, y) \geq 0$
- $\epsilon > 0$ specifies the size of the change (step size)
- $\tilde{x} = x + \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y))$
- δ is not a descent direction. BUT $-\delta$ is a descent direction!

(ii) [Rozsa, Rudd, Boult 2016]: $\tilde{x} = x + \epsilon \nabla_x \mathfrak{L}(x, y)$

(iii) [Dong et al. 2017]: $x^{k+1} = x^k + \epsilon \operatorname{sign} g^{k+1}$ mit $g^{k+1} = \mu g^k + \frac{\nabla_x \mathfrak{L}(x^k, y)}{\|\nabla_x \mathfrak{L}(x^k, y)\|}$

Critic: It could be that $\nabla_x \mathfrak{L}(x, y) = 0$.

LUND UNIVERSITY

(B) Gradient methods

(i) Approach of [Goodfellow, Shlens, Szegedy 2015]
- Linearization: $\mathfrak{L}(x+\delta, y) \approx \mathfrak{L}(x, y) + \delta^T \nabla_x \mathfrak{L}(x, y)$
- We want to choose $\delta \in \mathbb{R}^n$ such that $\delta^T \nabla_x \mathfrak{L}(x, y)$ is maximal, where $\|\delta\|_\infty \leq \epsilon$ is sufficiently small.
- Choose $\delta = \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y)) \Rightarrow \delta^T \nabla_x \mathfrak{L}(x, y) = \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y))^T \nabla_x \mathfrak{L}(x, y) \geq 0$
- $\epsilon > 0$ specifies the size of the change (step size)
- $\tilde{x} = x + \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, y))$
- δ is not a descent direction. BUT $-\delta$ is a descent direction!

(ii) [Rozsa, Rudd, Boult 2016]: $\tilde{x} = x + \epsilon \nabla_x \mathfrak{L}(x, y)$

(iii) [Dong et al. 2017]: $x^{k+1} = x^k + \epsilon \operatorname{sign} g^{k+1}$ mit $g^{k+1} = \mu g^k + \frac{\nabla_x \mathfrak{L}(x^k, y)}{\|\nabla_x \mathfrak{L}(x^k, y)\|}$

Critic: It could be that $\nabla_x \mathfrak{L}(x, y) = 0$.

(iv) Use target classification $\tilde{y}$ [Kurakin, Goodfellow, Bengio 2017]

$$\tilde{x} = x - \epsilon \operatorname{sign}(\nabla_x \mathfrak{L}(x, \tilde{y}))$$

Can also be extended iteratively.

LUND UNIVERSITY

(C) DeepFool [Moosavi-Dezfooli, Fawzi, Froosard 2016]

Assumptions:

- Consider binary classification, i.e. $Y = \{-1, +1\}$.
- $h_w(x) = \sigma_{\text{sign}}(f(x))$
- $f : \mathbb{R}^n \to \mathbb{R}$ an affine classifier $f(x) = \langle w, x \rangle + b$, $w \in \mathbb{R}^n$, $b \in \mathbb{R}$
- $F := \{x \in \mathbb{R}^n \mid f(x) = 0\}$ is the separating affine hyperplane.

LUND UNIVERSITY

(C) DeepFool [Moosavi-Dezfooli, Fawzi, Froosard 2016]

Assumptions:

- Consider binary classification, i.e. $Y = \{-1, +1\}$.
- $h_w(x) = \sigma_{\text{sign}}(f(x))$
- $f : \mathbb{R}^n \to \mathbb{R}$ an affine classifier $f(x) = \langle w, x \rangle + b$, $w \in \mathbb{R}^n$, $b \in \mathbb{R}$
- $F := \{x \in \mathbb{R}^n \mid f(x) = 0\}$ is the separating affine hyperplane.

Goal:

- Find the minimum change to alter the classifier's decision.

LUND UNIVERSITY

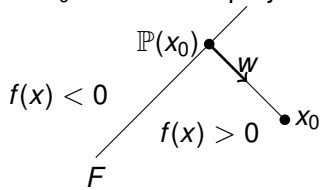
(C) DeepFool [Moosavi-Dezfooli, Fawzi, Froosard 2016]

Assumptions:

- Consider binary classification, i.e. $Y = \{-1, +1\}$.
- $h_w(x) = \sigma_{\text{sign}}(f(x))$
- $f : \mathbb{R}^n \to \mathbb{R}$ an affine classifier $f(x) = \langle w, x \rangle + b$, $w \in \mathbb{R}^n$, $b \in \mathbb{R}$
- $F := \{x \in \mathbb{R}^n \mid f(x) = 0\}$ is the separating affine hyperplane.

Goal:

- Find the minimum change to alter the classifier's decision.
- Let $x_0 \in \mathbb{R}^n$. Then project x_0 onto F:



$$\Rightarrow \tilde{x} = \mathbb{P}(x_0) = x_0 - \frac{f(x_0)}{\|w\|_2^2} w$$

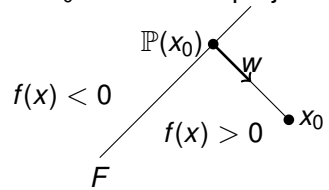
(C) DeepFool [Moosavi-Dezfooli, Fawzi, Froosard 2016]

Assumptions:

- Consider binary classification, i.e. $Y = \{-1, +1\}$.
- $h_w(x) = \sigma_{\text{sign}}(f(x))$
- $f : \mathbb{R}^n \to \mathbb{R}$ an affine classifier $f(x) = \langle w, x \rangle + b$, $w \in \mathbb{R}^n$, $b \in \mathbb{R}$
- $F := \{x \in \mathbb{R}^n \mid f(x) = 0\}$ is the separating affine hyperplane.

Goal:

- Find the minimum change to alter the classifier's decision.
- Let $x_0 \in \mathbb{R}^n$. Then project x_0 onto F:



$$\Rightarrow \tilde{x} = \mathbb{P}(x_0) = x_0 - \frac{f(x_0)}{\|w\|_2^2} w$$

- Extension to nonlinear functions f and multiclass problems

Questions