Moderne Methoden der Datenanalyse
– Ereignisklassifikation –

Roger Wolf

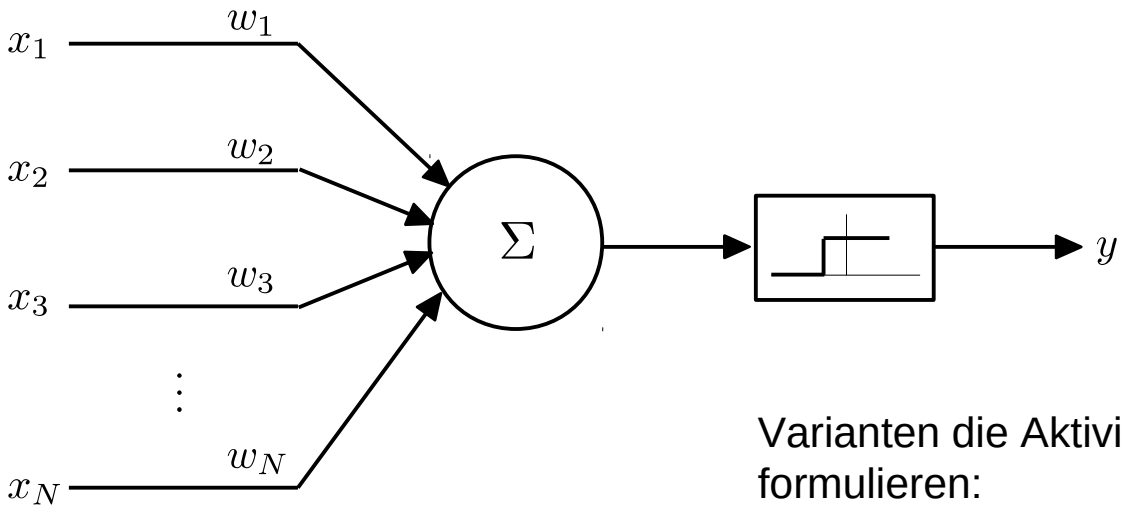
09. Juli 2020

Inhalt der Vorlesung

- MLP Begriffe und Definitionen.
- Eine vs. mehrere versteckte Lagen und die Bedeutung von Tiefe.
- Bedeutung der Aktivierungsfunktion.
- MLP training:
  - „Perceptron learning rule“ und Limitierungen.
  - Bedeutung der Aktivierungsfunktion.

Recap Perceptron

- In der letzten Vorlesung haben wir das boolsche Perceptron mit einer einfachen Stufenfunktion diskutiert:



Inputs Weights

$x_1 \dots x_N \in \mathbb{R}$

$w_1 \dots w_N \in \mathbb{R}$

Einheit feuert, wenn $\sum_i w_i x_i \geq T$

Varianten die Aktivierungslogik zu formulieren:

$$\sum_i w_i x_i \geq T \,,$$

$$\sum_i w_i x_i - T \geq 0 \,,$$

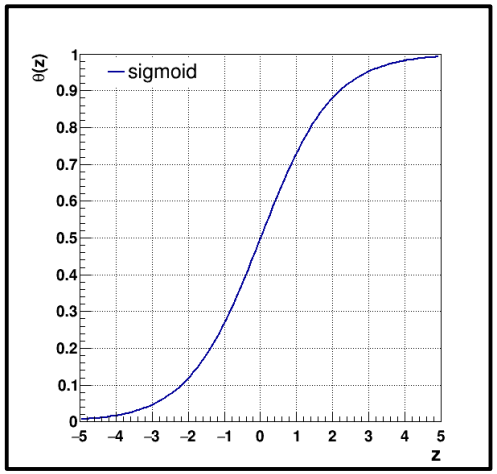
$$\theta\left(\sum_i w_i x_i - T\right)$$

Aktivierungsfunktionen

- In der heutigen Vorlesung werden wir die Bedeutung kontinuierlicher Aktivierungsfunktionen diskutieren. Beispiele hierfür sind:

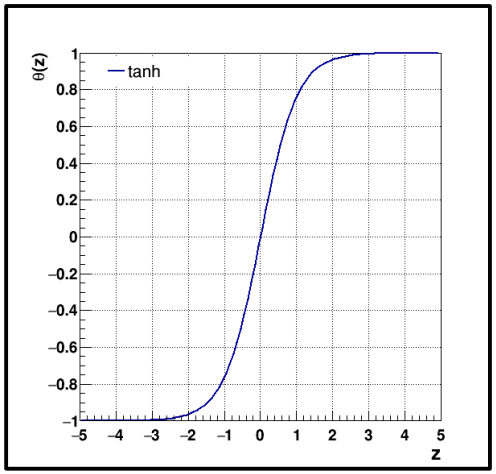
Sigmoid:

$$\theta(z) = \frac{1}{1+\exp(-z)}$$



tanh:

$$\theta(z) = \tanh(z)$$

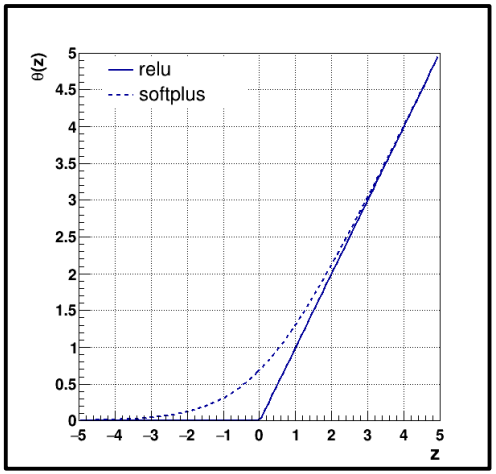


Rectified linear unit:

$$\theta(z) = \begin{cases} z & z \geq 0 \\ 0 & \text{sonst} \end{cases}$$

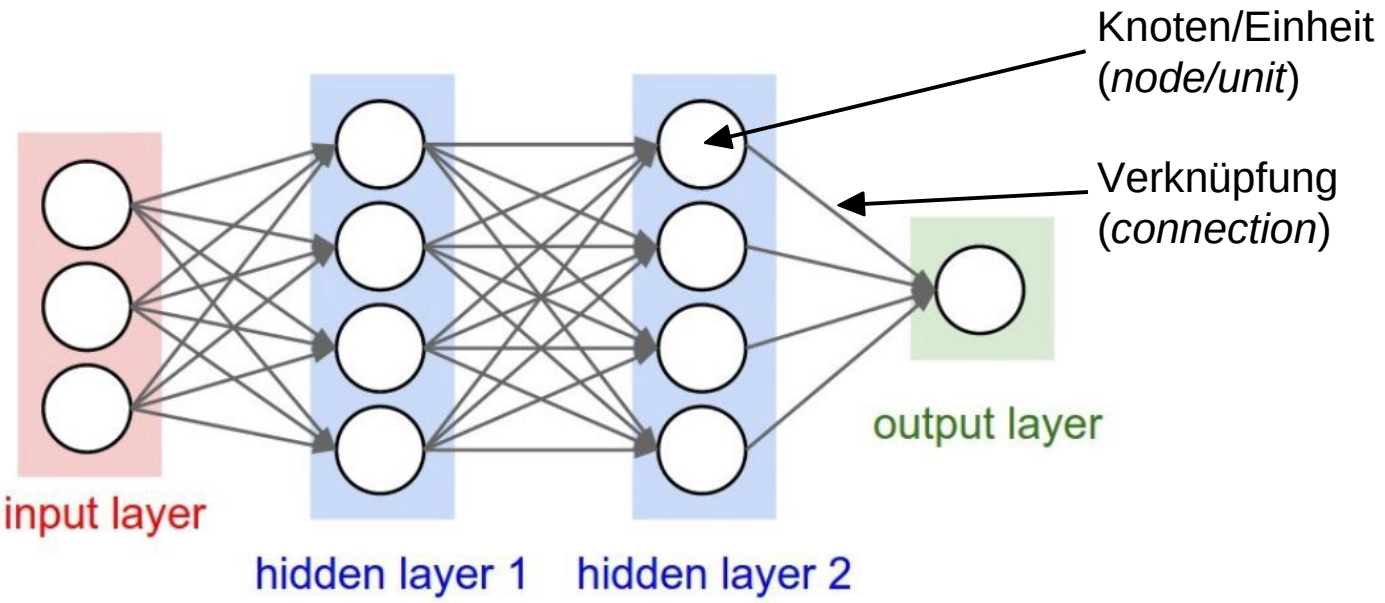
Softplus:

$$\theta(z) = \log(1+\exp(z))$$



Multilayer Perceptron – Begriffe

- Wir klären hier nochmal die Begrifflichkeiten des Multilayer Perceptrons (MLP):

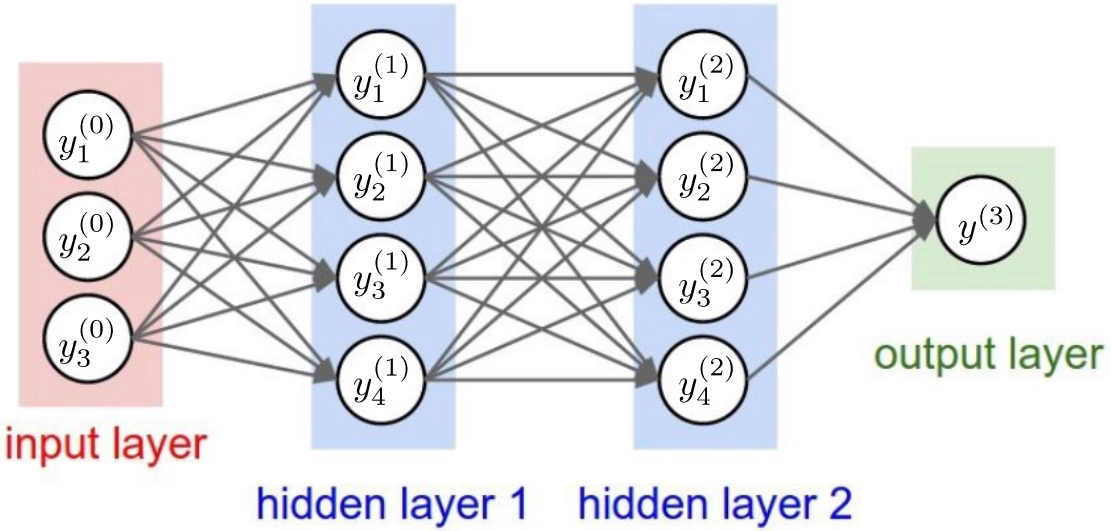


- Die *input* Lage werden wir im folgenden nicht explizit mitzählen, weil sie keine Perceptrons enthält.

- Ein MLP bezeichnen wir auch als (neuronales) Netzwerk (NN).

Multilayer Perceptron – Definitionen

- Für den output des MLP definieren wir die folgenden Größen/Variablen:



$$y_j^{(0)} = x_j$$

$$y_j^{(1)} = \theta_1 \left(\sum_i w_i y_i^{(0)} \right)$$

$$y_j^{(2)} = \theta_2 \left(\sum_i w_i \theta_1 \left(\sum_i w_i y_i^{(0)} \right) \right)$$

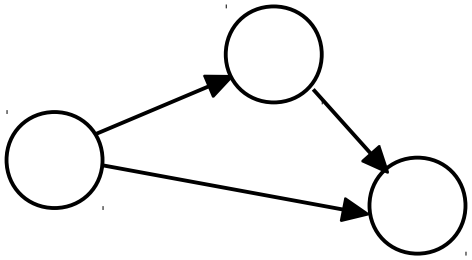
$$\vdots$$

$$y_j^{(k)} = \theta_k \left(z_j^{(k)} \right) \quad z_j^{(k)} = \sum_i w_i y_i^{(k-1)}$$

Tiefe

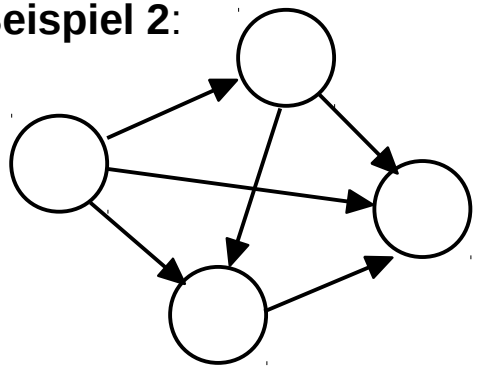
- In dieser Vorlesung werden wir nur strikte feed forward Netzwerke diskutieren.
- In einem solchen Zusammenhang kann man das Netzwerk als gerichteten Graph mit einer Tiefe d verstehen.
- Ein gerichteter Graph hat Quellen und Senken. Die Tiefe eines Graphen ist der längste Pfad von einer Quelle zu einer Senke.

- **Beispiel 1**:



Tiefe? –

- **Beispiel 2**:



Tiefe? –

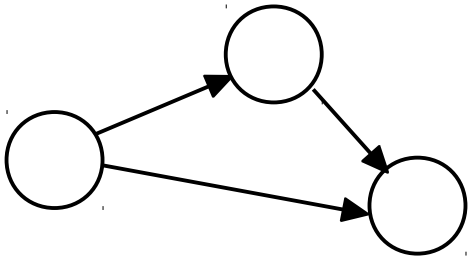
Tiefe

- In dieser Vorlesung werden wir nur strikte feed forward Netzwerke diskutieren.

- In einem solchen Zusammenhang kann man das Netzwerk als gerichteten Graph mit einer Tiefe d verstehen.

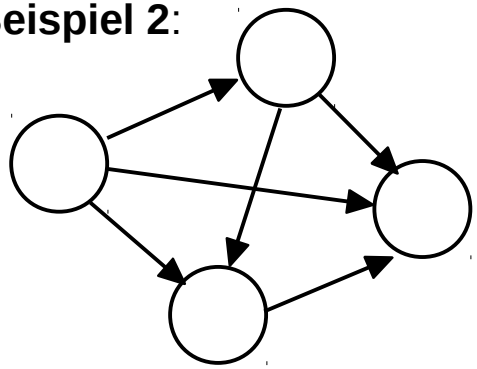
- Ein gerichteter Graph hat Quellen und Senken. Die Tiefe eines Graphen ist der längste Pfad von einer Quelle zu einer Senke.

- **Beispiel 1**:



Tiefe? – 2

- **Beispiel 2**:



Tiefe? – 3

- Ein Netzwerk mit einer Tiefe von $d \geq 2$ bezeichnet man als tief (*deep*).

MLP als Repäsentation boolscher Funktionen

- Wir haben in der letzten Vorlesung gesehen, dass ein MLP beliebige boolsche Funktionen abbilden kann. **Frage**: Wieviele Lagen benötigt das MLP dafür? –

MLP als Repäsentation boolscher Funktionen

- Wir haben in der letzten Vorlesung gesehen, dass ein MLP beliebige boolsche Funktionen abbilden kann. **Frage**: Wieviele Lagen benötigt das MLP dafür? – **2** (ohne input layer).

MLP als Repäsentation boolscher Funktionen

- Wir haben in der letzten Vorlesung gesehen, dass ein MLP beliebige boolsche Funktionen abbilden kann. **Frage**: Wieviele Lagen benötigt das MLP dafür? – **2** (ohne input layer).

- **Beweis**: jede boolsche Funktion läßt sich mit Hilfe einer Wahrheitstabelle darstellen.

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Beliebiges Beispiel

Disjunkte Normalform (DNF):

$$y = \underbrace{\overline{x}_1\overline{x}_2x_3x_4\overline{x}_5} \wedge \underbrace{\overline{x}_1x_2\overline{x}_3x_4x_5} \wedge \underbrace{\overline{x}_1x_2x_3\overline{x}_4\overline{x}_5} \wedge$$

$$\underbrace{x_1\overline{x}_2\overline{x}_3\overline{x}_4x_5} \wedge \underbrace{x_1\overline{x}_2x_3x_4x_5} \wedge \underbrace{x_1x_2\overline{x}_3\overline{x}_4x_5}$$

MLP als Repäsentation boolscher Funktionen

- Wir haben in der letzten Vorlesung gesehen, dass ein MLP beliebige boolsche Funktionen abbilden kann. **Frage**: Wieviele Lagen benötigt das MLP dafür? – **2** (ohne input layer).

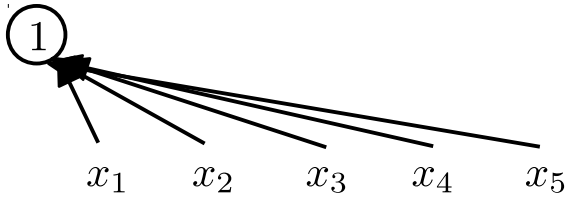
- **Beweis**: jede boolsche Funktion läßt sich mit Hilfe einer Wahrheitstabelle darstellen.

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Beliebiges Beispiel

Disjunkte Normalform (DNF):

$$y = \underbrace{\overline{x}_1\overline{x}_2x_3x_4\overline{x}_5}_{1} \wedge \underbrace{\overline{x}_1x_2\overline{x}_3x_4x_5}_{1} \wedge \underbrace{\overline{x}_1x_2x_3\overline{x}_4\overline{x}_5}_{1} \wedge$$
$$\underbrace{x_1\overline{x}_2\overline{x}_3\overline{x}_4x_5}_{1} \wedge \underbrace{x_1\overline{x}_2x_3x_4x_5}_{1} \wedge \underbrace{x_1x_2\overline{x}_3\overline{x}_4x_5}_{1}$$



MLP als Repäsentation boolscher Funktionen

- Wir haben in der letzten Vorlesung gesehen, dass ein MLP beliebige boolsche Funktionen abbilden kann. **Frage**: Wieviele Lagen benötigt das MLP dafür? – **2** (ohne input layer).

- **Beweis**: jede boolsche Funktion läßt sich mit Hilfe einer Wahrheitstabelle darstellen.

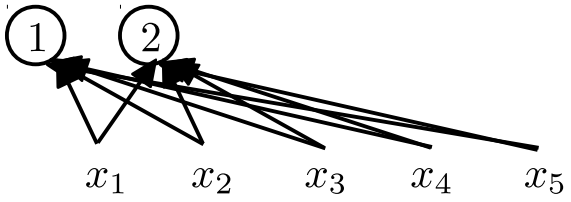
x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Beliebiges Beispiel

Disjunkte Normalform (DNF):

$$y = \underbrace{\overline{x}_1\overline{x}_2x_3x_4\overline{x}_5}_{1} \wedge \underbrace{\overline{x}_1x_2\overline{x}_3x_4x_5}_{2} \wedge \underbrace{\overline{x}_1x_2x_3\overline{x}_4\overline{x}_5}_{} \wedge$$

$$\underbrace{x_1\overline{x}_2\overline{x}_3\overline{x}_4x_5}_{} \wedge \underbrace{x_1\overline{x}_2x_3x_4x_5}_{} \wedge \underbrace{x_1x_2\overline{x}_3\overline{x}_4x_5}_{}$$



MLP als Repäsentation boolscher Funktionen

- Wir haben in der letzten Vorlesung gesehen, dass ein MLP beliebige boolsche Funktionen abbilden kann. **Frage**: Wieviele Lagen benötigt das MLP dafür? – **2** (ohne input layer).

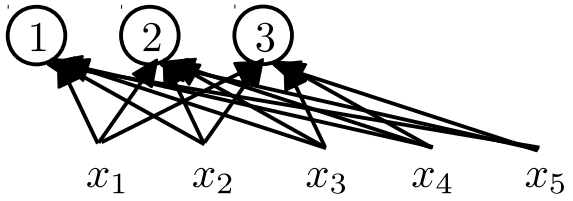
- **Beweis**: jede boolsche Funktion läßt sich mit Hilfe einer Wahrheitstabelle darstellen.

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Beliebiges Beispiel

Disjunkte Normalform (DNF):

$$y = \underbrace{\overline{x}_1\overline{x}_2x_3x_4\overline{x}_5}_{1} \wedge \underbrace{\overline{x}_1x_2\overline{x}_3x_4x_5}_{2} \wedge \underbrace{\overline{x}_1x_2x_3\overline{x}_4\overline{x}_5}_{3} \wedge$$
$$\underbrace{x_1\overline{x}_2\overline{x}_3\overline{x}_4x_5} \wedge \underbrace{x_1\overline{x}_2x_3x_4x_5} \wedge \underbrace{x_1x_2\overline{x}_3\overline{x}_4x_5}$$



MLP als Repäsentation boolscher Funktionen

- Wir haben in der letzten Vorlesung gesehen, dass ein MLP beliebige boolsche Funktionen abbilden kann. **Frage**: Wieviele Lagen benötigt das MLP dafür? – **2** (ohne input layer).

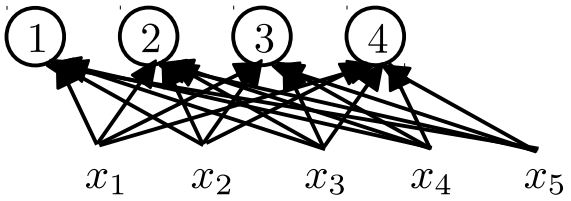
- **Beweis**: jede boolsche Funktion läßt sich mit Hilfe einer Wahrheitstabelle darstellen.

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Beliebiges Beispiel

Disjunkte Normalform (DNF):

$$y = \underbrace{\overline{x}_1\overline{x}_2x_3x_4\overline{x}_5}_{1} \wedge \underbrace{\overline{x}_1x_2\overline{x}_3x_4x_5}_{2} \wedge \underbrace{\overline{x}_1x_2x_3\overline{x}_4\overline{x}_5}_{3} \wedge$$
$$\underbrace{x_1\overline{x}_2\overline{x}_3\overline{x}_4x_5}_{4} \wedge \underbrace{x_1\overline{x}_2x_3x_4x_5}_{} \wedge \underbrace{x_1x_2\overline{x}_3\overline{x}_4x_5}_{}$$



MLP als Repäsentation boolscher Funktionen

- Wir haben in der letzten Vorlesung gesehen, dass ein MLP beliebige boolsche Funktionen abbilden kann. **Frage**: Wieviele Lagen benötigt das MLP dafür? – **2** (ohne input layer).

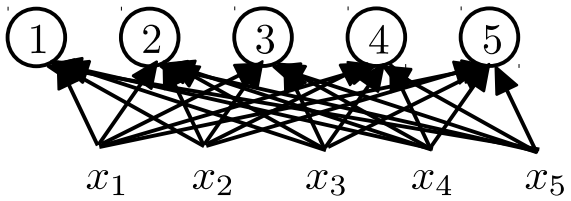
- **Beweis**: jede boolsche Funktion läßt sich mit Hilfe einer Wahrheitstabelle darstellen.

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Beliebiges Beispiel

Disjunkte Normalform (DNF):

$$y = \underbrace{\overline{x}_1\overline{x}_2x_3x_4\overline{x}_5}_{1} \wedge \underbrace{\overline{x}_1x_2\overline{x}_3x_4x_5}_{2} \wedge \underbrace{\overline{x}_1x_2x_3\overline{x}_4\overline{x}_5}_{3} \wedge$$
$$\underbrace{x_1\overline{x}_2\overline{x}_3\overline{x}_4x_5}_{4} \wedge \underbrace{x_1\overline{x}_2x_3x_4x_5}_{5} \wedge \underbrace{x_1x_2\overline{x}_3\overline{x}_4x_5}$$



MLP als Repäsentation boolscher Funktionen

- Wir haben in der letzten Vorlesung gesehen, dass ein MLP beliebige boolsche Funktionen abbilden kann. **Frage**: Wieviele Lagen benötigt das MLP dafür? – **2** (ohne input layer).

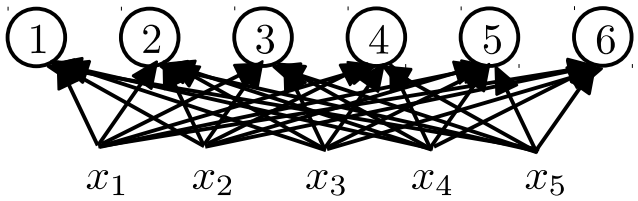
- **Beweis**: jede boolsche Funktion läßt sich mit Hilfe einer Wahrheitstabelle darstellen.

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Beliebiges Beispiel

Disjunkte Normalform (DNF):

$$y = \underbrace{\overline{x}_1\overline{x}_2x_3x_4\overline{x}_5}_{1} \wedge \underbrace{\overline{x}_1x_2\overline{x}_3x_4x_5}_{2} \wedge \underbrace{\overline{x}_1x_2x_3\overline{x}_4\overline{x}_5}_{3} \wedge$$
$$\underbrace{x_1\overline{x}_2\overline{x}_3\overline{x}_4x_5}_{4} \wedge \underbrace{x_1\overline{x}_2x_3x_4x_5}_{5} \wedge \underbrace{x_1x_2\overline{x}_3\overline{x}_4x_5}_{6}$$



MLP als Repäsentation boolscher Funktionen

- Wir haben in der letzten Vorlesung gesehen, dass ein MLP beliebige boolsche Funktionen abbilden kann. **Frage**: Wieviele Lagen benötigt das MLP dafür? – **2** (ohne input layer).

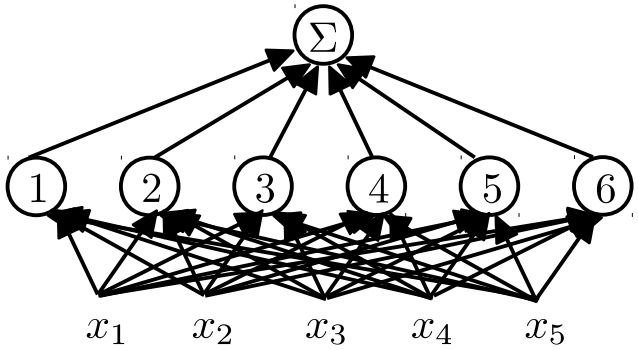
- **Beweis**: jede boolsche Funktion läßt sich mit Hilfe einer Wahrheitstabelle darstellen.

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Beliebiges Beispiel

Disjunkte Normalform (DNF):

$$y = \underbrace{\overline{x}_1\overline{x}_2x_3x_4\overline{x}_5}_{1} \wedge \underbrace{\overline{x}_1x_2\overline{x}_3x_4x_5}_{2} \wedge \underbrace{\overline{x}_1x_2x_3\overline{x}_4\overline{x}_5}_{3} \wedge$$
$$\underbrace{x_1\overline{x}_2\overline{x}_3\overline{x}_4x_5}_{4} \wedge \underbrace{x_1\overline{x}_2x_3x_4x_5}_{5} \wedge \underbrace{x_1x_2\overline{x}_3\overline{x}_4x_5}_{6}$$



MLP als Repäsentation boolscher Funktionen

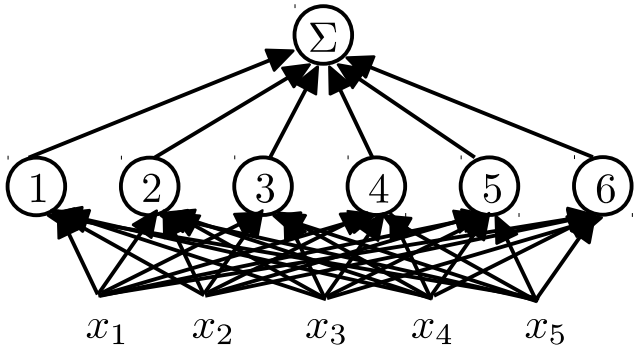
Ohne Beweis: für N inputs kann die benötigte Anzahl an Knoten in der versteckten Lage exponentiell groß werden (exp(N)). Es ist jedoch möglich die Anzahl der Knoten signifikant (bis zu $\log_2(N)$) zu reduzieren, wenn man dem Netzwerk mehr Tiefe erlaubt!

x_1	x_2	x_3	x_4	x_5	y
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1

Beliebiges Beispiel

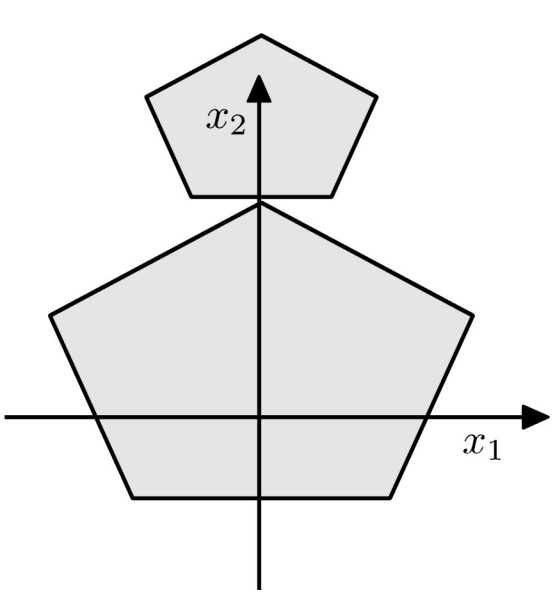
Disjunkte Normalform (DNF):

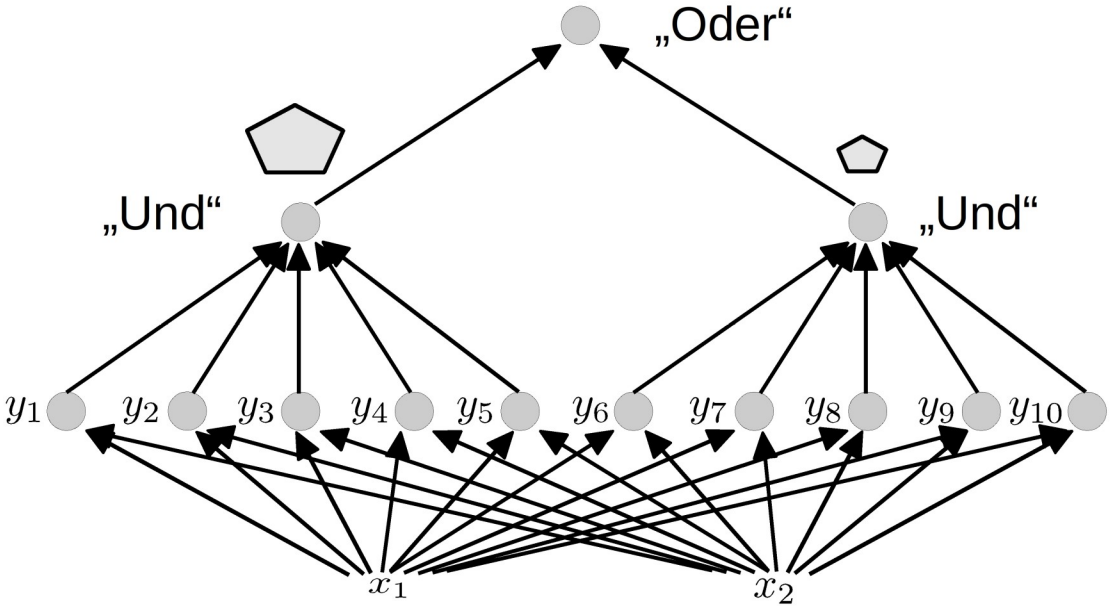
$$y = \underbrace{\overline{x}_1\overline{x}_2x_3x_4\overline{x}_5}_{1} \wedge \underbrace{\overline{x}_1x_2\overline{x}_3x_4x_5}_{2} \wedge \underbrace{\overline{x}_1x_2x_3\overline{x}_4\overline{x}_5}_{3} \wedge$$
$$\underbrace{x_1\overline{x}_2\overline{x}_3\overline{x}_4x_5}_{4} \wedge \underbrace{x_1\overline{x}_2x_3x_4x_5}_{5} \wedge \underbrace{x_1x_2\overline{x}_3\overline{x}_4x_5}_{6}$$



Klassifikation mit Hilfe von MLPs

- Wir haben in der letzten Vorlesung gesehen, dass ein MLP beliebige Konturen, wie die unten links gezeigte beliebig gut approximieren kann.





- Hierzu haben wir zwei versteckte insgesamt also drei Lagen verwendet.

- Glauben Sie, dass dies auch mit *einer einzigen* versteckten Lage möglich ist? Wenn nein warum nicht? Wenn ja wie?

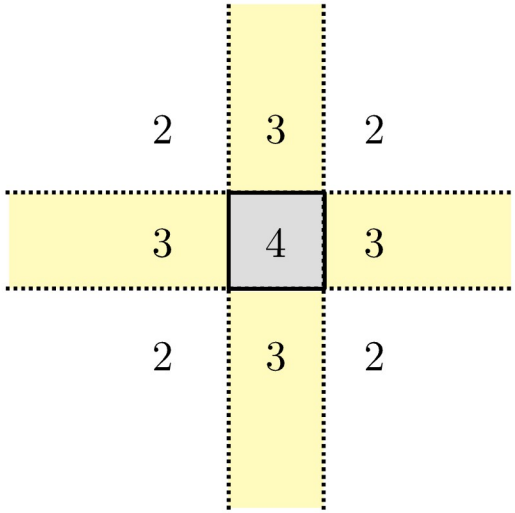
Klassifikation mit Hilfe von MLPs

- **Antwort**: es ist möglich, aber eine allgemeine Konstruktion ist schwieriger, als Sie vielleicht denken mögen.

- Das liegt an der Schwierigkeit mit dem MLP eine beliebige konvexe Kontur mit endlicher Fläche zu bilden:

- Ein **Viereck** kann mit Hilfe eines MLP mit einer versteckten Lage und 4 Knoten, wie auf Folie 11 gezeigt dargestellt werden.

- Wie groß ist das Verhältnis der abgedeckten Flächen für $T \geq N$ und $T \geq N-1$?



Kontur aus 4 Knoten

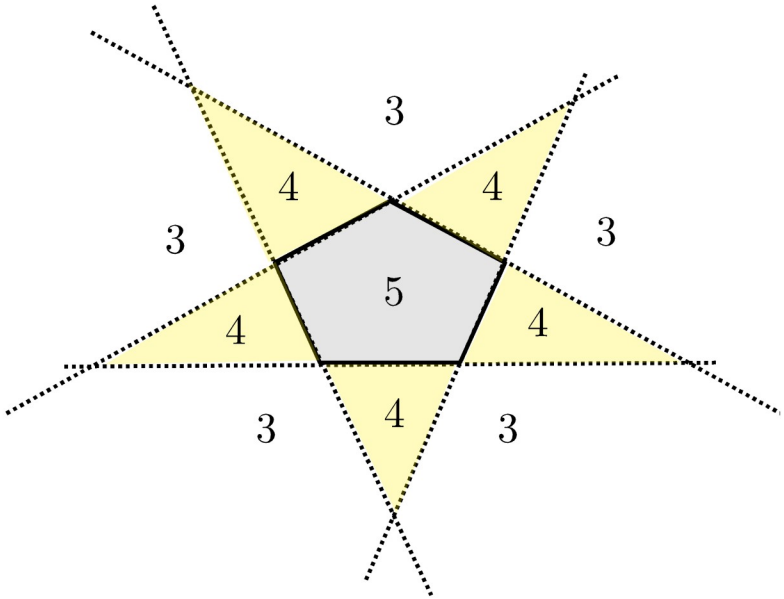
Klassifikation mit Hilfe von MLPs

- **Antwort**: es ist möglich, aber eine allgemeine Konstruktion ist schwieriger, als Sie vielleicht denken mögen.

- Das liegt an der Schwierigkeit mit dem MLP eine beliebige konvexe Kontur mit endlicher Fläche zu bilden:

- Ein **Fünfeck** kann mit Hilfe eines MLP mit einer versteckten Lage und 5 Knoten, wie auf Folie 11 gezeigt dargestellt werden.

- Wie groß ist das Verhältnis der abgedeckten Flächen für $T \geq N$ und $T \geq N-1$?



Kontur aus 5 Knoten

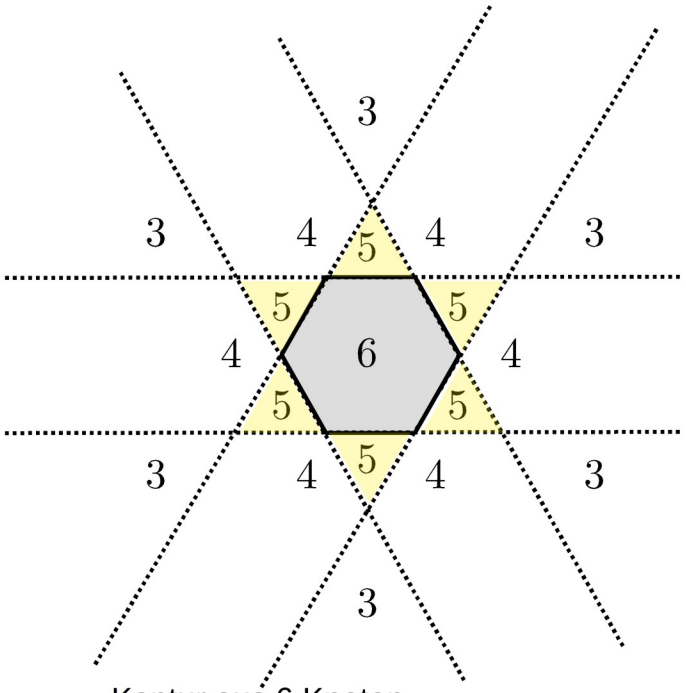
Klassifikation mit Hilfe von MLPs

- **Antwort**: es ist möglich, aber eine allgemeine Konstruktion ist schwieriger, als Sie vielleicht denken mögen.

- Das liegt an der Schwierigkeit mit dem MLP eine beliebige konvexe Kontur mit endlicher Fläche zu bilden:

- Ein **Sechseck** kann mit Hilfe eines MLP mit einer versteckten Lage und 6 Knoten, wie auf Folie 11 gezeigt dargestellt werden.

- Wie groß ist das Verhältnis der abgedeckten Flächen für $T \geq N$ und $T \geq N-1$?



Kontur aus 6 Knoten

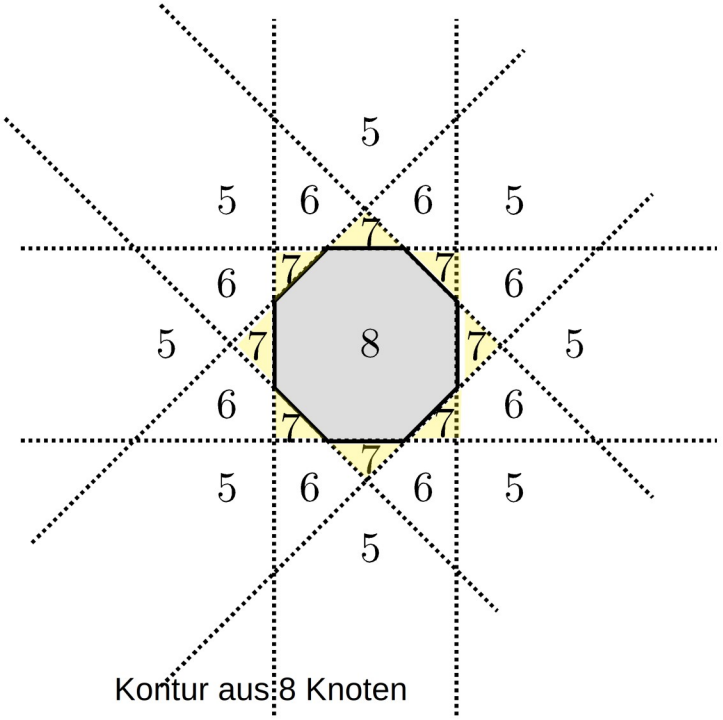
Klassifikation mit Hilfe von MLPs

- **Antwort**: es ist möglich, aber eine allgemeine Konstruktion ist schwieriger, als Sie vielleicht denken mögen.

- Das liegt an der Schwierigkeit mit dem MLP eine beliebige konvexe Kontur mit endlicher Fläche zu bilden:

- Ein **Achteck** kann mit Hilfe eines MLP mit einer versteckten Lage und 8 Knoten, wie auf Folie 11 gezeigt dargestellt werden.

- Wie groß ist das Verhältnis der abgedeckten Flächen für $T \geq N$ und $T \geq N - 1$?



Kontur aus 8 Knoten

Klassifikation mit Hilfe von MLPs

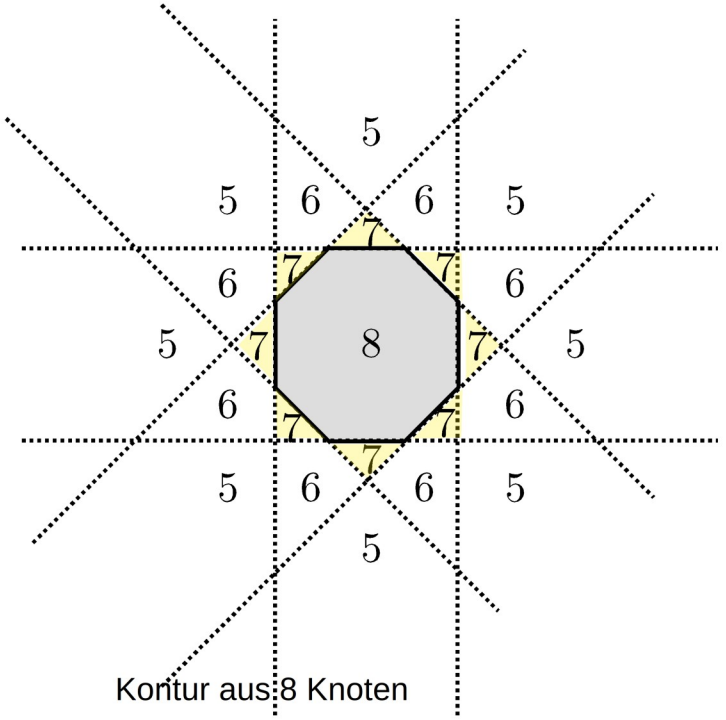
- **Antwort**: es ist möglich, aber eine allgemeine Konstruktion ist schwieriger, als Sie vielleicht denken mögen.

- Das liegt an der Schwierigkeit mit dem MLP eine beliebige konvexe Kontur mit endlicher Fläche zu bilden:

- Ein **Achteck** kann mit Hilfe eines MLP mit einer versteckten Lage und 8 Knoten, wie auf Folie 11 gezeigt dargestellt werden.

- Wie groß ist das Verhältnis der abgedeckten Flächen für $T \geq N$ und $T \geq N - 1$?

- Im Grenzübergang $N \to \infty$ erzeugen Sie so einen Zylinder der im Inneren einen beliebigen Wert (z.B. 1) animmt und außerhalb exponentiell schnell gegen 0 abfällt.

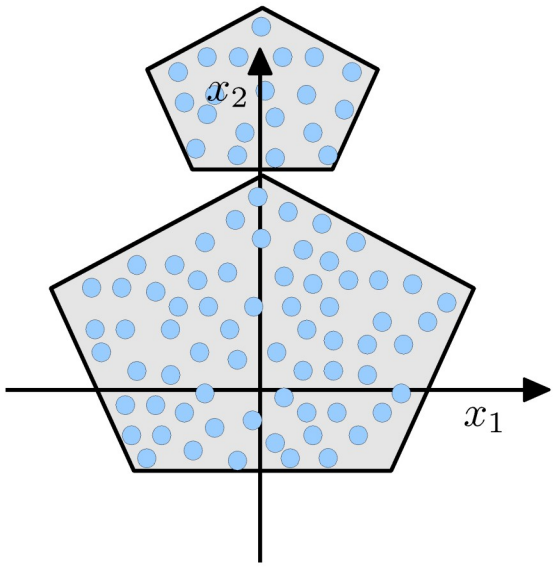


Kontur aus 8 Knoten

Klassifikation mit Hilfe von MLPs

- **Antwort**: es ist möglich, aber eine allgemeine Konstruktion ist schwieriger, als Sie vielleicht denken mögen.

- Mit einer einzigen versteckten Lage können Sie grundsätzlich jede beliebige Kontur approximieren, indem Sie sie mit Zylindern auffüllen deren Schwellen Sie in der output Lage addieren[(1)]:



- Ein solches Netz benötigt i.a. jedoch unendlich viele Knoten.

- Tiefe erlaubt es auch hier komplexe Konturen mit deutlich reduzierter Anzahl an Knoten abzubilden.

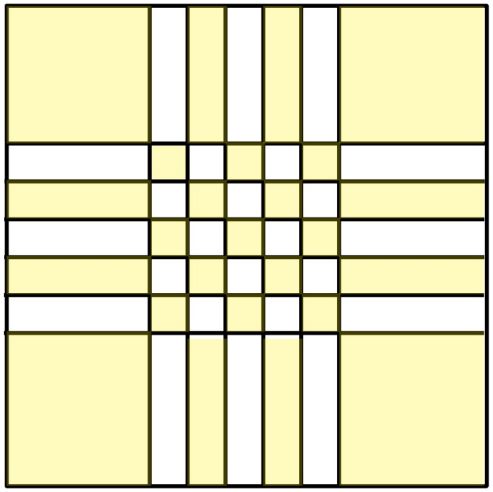
(1) Sie könnten sich fragen: Wozu müssen die Konturen überhaupt begrenzt sein? Ich interessiere mich doch nur für die Stichproben oberhalb der Schwelle. Die Antwort auf diese Frage sehen Sie hier: Wenn Sie die MLPs für die einzelnen Zylinder addieren beeinflussen sich die MLPs untereinander gegenseitig, wenn sie nicht begrenzt sind.

Zusammenfassung

- Es ist möglich, jede beliebige boolsche Funktion mit einem MLP mit einer einzigen versteckten Lage abzubilden.
- Es ist möglich, jede beliebige Kontur im Rahmen eines Klassifikationsproblems mit einem MLP mit einer einzigen versteckten Lage abzubilden.
- Das gleiche gilt für die Approximation beliebiger reelwertiger Funktionen (hier nicht diskutiert, aber analog zur Kontur).
- Im allgemeinen benötigt man hierzu jedoch unendlich viele Knoten.
- Die Anzahl der Knoten läßt sich signifikant reduzieren, wenn man dem MLP Tiefe verleiht → *depth matters*.
- Man sagt tiefe Netzwerke sind i.a. ausdrucksstärker (*expressiver*), als flache Netzwerke.

Flache vs. tiefe Netzwerke

- Betrachten Sie das folgende Beispiel:



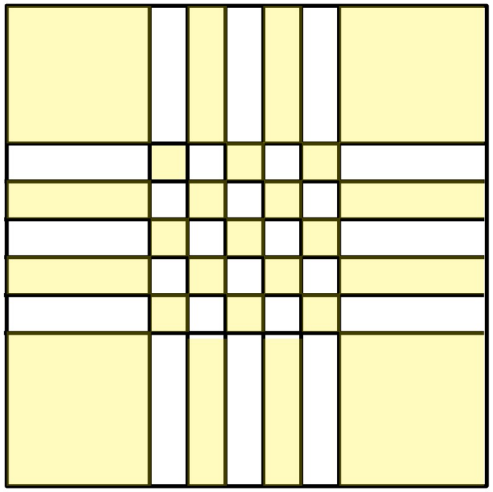
Wert 1

Wert 0

- Wieviele Knoten benötigen Sie, um diese Kontur in einem MLP mit einer versteckten Lage abzubilden? –

Flache vs. tiefe Netzwerke

- Betrachten Sie das folgende Beispiel:



Wert 1

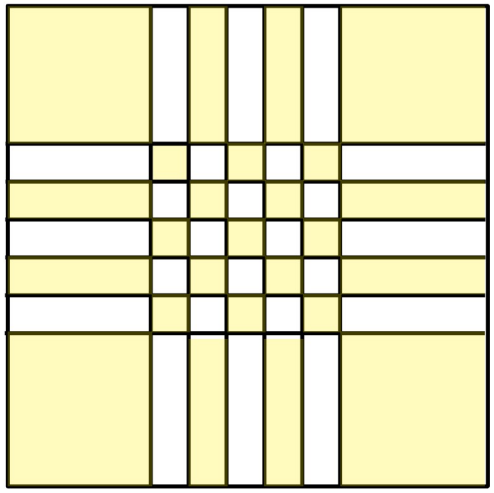
Wert 0

- Wieviele Knoten benötigen Sie, um diese Kontur in einem MLP mit einer versteckten Lage abzubilden? – **unendlich viele!**

- Ein MLP mit zwei versteckten Lagen benötigt hierzu nicht merh als **61 Knoten**. Wie komme ich darauf?

Flache vs. tiefe Netzwerke

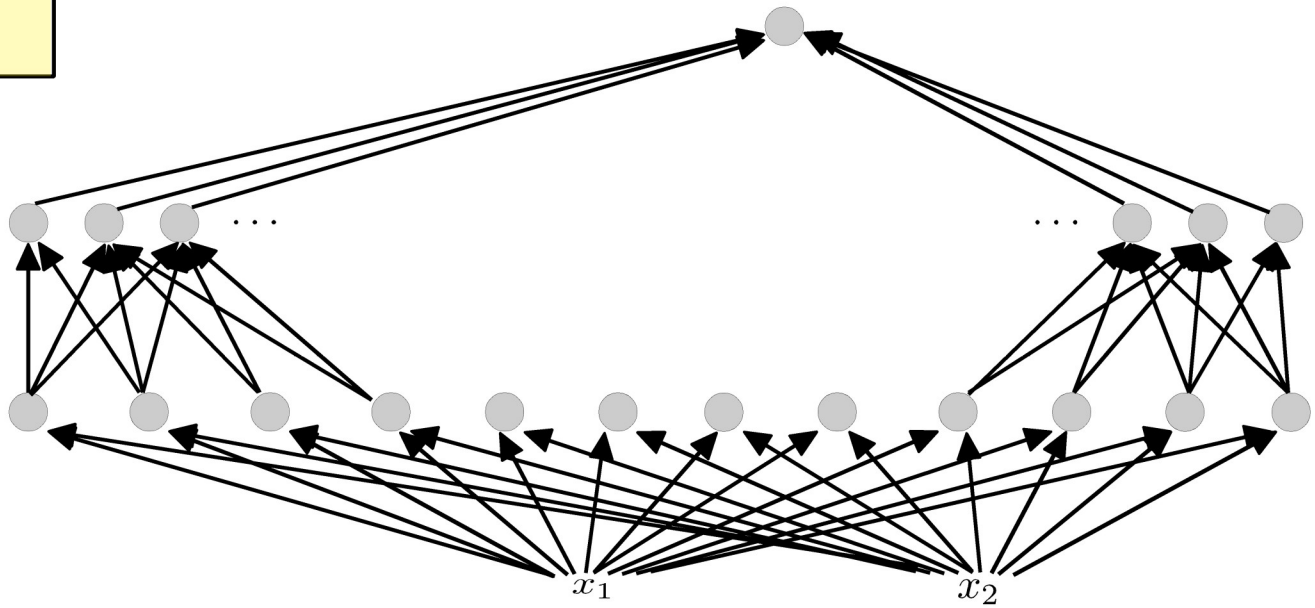
- Betrachten Sie das folgende Beispiel:



Wert 1

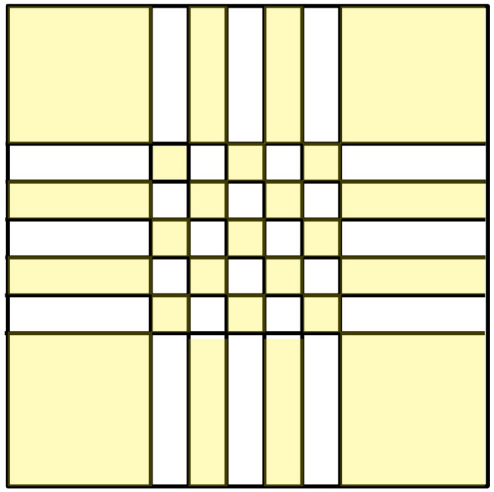
Wert 0

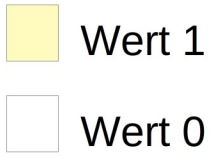
- Die Kontur besteht aus 12 Linien, die in der ersten versteckten Lage identifiziert werden.
- In der zweiten versteckten Lage werden aus dem output der ersten versteckten Lage $7 \times 7 = 49$ Boxen identifiziert, die in der output Lage addiert werden.



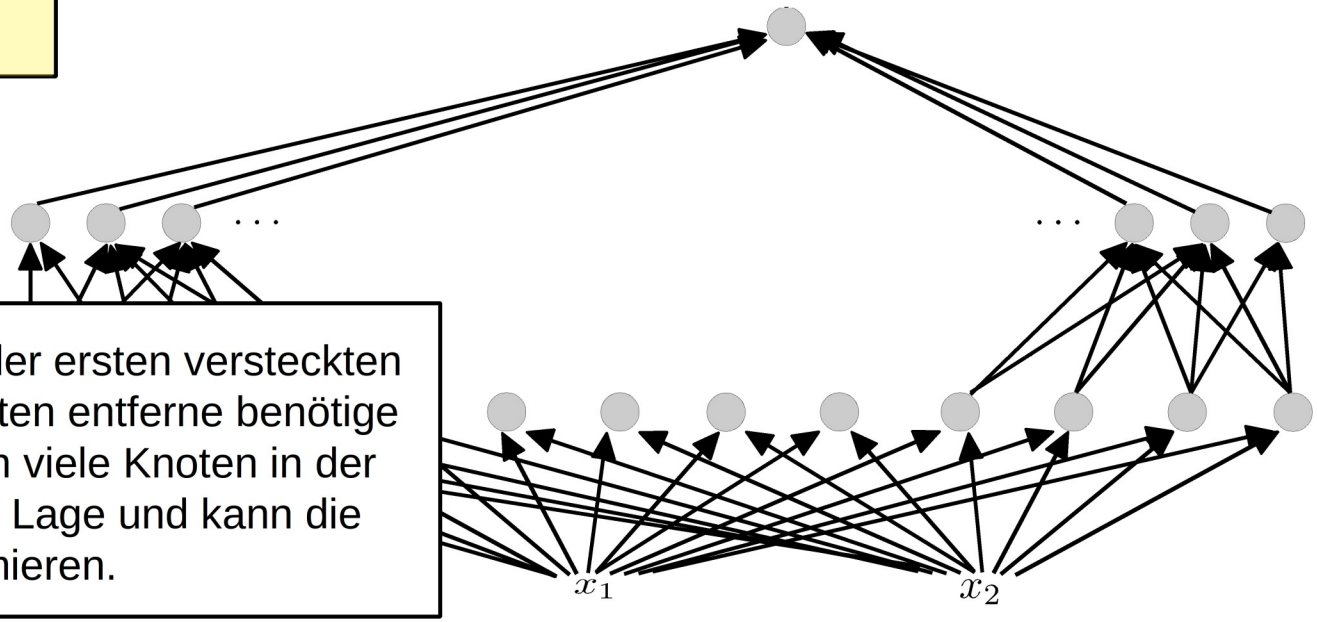
Flache vs. tiefe Netzwerke

- Betrachten Sie das folgende Beispiel:





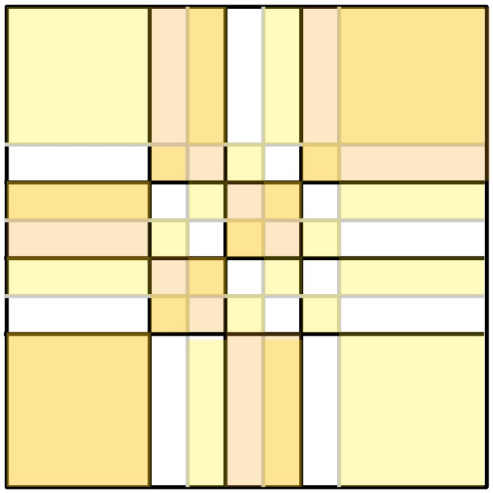
- Die Kontur besteht aus 12 Linien, die in der ersten versteckten Lage identifiziert werden.
- In der zweiten versteckten Lage werden aus dem output der ersten versteckten Lage $7 \times 7 = 49$ Boxen identifiziert, die in der output Lage addiert werden.



NB: Wenn ich aus der ersten versteckten Lage nur einen Knoten entferne benötige ich wieder unendlich viele Knoten in der zweiten versteckten Lage und kann die Kontur nur approximieren.

Flache vs. tiefe Netzwerke

- Betrachten Sie das folgende Beispiel:

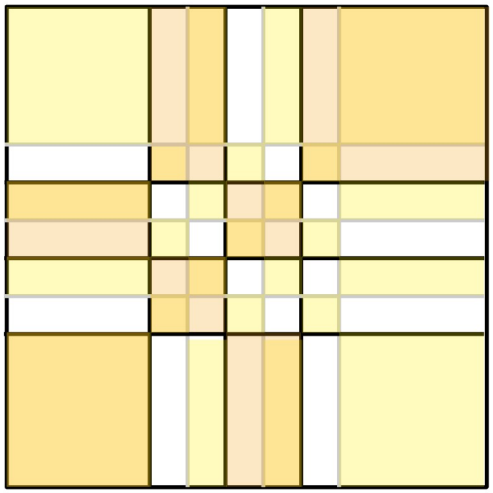


- Stellen Sie sich vor Sie haben in der ersten versteckten Lage nicht 12 sondern nur 6 Knoten zu Verfügung.

- Auch in einem solchen Arrangement, wie links gezeigt würden Sie unendlich viele Knoten benötigen, weil Sie nach passieren der ersten versteckten Lage nicht wissen, wo Sie sich in der orangefarbenen Box befinden.

Flache vs. tiefe Netzwerke

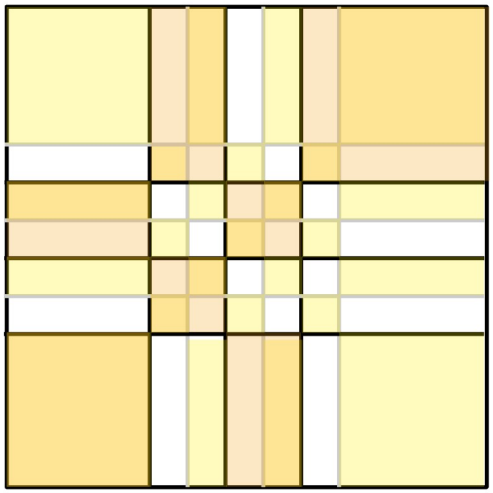
- Betrachten Sie das folgende Beispiel:



- Stellen Sie sich vor Sie haben in der ersten versteckten Lage nicht 12 sondern nur 6 Knoten zu Verfügung.
- Auch in einem solchen Arrangement, wie links gezeigt würden Sie unendlich viele Knoten benötigen, weil Sie nach passieren der ersten versteckten Lage nicht wissen, wo Sie sich in der orangefarbenen Box befinden.
- Das liegt an der Verwendung einer einfachen Stufenfunktion als Aktivierungsfunktion in unserem Beispiel.

Flache vs. tiefe Netzwerke

- Betrachten Sie das folgende Beispiel:



- Stellen Sie sich vor Sie haben in der ersten versteckten Lage nicht 12 sondern nur 6 Knoten zu Verfügung.
- Auch in einem solchen Arrangement, wie links gezeigt würden Sie unendlich viele Knoten benötigen, weil Sie nach passieren der ersten versteckten Lage nicht wissen, wo Sie sich in der orangefarbenen Box befinden.
- Das liegt an der Verwendung einer einfachen Stufenfunktion als Aktivierungsfunktion in unserem Beispiel.

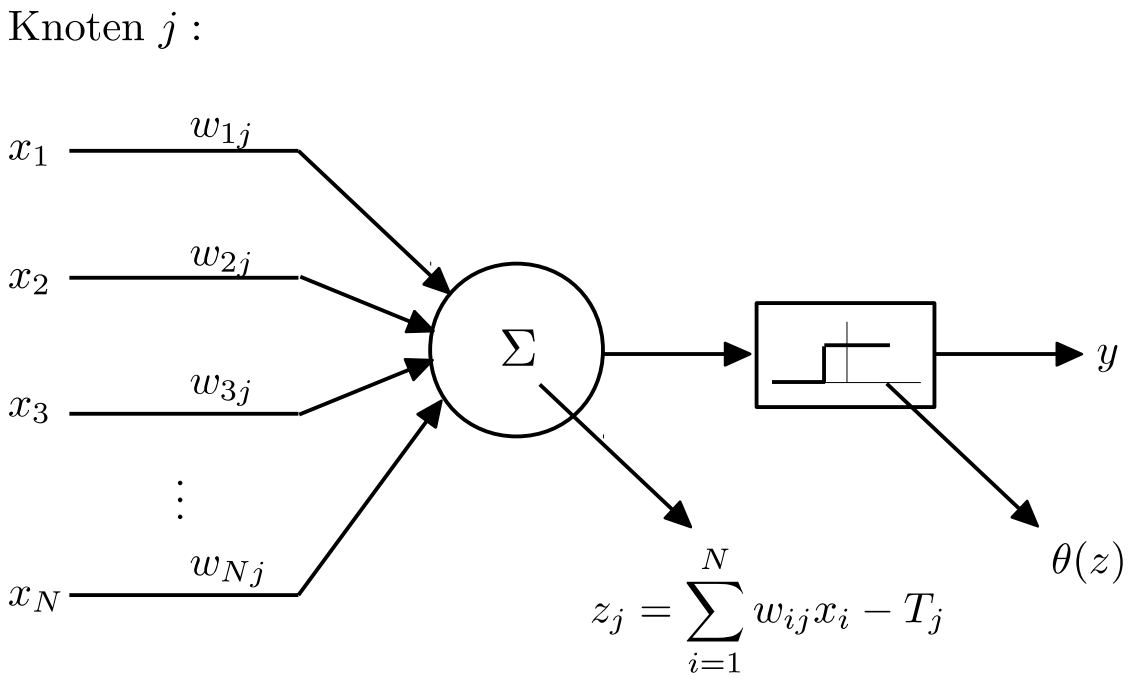
NB: Wählen Sie als Aktivierungsfunktion Ihrer Knoten eine Funktion, die noch Information darüber enthält, wo Sie sich relativ zur Schwelle der begrenzenden Hyperfläche befinden (siehe Folie 4) erlaubt das Ihrem MLP Information, die die erste Lage nicht verarbeiten kann in folgenden Lagen zu rekonstruieren. Das funktioniert umso besser je besser die Aktivierungsfunktion diese Information übermitteln kann.

Lernen

- Bisher haben wir uns mit der Frage, wie man die richtigen Gewichte und Schwellen (*biases*) zur Konfiguration der MLPs findet noch nicht beschäftigt. Das passiert im folgenden.

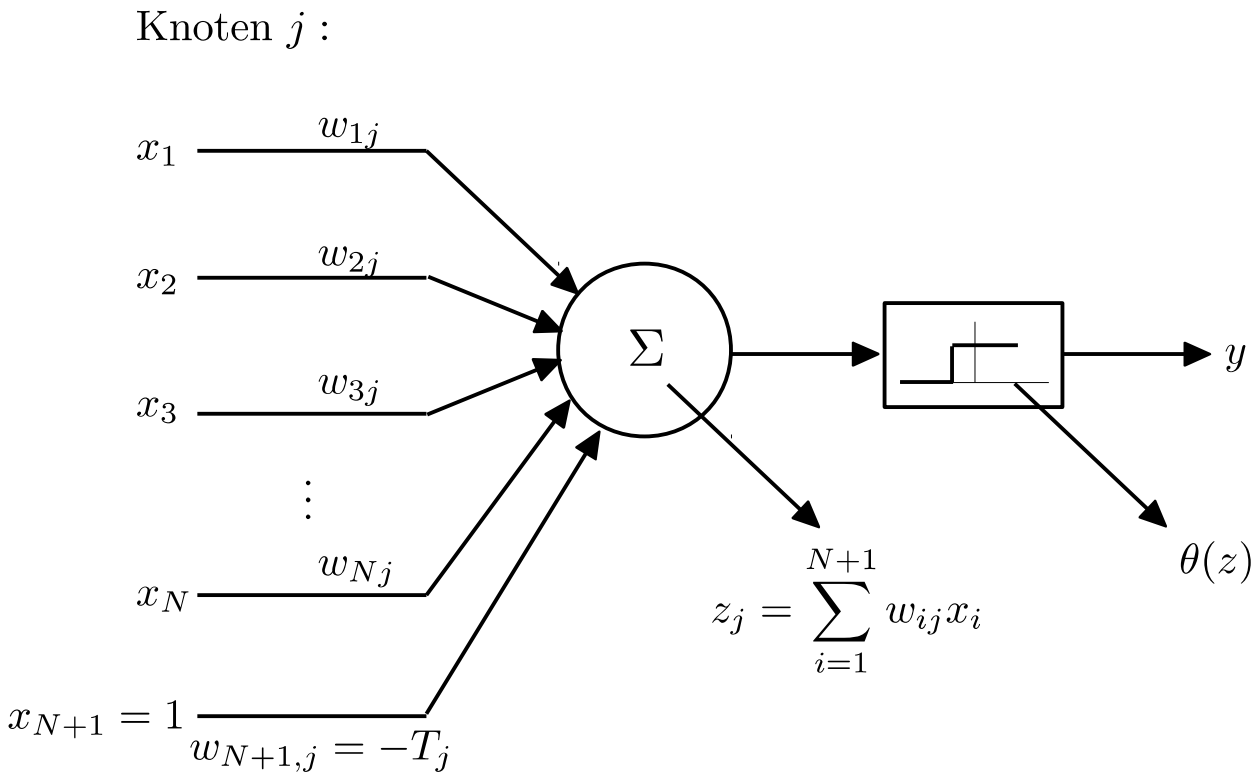
Erinnerung Perceptron

- **Erinnerung**: Das MLP ist eine wohldefinierte multidimensionale Funktion, sowohl der Eingangsparameter $\{x_i\}$ als auch der Gewichte $\{w_{ij}\}$ und Schwellen $\{T_j\}$:



Erinnerung Perceptron

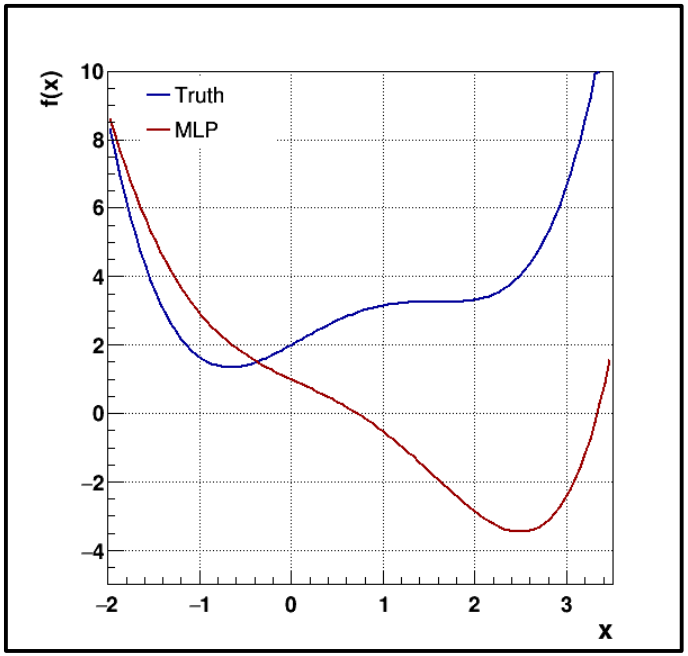
- **Erinnerung**: Das MLP ist eine wohldefinierte multidimensionale Funktion, sowohl der Eingangsparameter $\{x_i\}$ als auch der Gewichte $\{w_{ij}\}$ und Schwellen $\{T_j\}$:



- **NB**: Wir werden im folgenden aus Gründen der Klarheit auch diese strikt äquivalente Formulierung des boolschen Perceptrons benutzen in der nur Gewichte vorkommen.

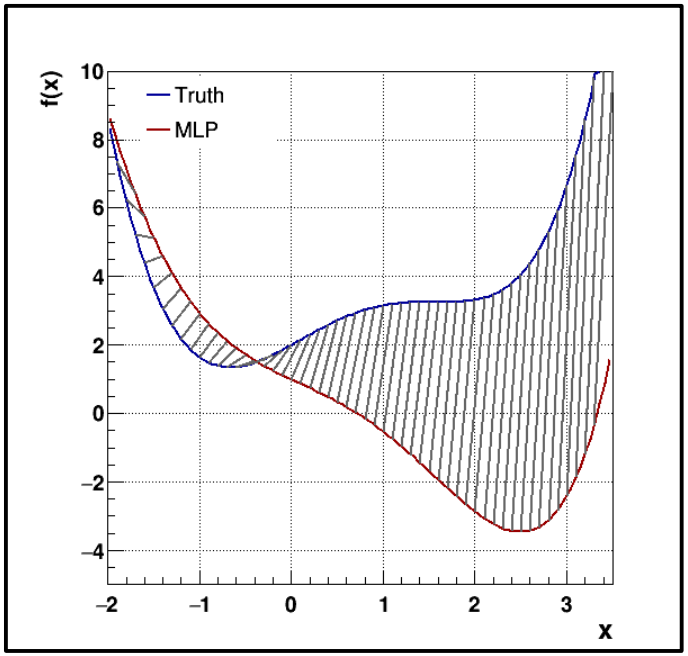
Wahrheit vs. Vorhersage

- Nehmen wir an das MLP solle eine beliebige Funktion darstellen (→ blaue Kurve, rechts):
- Nehmen wir weiter an das MLP besäße eine geeignete Architektur, um das zu tun.
- Eine zufällige Wahl der $\{w_{ij}\}$ wird sicher nicht die gewünschte Funktion abbilden (→ rote Kurve, rechts).



Wahrheit vs. Vorhersage

- Nehmen wir an das MLP solle eine beliebige Funktion darstellen (→ blaue Kurve, rechts):
- Nehmen wir weiter an das MLP besäße eine geeignete Architektur, um das zu tun.
- Eine zufällige Wahl der $\{w_{ij}\}$ wird sicher nicht die gewünschte Funktion abbilden (→ rote Kurve, rechts).



- Der erste Schritt zu einem besseren Ergebnis besteht darin, den Unterschied der dargestellten und der gewünschten Funktion zu quantifizieren. Dies ist in diesem Beispiel z.B. durch das Integral des Betrages der Differenz beider Funktionen möglich.
- Auf dieser Metrik handelt es sich bei der Lösung der Aufgabe um ein multidimensionales Minimierungsproblem.

Stichprobe und Training

- Im Allgemeinen kennen wir die wahre Funktion nicht (siehe Lecture-10 Folie 3). Wir müssen sie aus einer Stichprobe bestimmen und hoffen, dass diese Stichprobe repäsentativ für die zu Grunde liegende Kontur oder Funktion ist.

- Diesen Prozess bezeichnet man als *training*.

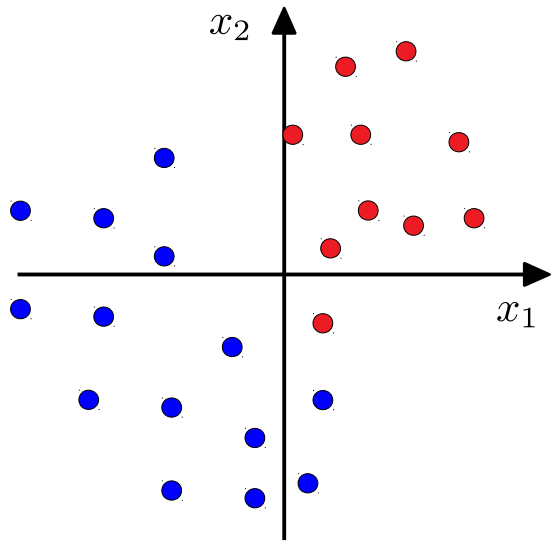
- Damit die Stichprobe repäsentativ sein kann muss sie alle wesentlichen Eigenschaften der wahren Funktion erfassen. Individuelle Eigenschaften der Stichprobe sollten den Minimierungsprozess nicht beeinflussen. Man bezeichnet diesen Komplex als Generalisierungseigenschaft.

- Ein MLP mit so vielen Parametern, dass es jeden Punkt der Stichprobe trifft läuft in die Gefahr des *overfitting* (hier auch *overtraining* genannt). Ein MLP mit zu wenigen Parametern, um alle Eigenschaften der wahren Funktion erfassen zu können wird nicht die optimale Performance aufweisen (*underfitting*).

- Da die wahre Funktion nicht bekannt ist besteht bei der Wahl der MLP Struktur i.a. die Tendenz zu viele Parameter zu wählen. Was dann passieren kann ist, dass das MLP (im Fall von overtraining) nicht die optimalen Generalisierungseigenschaften aufweist.

Perceptron lerning rule

- **Beispiel**: Training eines einzelnen boolschen Perceptrons zur Trennung zweier Ereignisklassen mit Hilfe gelabelter Stichproben (hier als Punkte dargestellt)[(1)]:



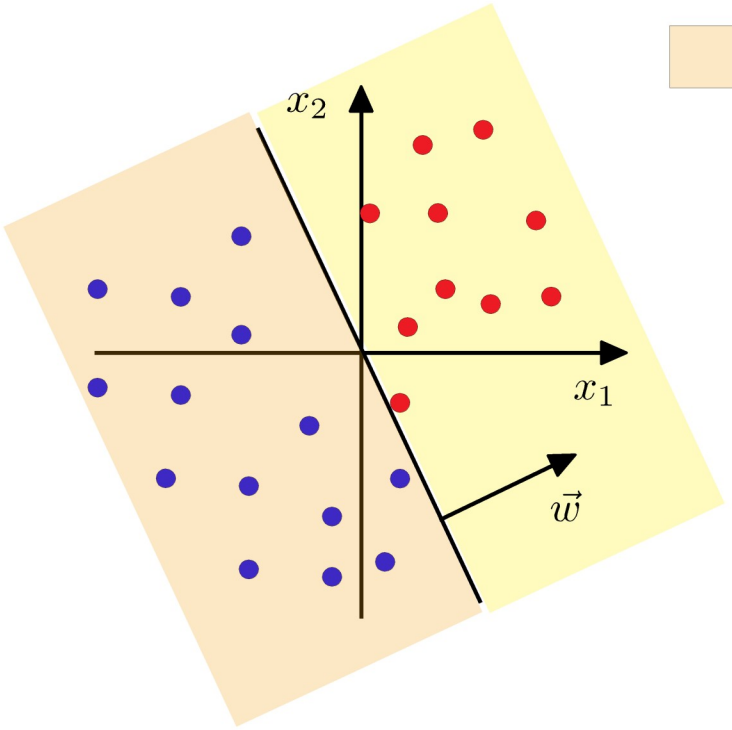
- **Aufgabe**: Bestimme die Gewichte des Perceptrons so, dass die roten Punkte den Wert 1 und die blauen Punkte den Wert 0 erhalten.

- **NB**: Die Lösung ist nicht eindeutig, weil die Stichprobe nicht den gesamten Raum über x_1 und x_2 erfasst.

(1) Es handelt sich hier um das historische Beispiel von Frank Rosenblatt aus den 1960er Jahren.

Beispiel: Perceptron lerning rule

- **Lösung**: Hyperebene in Jordan'scher Normalenform $\sum_i w_i x_i = 0$ d.h. $\vec{w} \perp \vec{x} \quad \forall \vec{x}$ in der Ebene.



$\vec{w} \cdot \vec{x} < 0$ —— $\vec{w} \cdot \vec{x} = 0$ $\vec{w} \cdot \vec{x} > 0$

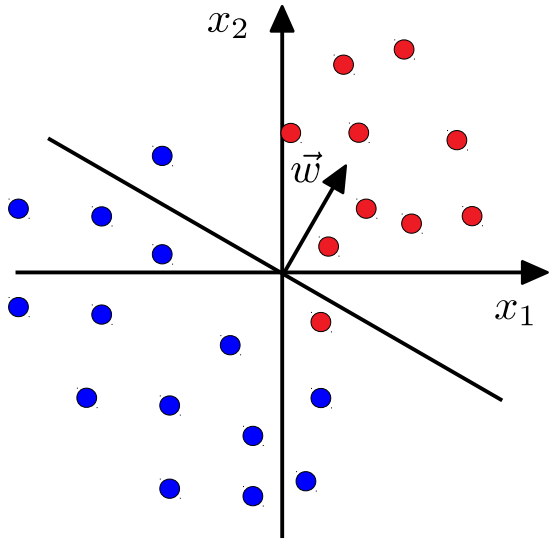
Algorithmus:

- Initialisiere Gewichte und Schwellen zufällig.
- Aktualisiere nur für Stichproben mit falscher Vorhersage.
- Für diese gilt die folgende Lernregel:

$$\vec{w}' = \begin{cases} \vec{w} + \vec{x}_k & \text{rot} \\ \vec{w} - \vec{x}_k & \text{blau} \end{cases}$$

Beispiel: Perceptron lerning rule

- **Lösung**: Hyperebene in Jordan‘scher Normalenform $\sum_i w_i x_i = 0$ d.h. $\vec{w} \perp \vec{x} \quad \forall \vec{x}$ in der Ebene.



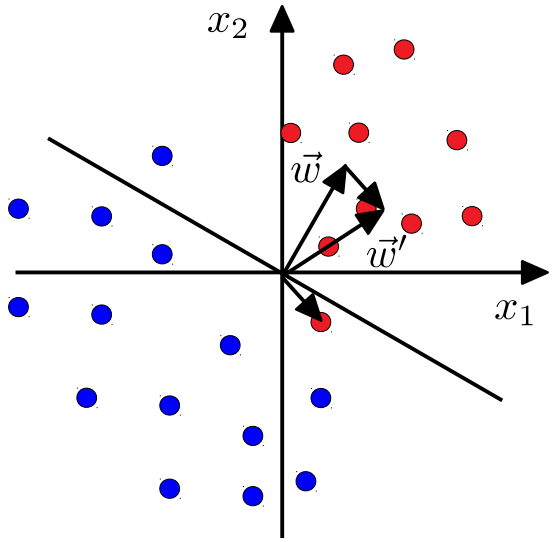
Algorithmus:

- Initialisiere Gewichte und Schwellen zufällig.
- Aktualisiere nur für Stichproben mit falscher Vorhersage.
- Für diese gilt die folgende Lernregel:

$$\vec{w}' = \begin{cases} \vec{w} + \vec{x}_k & \text{rot} \\ \vec{w} - \vec{x}_k & \text{blau} \end{cases}$$

Beispiel: Perceptron lerning rule

- **Lösung**: Hyperebene in Jordan‘scher Normalenform $\sum_i w_i x_i = 0$ d.h. $\vec{w} \perp \vec{x} \quad \forall \vec{x}$ in der Ebene.



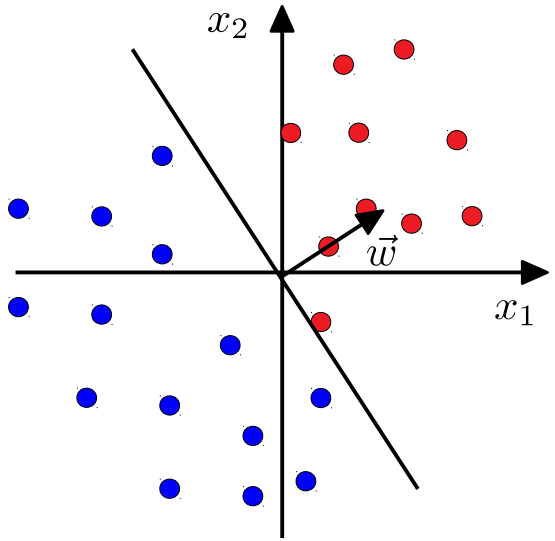
Algorithmus:

- Initialisiere Gewichte und Schwellen zufällig.
- Aktualisiere nur für Stichproben mit falscher Vorhersage.
- Für diese gilt die folgende Lernregel:

$$\vec{w}' = \begin{cases} \vec{w} + \vec{x}_k & \text{rot} \\ \vec{w} - \vec{x}_k & \text{blau} \end{cases}$$

Beispiel: Perceptron lerning rule

- **Lösung**: Hyperebene in Jordan‘scher Normalenform $\sum_i w_i x_i = 0$ d.h. $\vec{w} \perp \vec{x} \quad \forall \vec{x}$ in der Ebene.



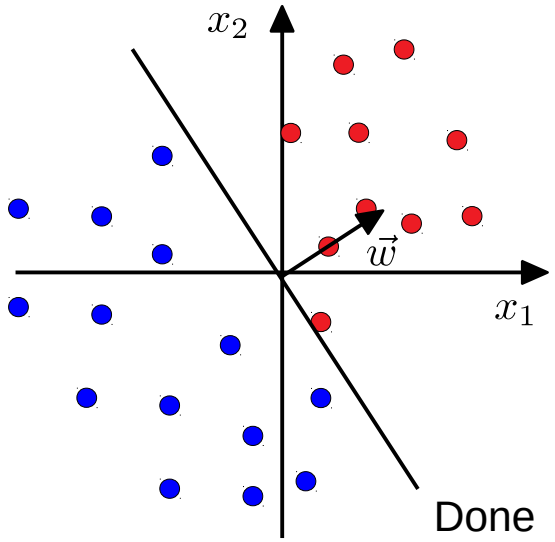
Algorithmus:

- Initialisiere Gewichte und Schwellen zufällig.
- Aktualisiere nur für Stichproben mit falscher Vorhersage.
- Für diese gilt die folgende Lernregel:

$$\vec{w}' = \begin{cases} \vec{w} + \vec{x}_k & \text{rot} \\ \vec{w} - \vec{x}_k & \text{blau} \end{cases}$$

Beispiel: Perceptron lerning rule

- **Lösung**: Hyperebene in Jordan‘scher Normalenform $\sum_i w_i x_i = 0$ d.h. $\vec{w} \perp \vec{x} \quad \forall \vec{x}$ in der Ebene.



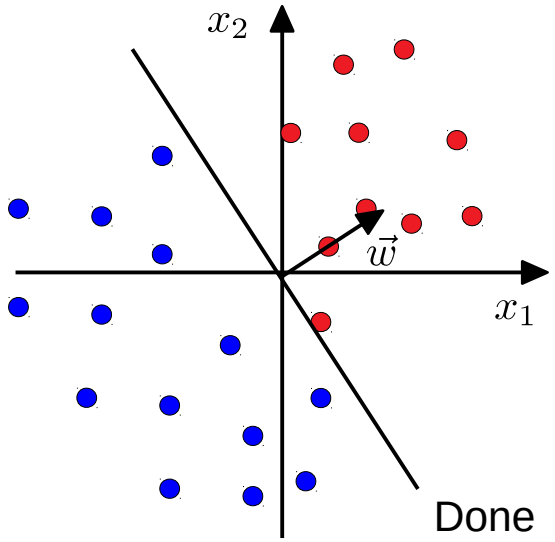
Algorithmus:

- Initialisiere Gewichte und Schwellen zufällig.
- Aktualisiere nur für Stichproben mit falscher Vorhersage.
- Für diese gilt die folgende Lernregel:

$$\vec{w}' = \begin{cases} \vec{w} + \vec{x}_k & \text{rot} \\ \vec{w} - \vec{x}_k & \text{blau} \end{cases}$$

Beispiel: Perceptron lerning rule

- **Lösung**: Hyperebene in Jordan‘scher Normalenform $\sum_i w_i x_i = 0$ d.h. $\vec{w} \perp \vec{x} \quad \forall \vec{x}$ in der Ebene.



Algorithmus:

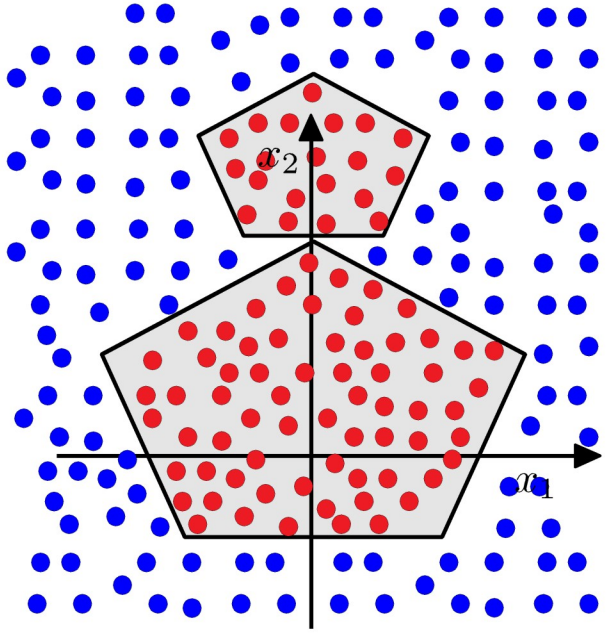
- Initialisiere Gewichte und Schwellen zufällig.
- Aktualisiere nur für Stichproben mit falscher Vorhersage.
- Für diese gilt die folgende Lernregel:

$$\vec{w}' = \begin{cases} \vec{w} + \vec{x}_k & \text{rot} \\ \vec{w} - \vec{x}_k & \text{blau} \end{cases}$$

- **NB**: Rosenblatt hat gezeigt, dass ein einfaches logisches Perceptron für linear separierbare Probleme immer nach endlich vielen Schritten auf eine richtige Lösung konvergiert. Es lässt sich sogar eine obere Schranke für die Anzahl der benötigten Schritte angeben.

Beispiel: Perceptron lerning rule

- Der Versuch diese Lernregel auf komplexere Konturen anzuwenden scheitert. Warum?



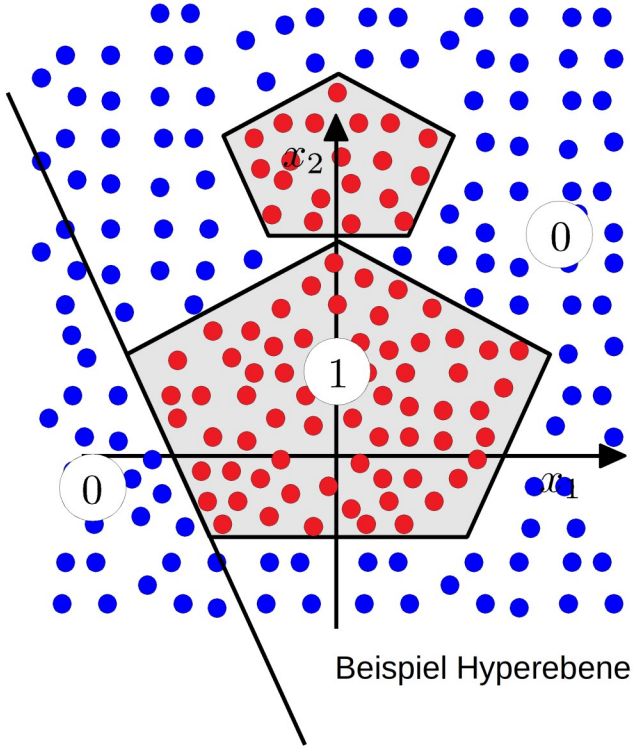
Algorithmus:

- Initialisiere Gewichte und Schwellen zufällig.
- Aktualisiere nur für Stichproben mit falscher Vorhersage.
- Für diese gilt die folgende Lernregel:

$$\vec{w}' = \begin{cases} \vec{w} + \vec{x}_k & \text{rot} \\ \vec{w} - \vec{x}_k & \text{blau} \end{cases}$$

Beispiel: Perceptron lerning rule

- Der Versuch diese Lernregel auf komplexere Konturen anzuwenden scheitert. Warum?
- Für die angegebene Kontur sind die Stichproben nicht linear separierbar!



Beispiel Hyperebene

Algorithmus:

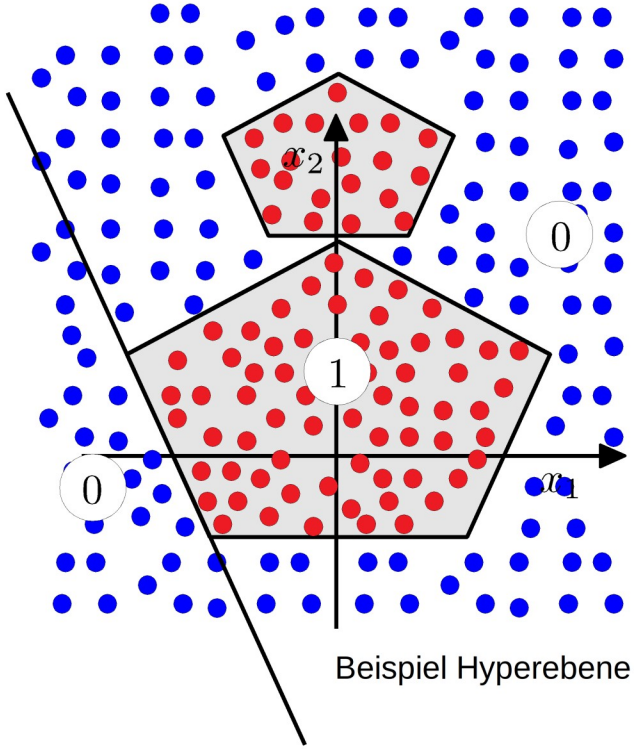
- Initialisiere Gewichte und Schwellen zufällig.
- Aktualisiere nur für Stichproben mit falscher Vorhersage.
- Für diese gilt die folgende Lernregel:

$$\vec{w}' = \begin{cases} \vec{w} + \vec{x}_k & \text{rot} \\ \vec{w} - \vec{x}_k & \text{blau} \end{cases}$$

Priv. Doz. Dr. Roger Wolf
http://ekpwww.physik.uni-karlsruhe.de/~rwolf/

Beispiel: Perceptron lerning rule

- Der Versuch diese Lernregel auf komplexere Konturen anzuwenden scheitert. Warum?
- Für die angegebene Kontur sind die Stichproben nicht linear separierbar!



Beispiel Hyperebene

Algorithmus:

- Initialisiere Gewichte und Schwellen zufällig.
- Aktualisiere nur für Stichproben mit falscher Vorhersage.
- Für diese gilt die folgende Lernregel:

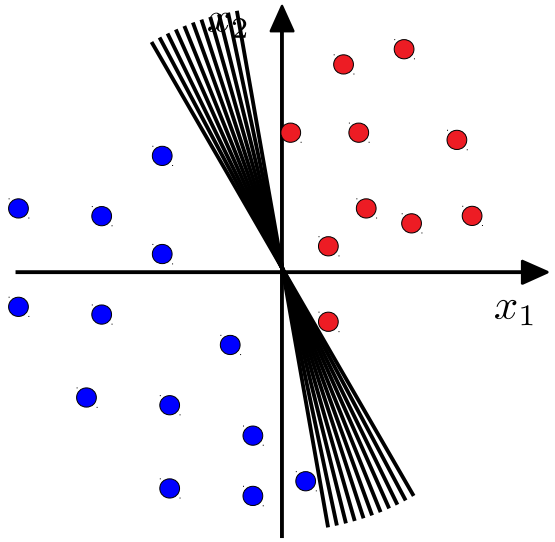
$$\vec{w}' = \begin{cases} \vec{w} + \vec{x}_k & \text{rot} \\ \vec{w} - \vec{x}_k & \text{blau} \end{cases}$$

- **NB**: das Problem ist immer noch lösbar. Man müsste für jede Hyperebene in der Kontur neu labeln. Die Komplexität des Problems wächst jedoch wieder exponentiell an.

Aktivierungsfunktion – revisited

- Das Problem der Perceptron learning rule kommt der aus Verwedung der Stufenfunktion als Schwellenfunktion:

- Kleine Änderungen der Gewichte des Perceptron haben keinen Einfluss auf seinen *output*.



Jede Kurve hier führt zum gleichen output.

- **Lösung**: Führe eine kontinuierliche Aktivierungs-funktion ein.

- Damit ist der output des gesamten Netzwerks differenzierbar in jeder Variablen:

$$y^{(k)}(\vec{w},\vec{x}) = \theta(z(\vec{w},\vec{x})) \quad z^{(k)} = \sum_i w_i y_i^{(k-1)}$$

$$\frac{\mathrm{d}y^{(k)}}{\mathrm{d}w_i} = \frac{\mathrm{d}y^{(k)}}{\mathrm{d}z^{(k)}}\frac{\mathrm{d}z^{(k)}}{\mathrm{d}w_i} = \theta'(z)\,x_i$$

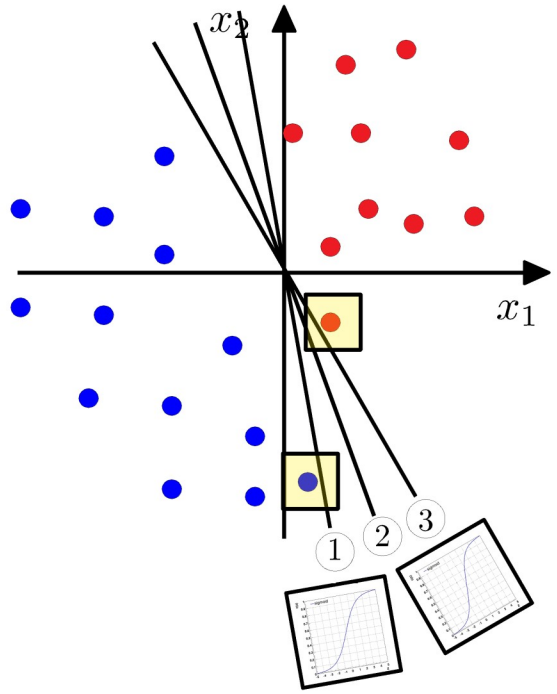
$$\frac{\mathrm{d}y^{(k)}}{\mathrm{d}x_i} = \frac{\mathrm{d}y^{(k)}}{\mathrm{d}z^{(k)}}\frac{\mathrm{d}z^{(k)}}{\mathrm{d}x_i} = \theta'(z)\,w_i$$

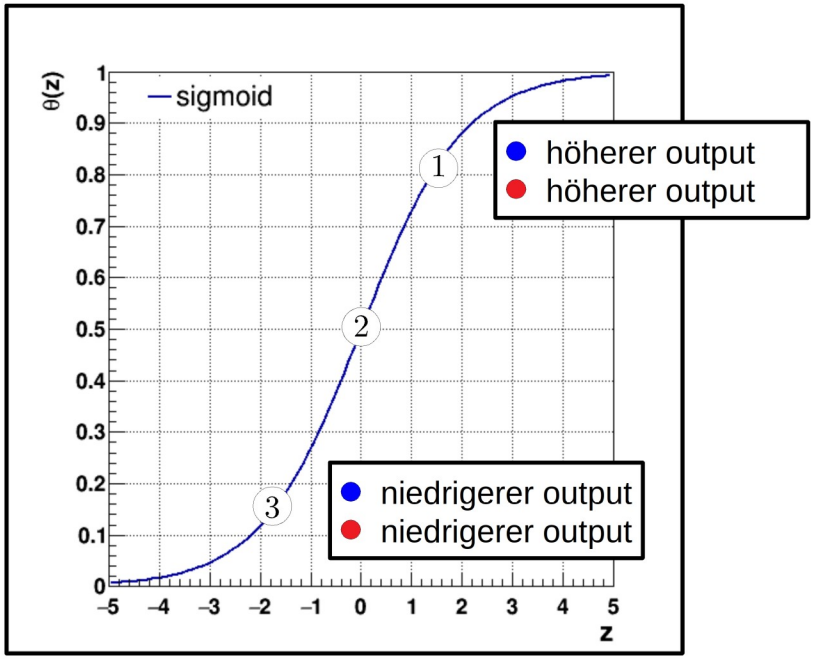
- Das ermöglicht es zu überprüfen, welche kleinen Änderungen in $\vec{w}$ zu welchen Änderungen in $y^{(k)}$ führen.

Aktivierungsfunktion – revisited

- Das Problem der Perceptron learning rule kommt der aus Verwedung der Stufenfunktion als Schwellenfunktion:

- **Beispiel** Sigmoid Funktion:



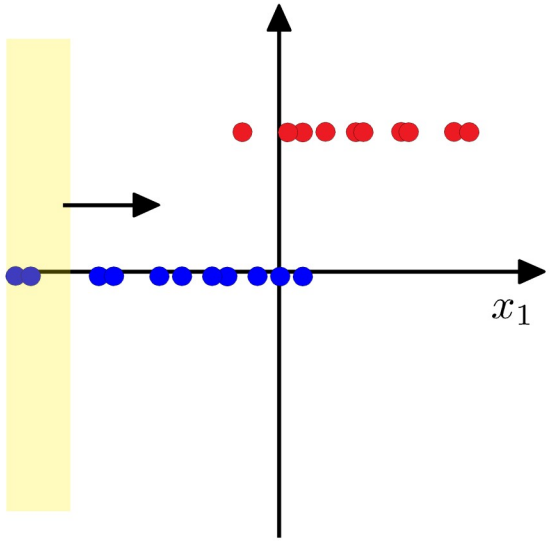


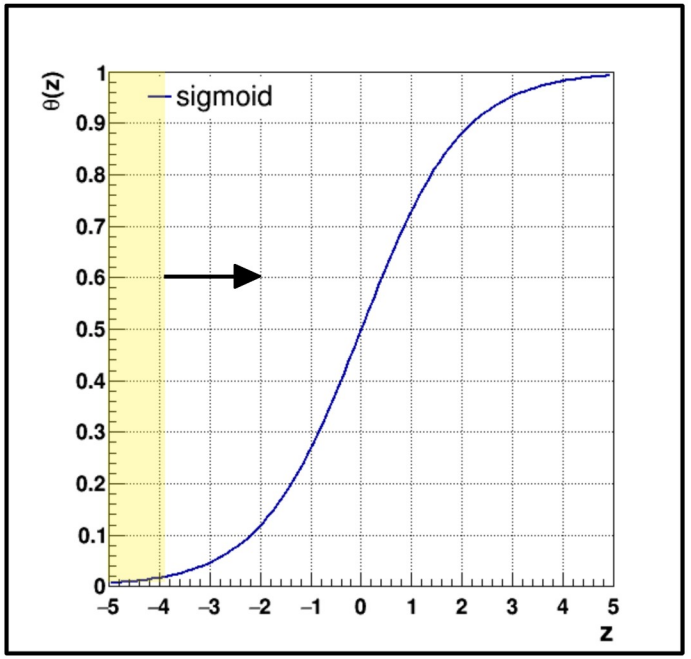
- Graduelle Änderungen der Gewichte führen zu graduellen Änderungen des *outputs*.

Aktivierungsfunktion – revisited

- Das Problem der Perceptron learning rule kommt der aus Verwedung der Stufenfunktion als Schwellenfunktion:

- Die Sigmoid Funktion erlaubt es dem MLP sogar nicht linear-separable Probleme zu addressieren. Wir demonstrieren das an einem 1d Beispiel:



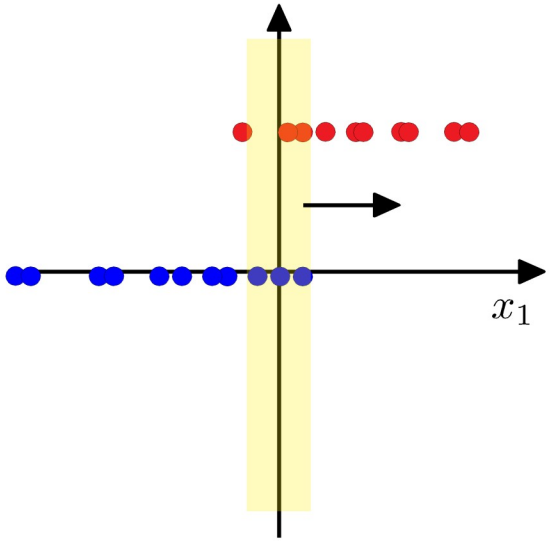


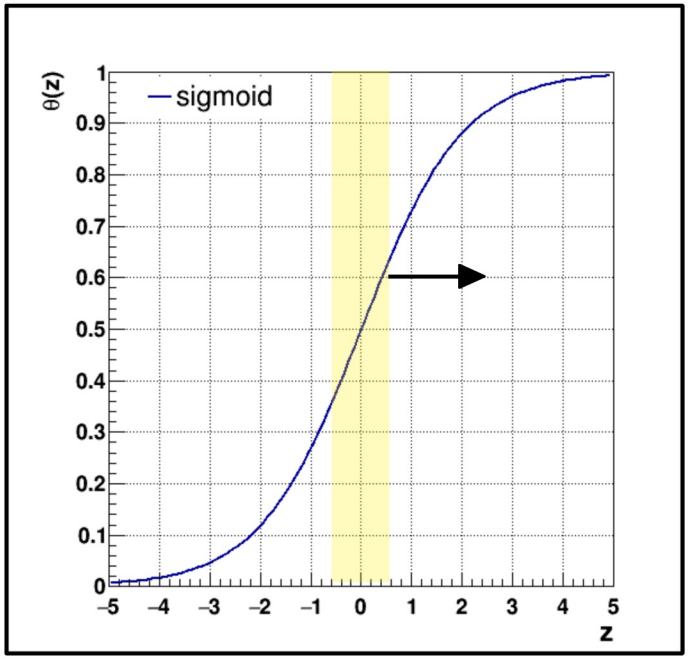
- Stellen Sie sich vor Sie überstreichen x_1 mit einem Fenster konstanter Breite und zählen die rel. Häufigkeit blauer und roter Stichproben. Die Sigmoid Funktion repäsentiert dann in guter Näherung die Wahrscheinlichkeit dafür rot anzutreffen.

Aktivierungsfunktion – revisited

- Das Problem der Perceptron learning rule kommt der aus Verwedung der Stufenfunktion als Schwellenfunktion:

- Die Sigmoid Funktion erlaubt es dem MLP sogar nicht linear-separable Probleme zu addressieren. Wir demonstrieren das an einem 1d Beispiel:



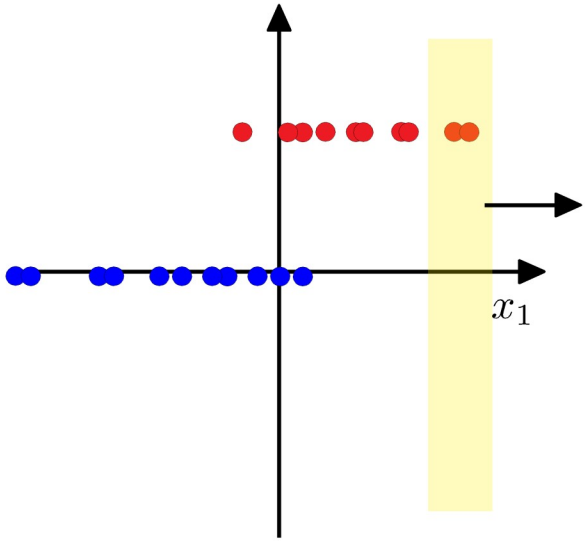


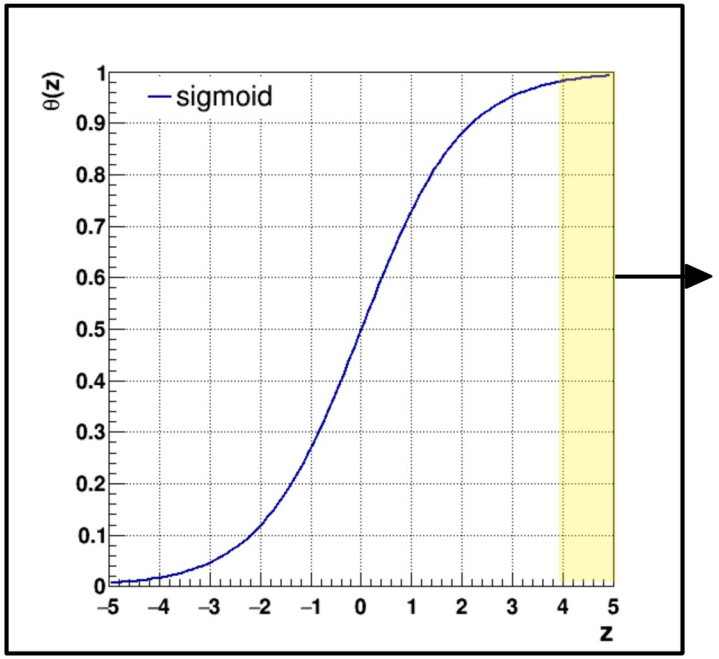
- Stellen Sie sich vor Sie überstreichen x_1 mit einem Fenster konstanter Breite und zählen die rel. Häufigkeit blauer und roter Stichproben. Die Sigmoid Funktion repäsentiert dann in guter Näherung die Wahrscheinlichkeit dafür rot anzutreffen.

Aktivierungsfunktion – revisited

- Das Problem der Perceptron learning rule kommt der aus Verwedung der Stufenfunktion als Schwellenfunktion:

- Die Sigmoid Funktion erlaubt es dem MLP sogar nicht linear-separable Probleme zu addressieren. Wir demonstrieren das an einem 1d Beispiel:





- Stellen Sie sich vor Sie überstreichen x_1 mit einem Fenster konstanter Breite und zählen die rel. Häufigkeit blauer und roter Stichproben. Die Sigmoid Funktion repäsentiert dann in guter Näherung die Wahrscheinlichkeit dafür rot anzutreffen.

Diskussion der Differenzierbarbeit

- Kontinuierliche Aktivierungsfunktionen erlauben Informationen jenseits der Hyperebene des Perceptrons zu erhalten.

- Graduelle Änderungen der Gewichte führen auf graduelle Änderungen des MLP *outputs*. So ist das Minimierungsproblem grundsätzlich analystisch lösbar!

- **Beachten Sie**: Zur Lösung des Minimierungsproblems benötigen Sie die zudem Differenzierbarkeit der Metrik zum Vergleich von MLP output und Wahrheit (siehe Folie 22).

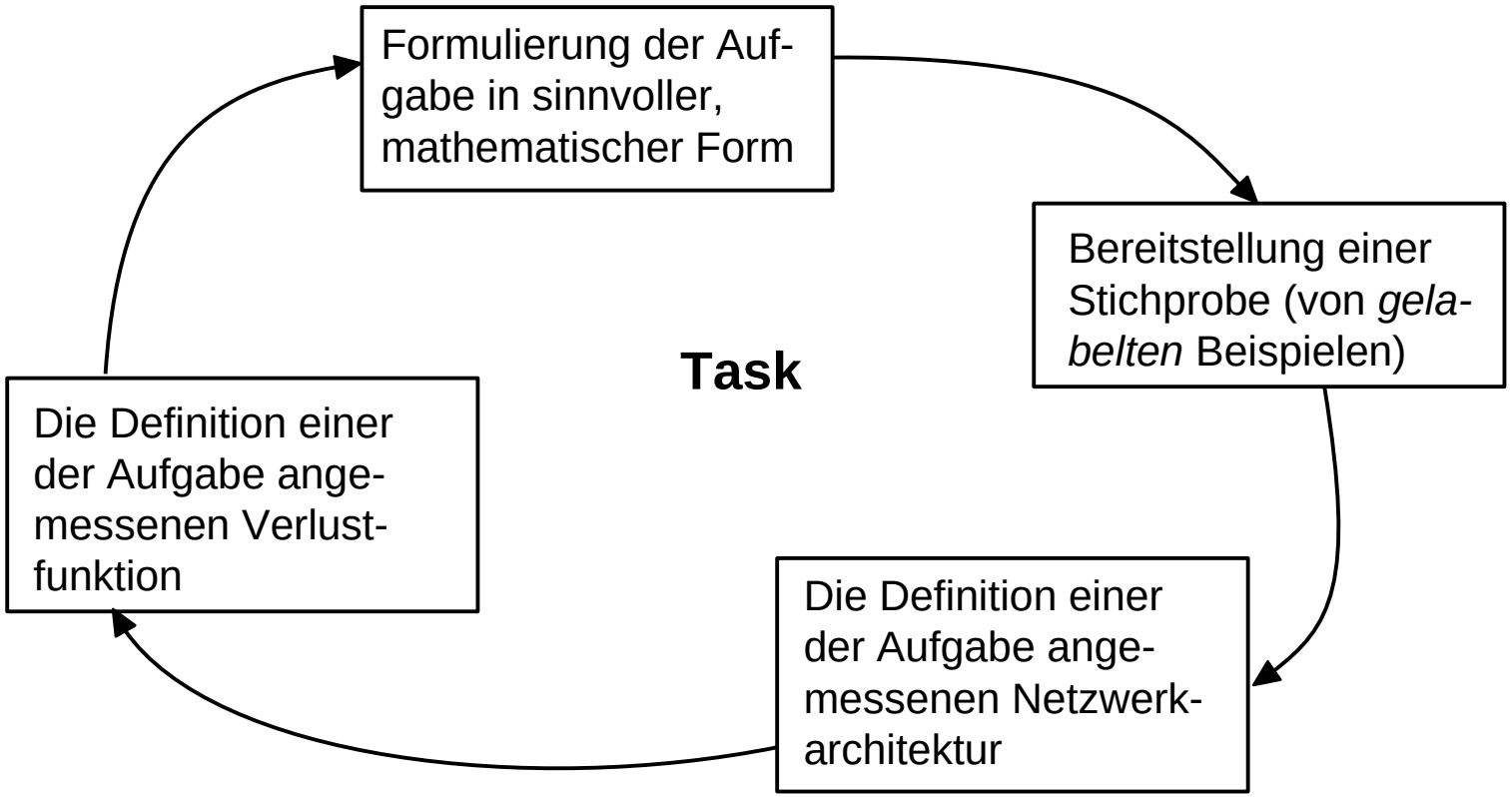
- Wäre Ihnen die Wahrheit bekannt wäre die Größe, die Sie minimieren wollen die erwartete Varianz. Da Ihnen die Wahrheit i.a. nicht bekannt ist handelt es sich um eine statistische Risikoanalyse.

- Die Risikofunktion wird in diesem Zusammenhang als Verlustfunktion (*loss*) bezeichnet.

- Die Verlustfunktion muss nicht streng und von allen Seiten differenzierbar sein. Es genügt, wenn sie bei graduellen Änderungen der Parameter des MLP zu graduellen Änderungen des Wertebereichs führt.

Die NN Aufgabe

- NNs werden konzipiert, um spezifische Aufgaben (*tasks*) zu lösen.
- Zu jeder konkreten Realisierung einer solchen Aufgabe gehören:



Anmerkung

- Wir hatten in der letzten Vorlesung darüber gesprochen, dass es sich beim MLP um eine universelle Linearisierung nicht-trivialer nicht-linearer Problemstellungen in einem höher-dimensionalen Dualraum (der MLP Parameter) handelt.

- Einzelne Perceptrons definieren z.B. Hyperebenen im Raum der Eingangsparameter.

- Diese Aussage erscheint zunächst im Widerspruch dazu, dass in der Wahrnehmung vieler eine nichtlineare Aktivierungsfunktion des MLP eine entscheidene Rolle spielt.

- Beachten Sie jedoch: bisher haben wir zur Diskussion zu keinem Zeitpunkt eine andere Aktivierungsfunktion als die Stufenfunktion verwendet.

- Wie Sie in dieser Vorlesung gesehen haben genügt die Stufenfunktion, um theoretisch alle Probleme in beliebiger Präzision zu lösen. Der Aktivierungsfunktion kommt an zwei (eher technischen) Stellen eine besondere Rolle zu:

  - Bei der Rekonstruktion anderweitig verlorener Information über mehrere Lagen hinweg.

  - Beim Training.

Backup