Moderne Methoden der Datenanalyse
– Ereignisklassifikation –

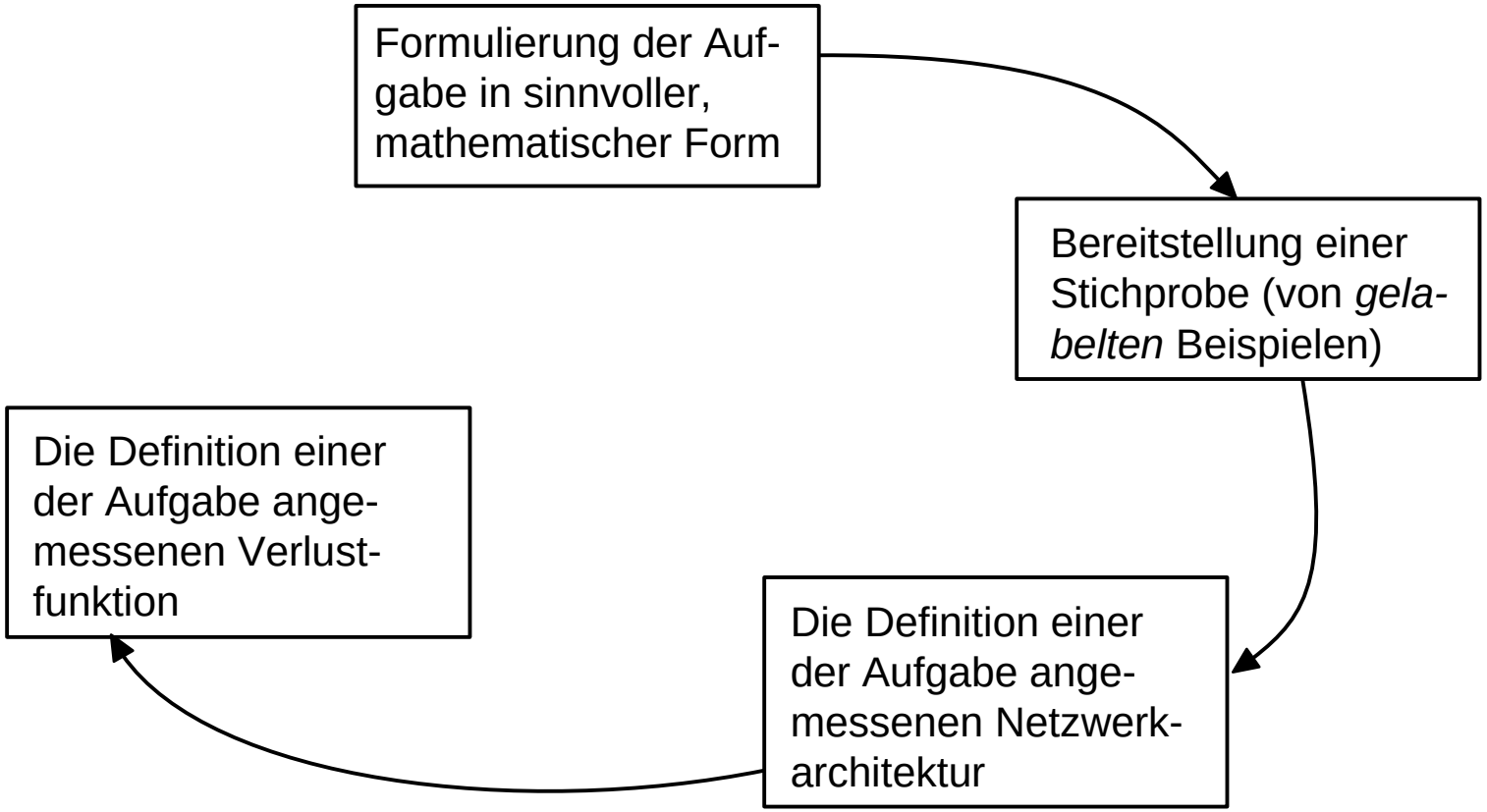
Roger Wolf

16. Juli 2020

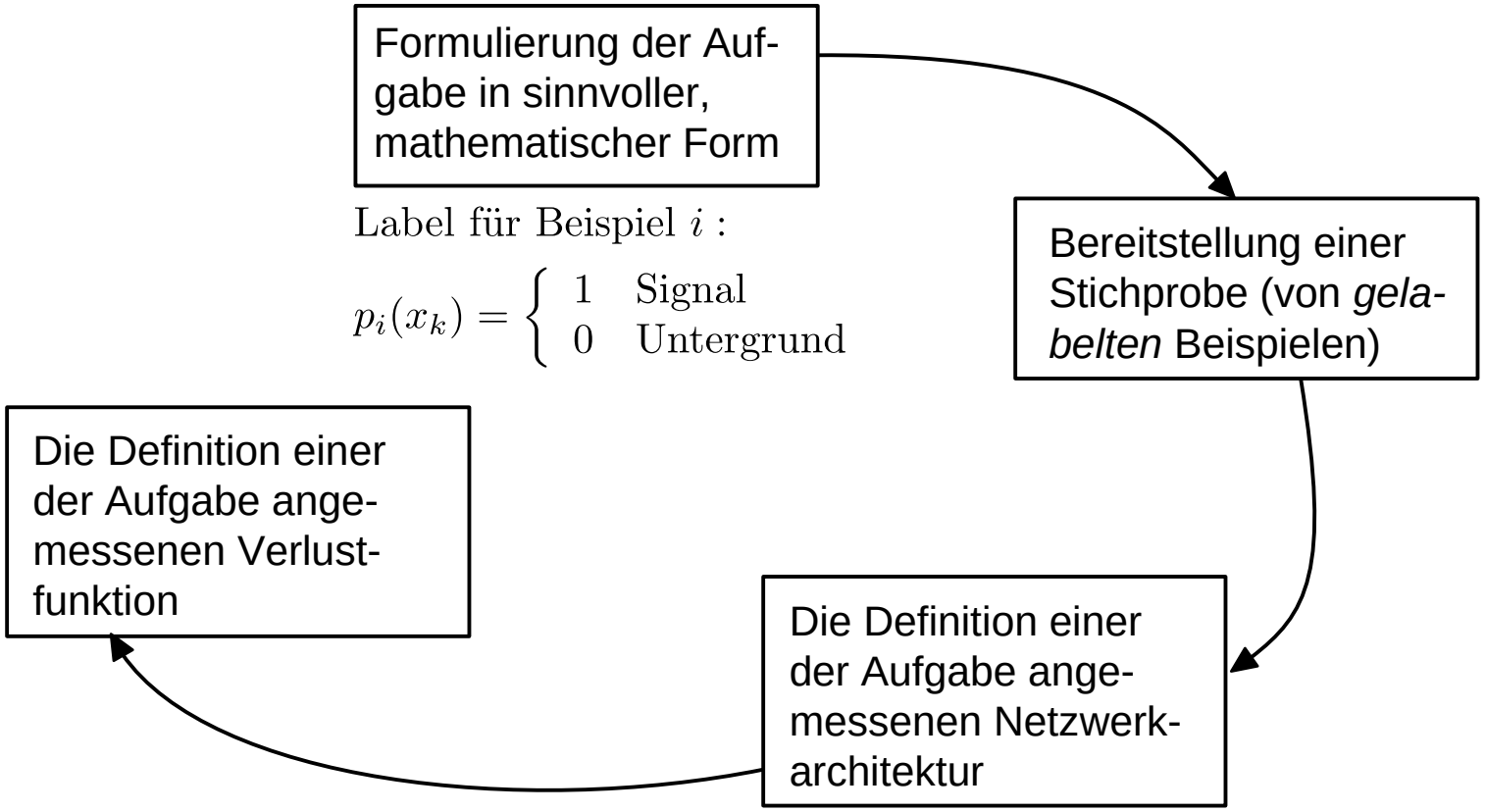
Inhalt der Vorlesung

- Praktisches Vorgehen beim Training.
- Initialisierung der Gewichte vor Trainingsbeginn.
- Trainingsablauf
- Methoden zur Regularisierung

Die NN Aufgabe



Die NN Aufgabe

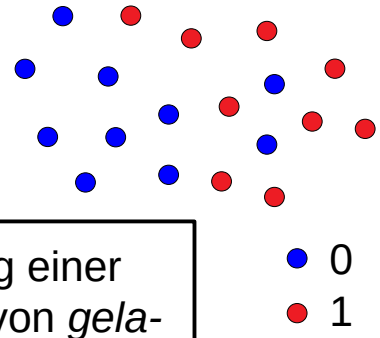


Die NN Aufgabe

Formulierung der Aufgabe in sinnvoller, mathematischer Form

Label für Beispiel i :

$$p_i(x_k) = \begin{cases} 1 & \text{Signal} \\ 0 & \text{Untergrund} \end{cases}$$



Bereitstellung einer Stichprobe (von *gelabelten* Beispielen)

Die Definition einer der Aufgabe angemessenen Netzwerkarchitektur

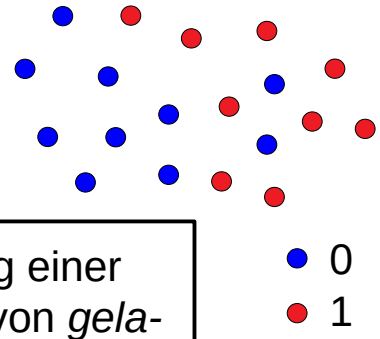
Die Definition einer der Aufgabe angemessenen Verlustfunktion

Die NN Aufgabe

Formulierung der Aufgabe in sinnvoller, mathematischer Form

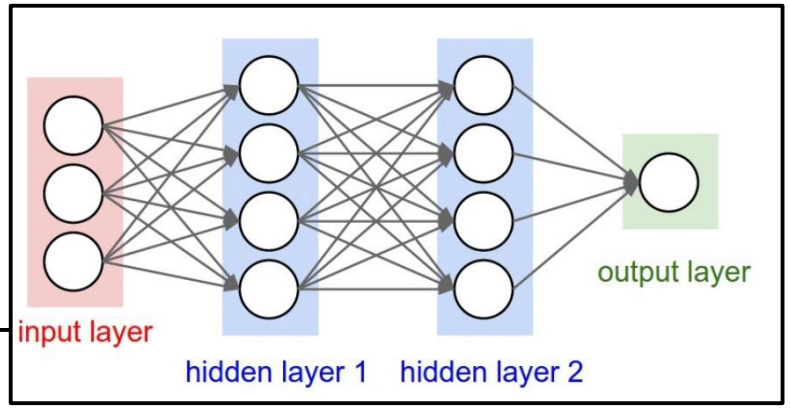
Label für Beispiel i :

$$p_i(x_k) = \begin{cases} 1 & \text{Signal} \\ 0 & \text{Untergrund} \end{cases}$$



Bereitstellung einer Stichprobe (von *gelabelten* Beispielen)

Die Definition einer der Aufgabe angemessenen Netzwerkarchitektur



Die Definition einer der Aufgabe angemessenen Verlustfunktion

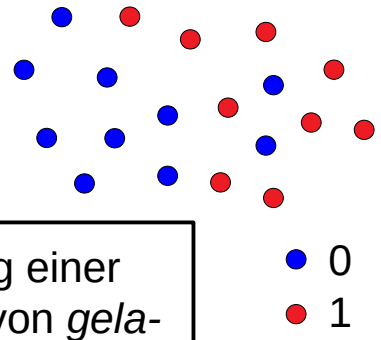
Priv. Doz. Dr. Roger Wolf
http://ekpwww.physik.uni-karlsruhe.de/~rwolf/

Die NN Aufgabe

Formulierung der Aufgabe in sinnvoller, mathematischer Form

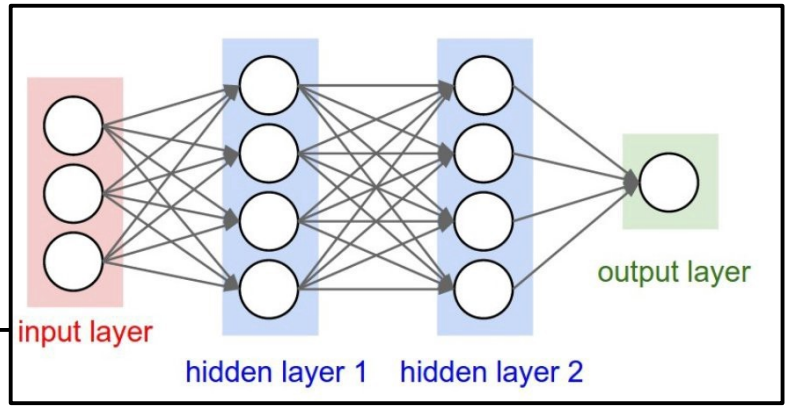
Label für Beispiel i :

$$p_i(x_k) = \begin{cases} 1 & \text{Signal} \\ 0 & \text{Untergrund} \end{cases}$$



Bereitstellung einer Stichprobe (von *gelabelten* Beispielen)

Die Definition einer der Aufgabe angemessenen Netzwerkarchitektur



Die Definition einer der Aufgabe angemessenen Verlustfunktion

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i)$$

n : Länge der Stichprobe

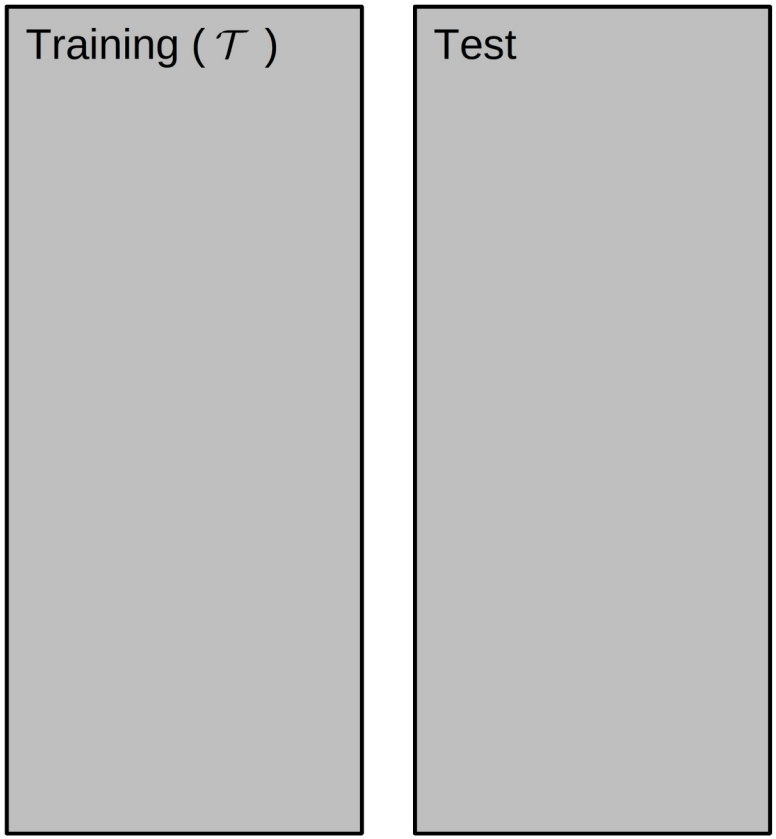
y_i : MLP output für Beispiel i

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

Aufbereitung der Daten zum Training

- **Test Datensatz**:
  Hierauf soll das trainierte NN angewandt werden.

- **Trainingsdatensatz ($\mathcal{T}$)**:
  Hierauf wird das NN trainiert.

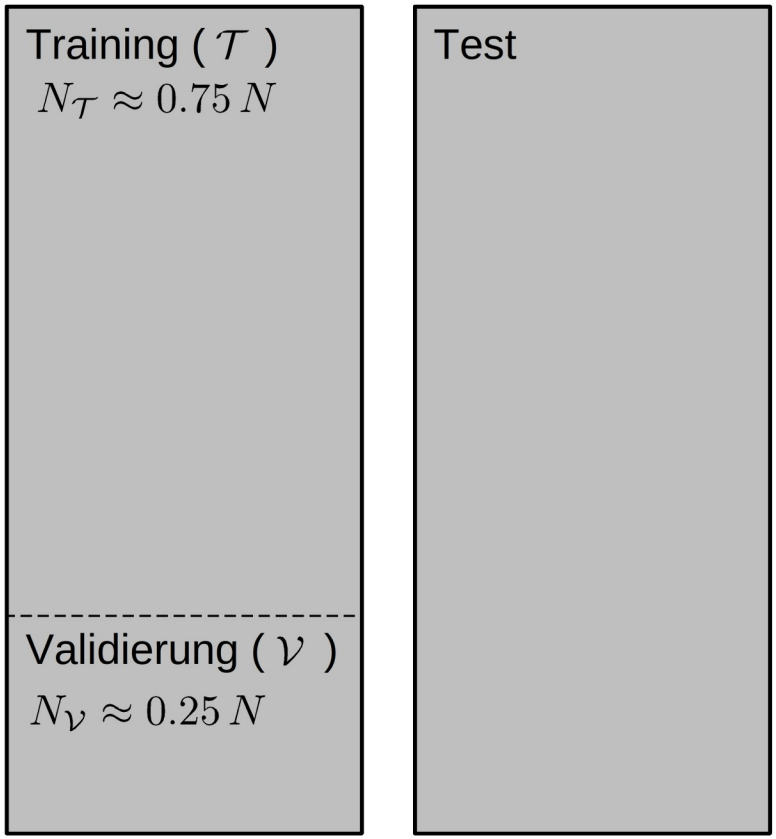


Aufbereitung der Daten zum Training

- **Test Datensatz**:
  Hierauf soll das trainierte NN angewandt werden.

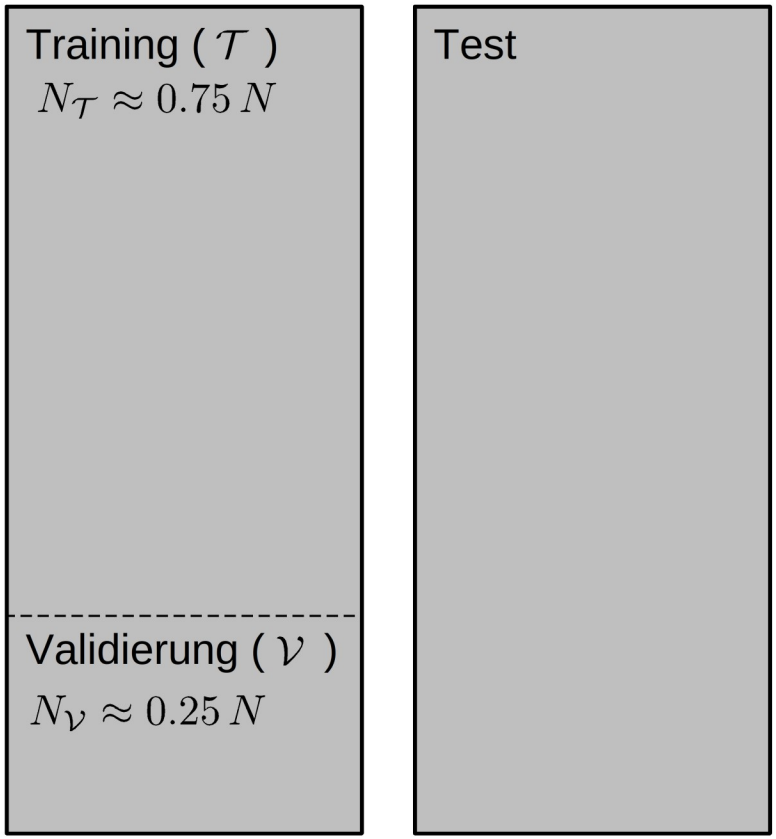
- **Trainingsdatensatz ($\mathcal{T}$)**:
  Hierauf wird das NN trainiert.

- **Validierungsdatensatz ($\mathcal{V}$)**:
  Hierauf wird das NN training validiert.



K-fache Kreuzvalidierung

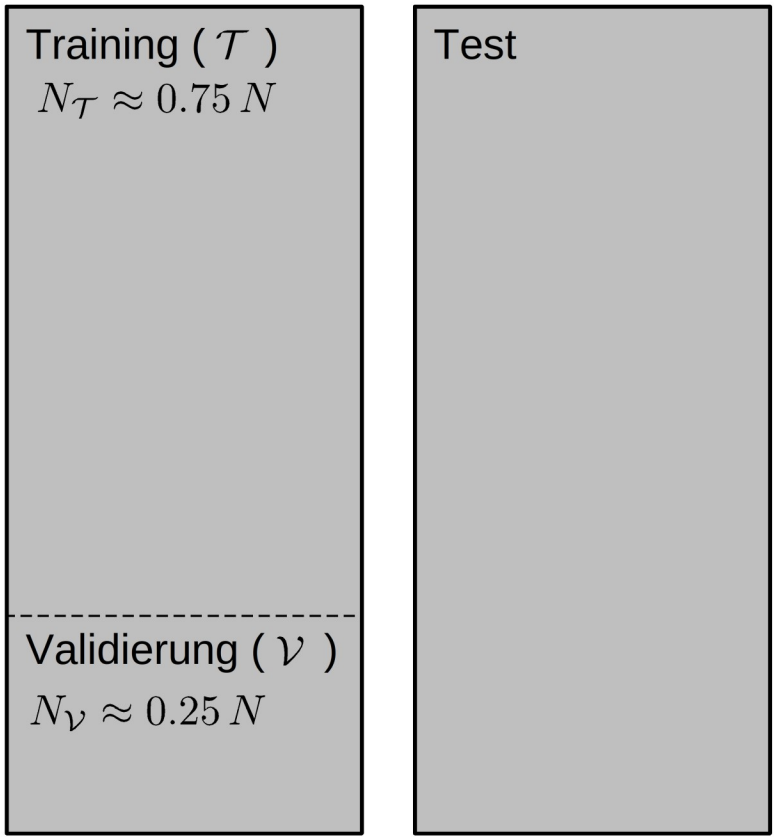
- In der Teilchenphysik verwenden wir zum Training i.a. die Datensätze unseres Untergrundmodells.



K-fache Kreuzvalidierung

- In der Teilchenphysik verwenden wir zum Training i.a. die Datensätze unseres Untergrundmodells.

- Diese in stat. unabhängige Trainings- und Testdatesätze zu trennen bedeutet den Verlust des Trainingsdatensatzes für die stat. Sigalbestimmung.

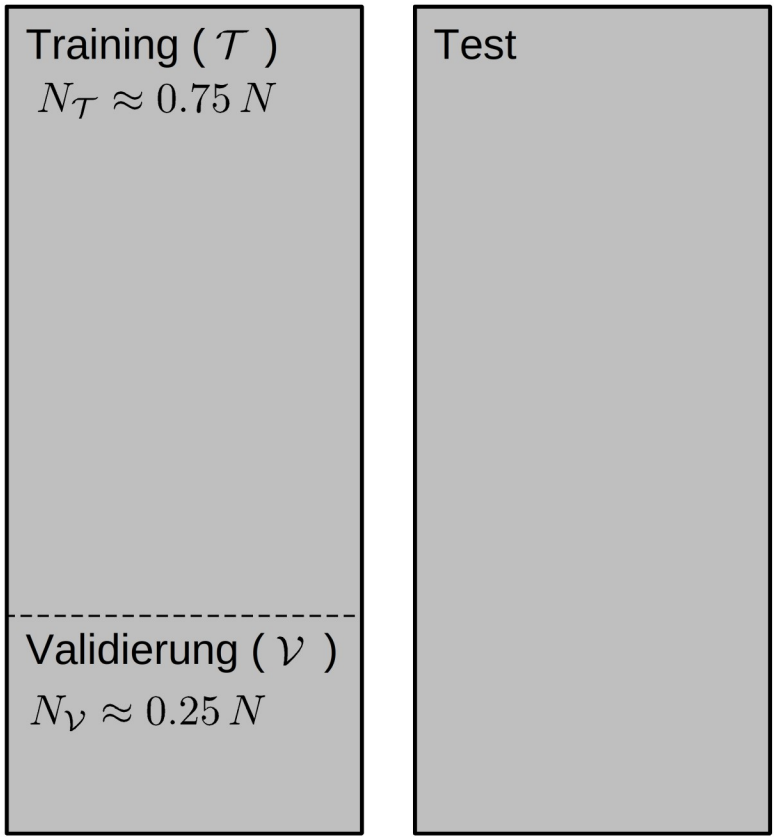


K-fache Kreuzvalidierung

- In der Teilchenphysik verwenden wir zum Training i.a. die Datensätze unseres Untergrundmodells.

- Diese in stat. unabhängige Trainings- und Testdatesätze zu trennen bedeutet den Verlust des Trainingsdatensatzes für die stat. Sigalbestimmung.

- Eine Möglichkeit stat. Unabhängigkeit beider Datensätze zu wahren und trotzdem den vollen Datensatz für die stat. Signalbestimmung nutzen zu können besteht in der k-fachen Kreuzvalidierung (*k-fold cross validation*).



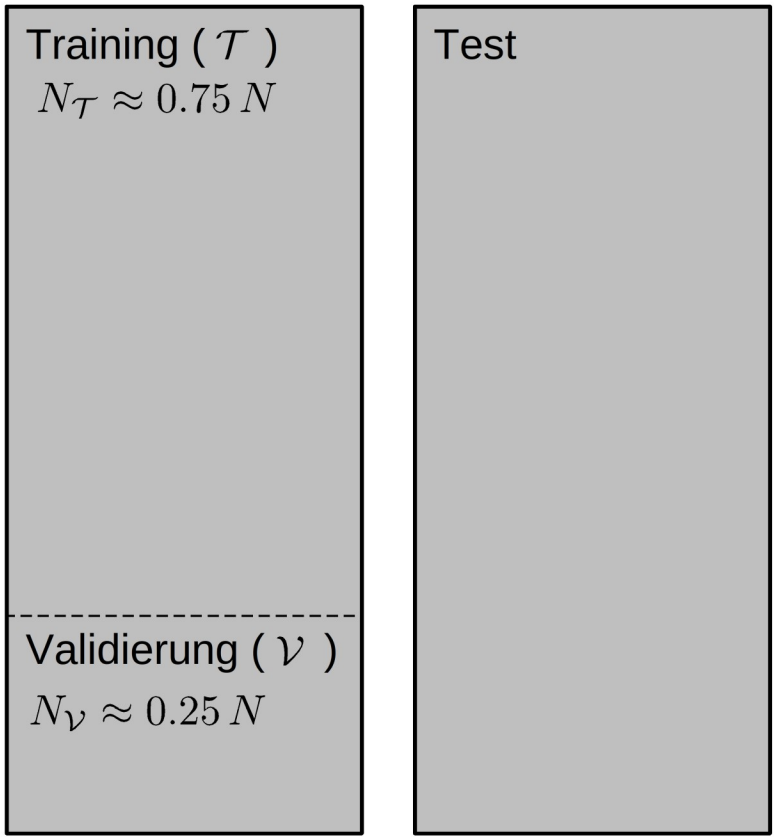
K-fache Kreuzvalidierung

- In der Teilchenphysik verwenden wir zum Training i.a. die Datensätze unseres Untergrundmodells.

- Diese in stat. unabhängige Trainings- und Testdatesätze zu trennen bedeutet den Verlust des Trainingsdatensatzes für die stat. Sigalbestimmung.

- Eine Möglichkeit stat. Unabhängigkeit beider Datensätze zu wahren und trotzdem den vollen Datensatz für die stat. Signalbestimmung nutzen zu können besteht in der k-fachen Kreuzvalidierung (*k-fold cross validation*).

- Dabei wird der Datensatz in k Teile aufgespalten und das Training k-mal durchgeführt. Anschließend werden die Ergebnisse der k Trainings addiert.



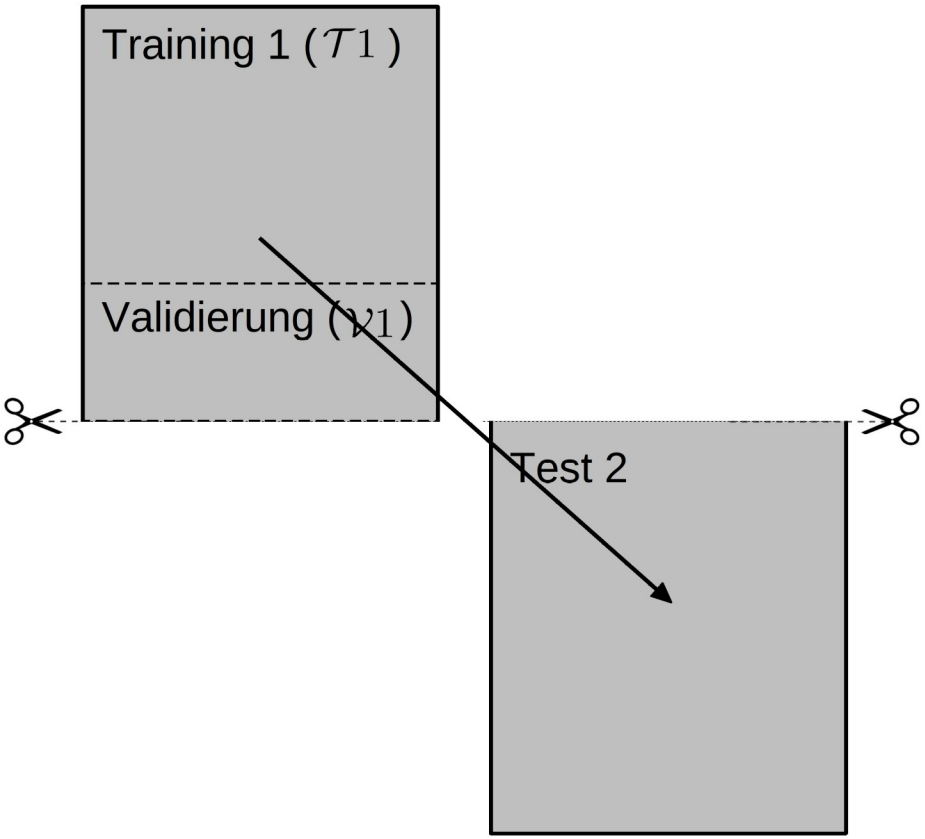
K-fache Kreuzvalidierung

- In der Teilchenphysik verwenden wir zum Training i.a. die Datensätze unseres Untergrundmodells.

- Diese in stat. unabhängige Trainings- und Testdatesätze zu trennen bedeutet den Verlust des Trainingsdatensatzes für die stat. Sigalbestimmung.

- Eine Möglichkeit stat. Unabhängigkeit beider Datensätze zu wahren und trotzdem den vollen Datensatz für die stat. Signalbestimmung nutzen zu können besteht in der k-fachen Kreuzvalidierung (*k-fold cross validation*).

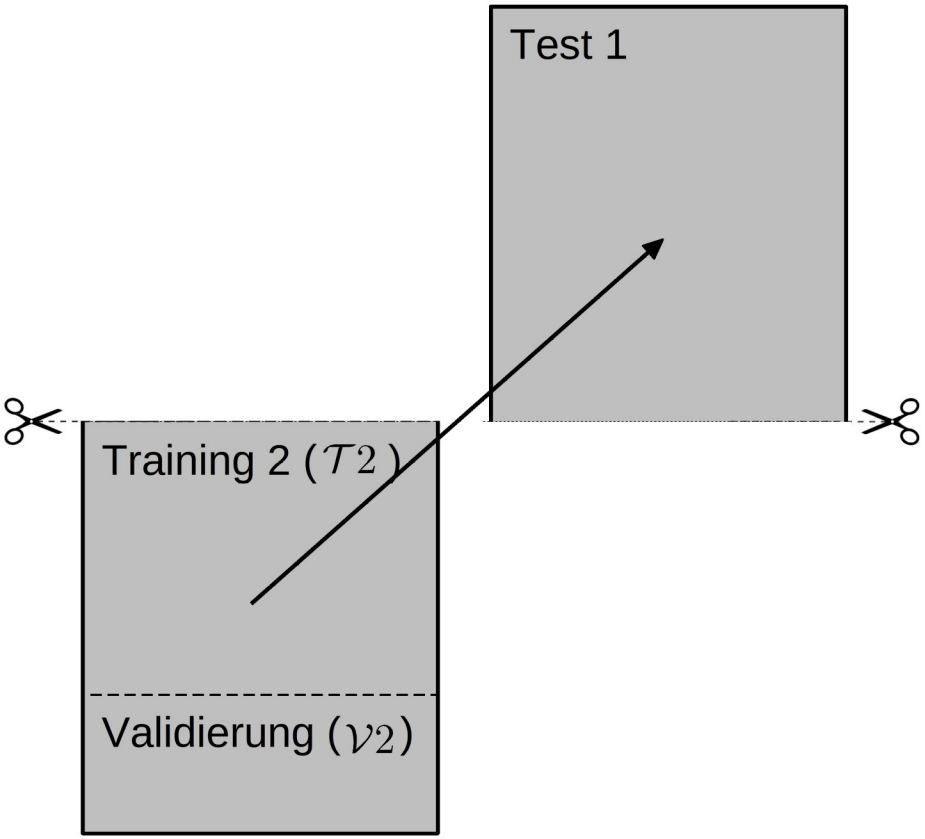
- Dabei wird der Datensatz in k Teile aufgespalten und das Training k-mal durchgeführt. Anschließend werden die Ergebnisse der k Trainings addiert.



(Hier gezeigt für *2-fold cross validation*)

K-fache Kreuzvalidierung

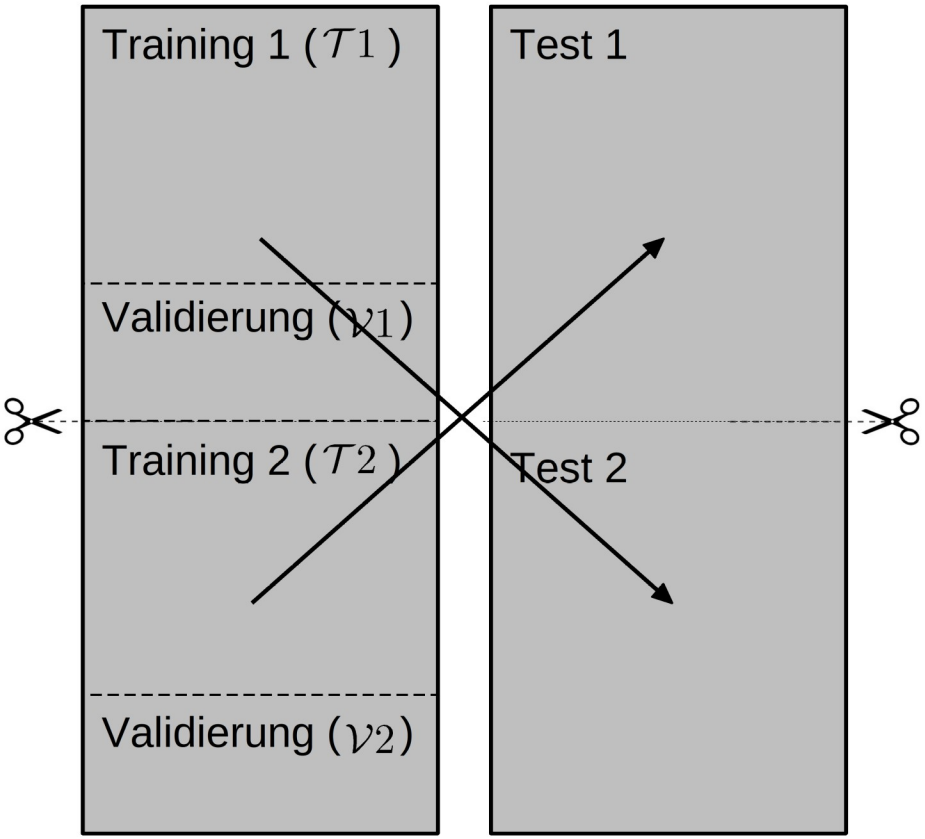
- In der Teilchenphysik verwenden wir zum Training i.a. die Datensätze unseres Untergrundmodells.
- Diese in stat. unabhängige Trainings- und Testdatesätze zu trennen bedeutet den Verlust des Trainingsdatensatzes für die stat. Sigalbestimmung.
- Eine Möglichkeit stat. Unabhängigkeit beider Datensätze zu wahren und trotzdem den vollen Datensatz für die stat. Signalbestimmung nutzen zu können besteht in der k-fachen Kreuzvalidierung (*k-fold cross validation*).
- Dabei wird der Datensatz in k Teile aufgespalten und das Training k-mal durchgeführt. Anschließend werden die Ergebnisse der k Trainings addiert.



(Hier gezeigt für *2-fold cross validation*)

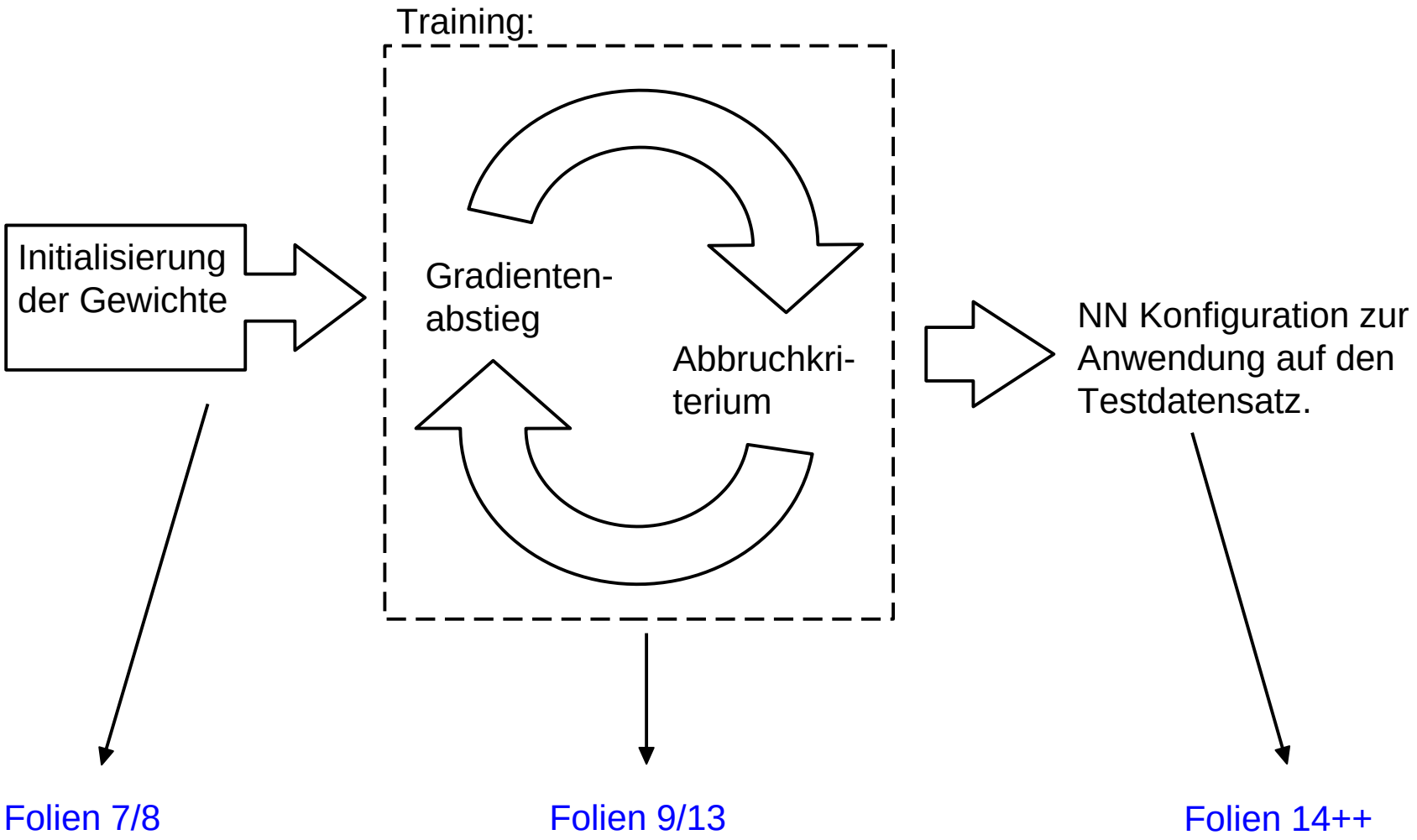
K-fache Kreuzvalidierung

- In der Teilchenphysik verwenden wir zum Training i.a. die Datensätze unseres Untergrundmodells.
- Diese in stat. unabhängige Trainings- und Testdatesätze zu trennen bedeutet den Verlust des Trainingsdatensatzes für die stat. Sigalbestimmung.
- Eine Möglichkeit stat. Unabhängigkeit beider Datensätze zu wahren und trotzdem den vollen Datensatz für die stat. Signalbestimmung nutzen zu können besteht in der k-fachen Kreuzvalidierung (*k-fold cross validation*).
- Dabei wird der Datensatz in k Teile aufgespalten und das Training k-mal durchgeführt. Anschließend werden die Ergebnisse der k Trainings addiert.



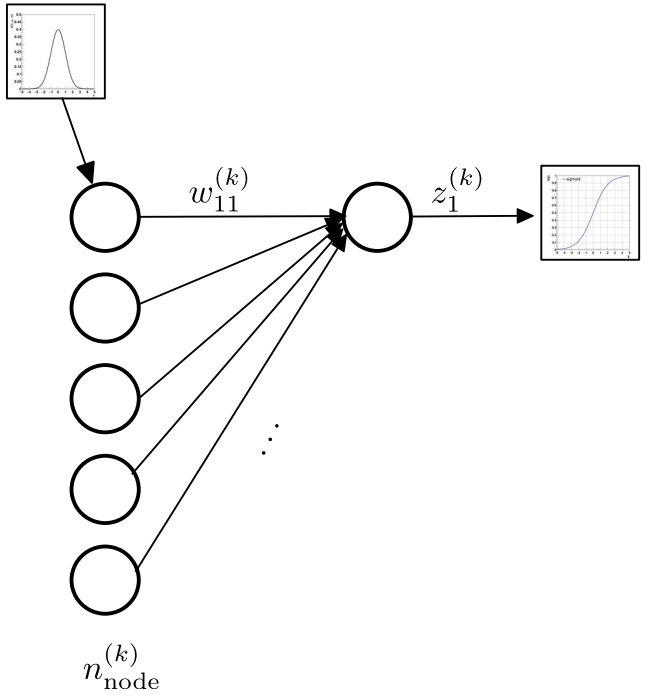
(Hier gezeigt für *2-fold cross validation*)

Trainingsablauf



Initialisierung der trainierbaren MLP Parameter

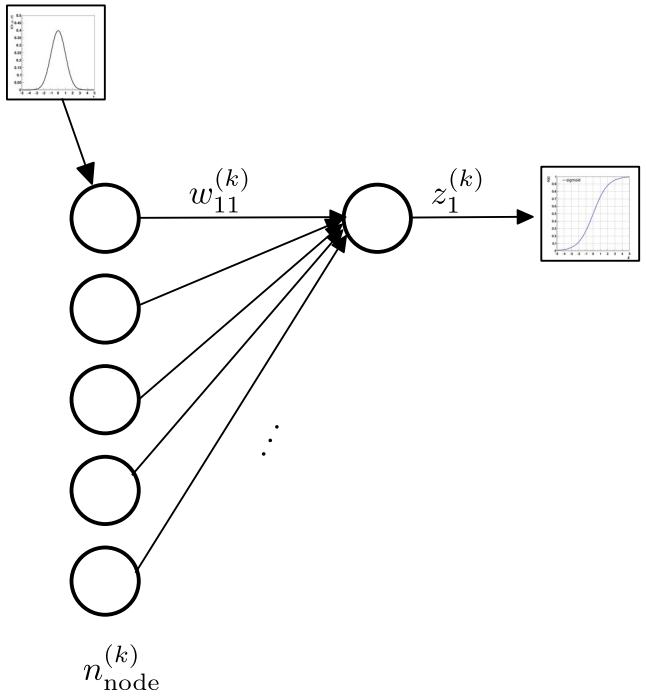
- Die Initialisierung der Schwellen (*biases*) erfolgt üblicherweise auf 0.
- Die Initialisierung der Gewichte (*weights*) erfolgt zufällg normal- oder uniform verteilt.
- Naiver Ansatz:



Initialisierung der trainierbaren MLP Parameter

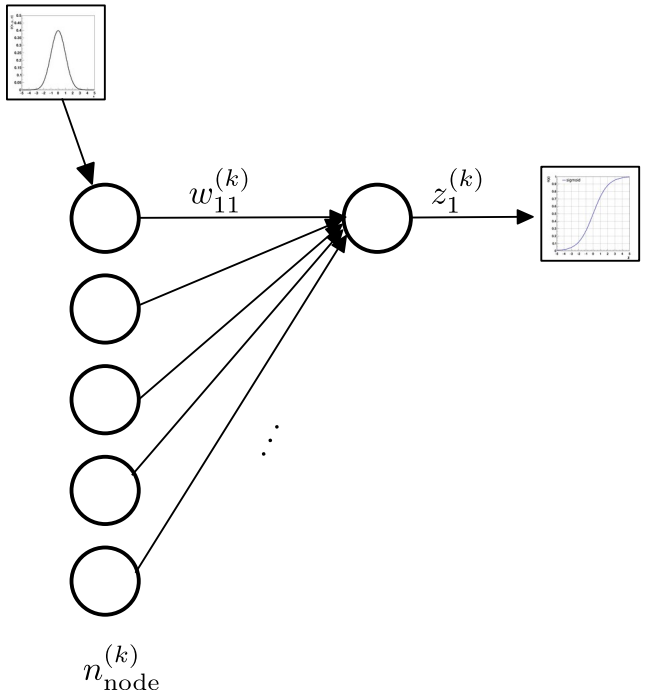
- Die Initialisierung der Schwellen (*biases*) erfolgt üblicherweise auf 0.
- Die Initialisierung der Gewichte (*weights*) erfolgt zufällg normal- oder uniform verteilt.
- Naiver Ansatz:
  - Nehme an alle Gewicht seien standardnormal-verteilt initialisiert:

$$\mu(w_{ij}^{(k)}) = 0,\ \sigma(w_{ij}^{(k)}) = 1 \quad \forall\, i,\ j,\ k$$



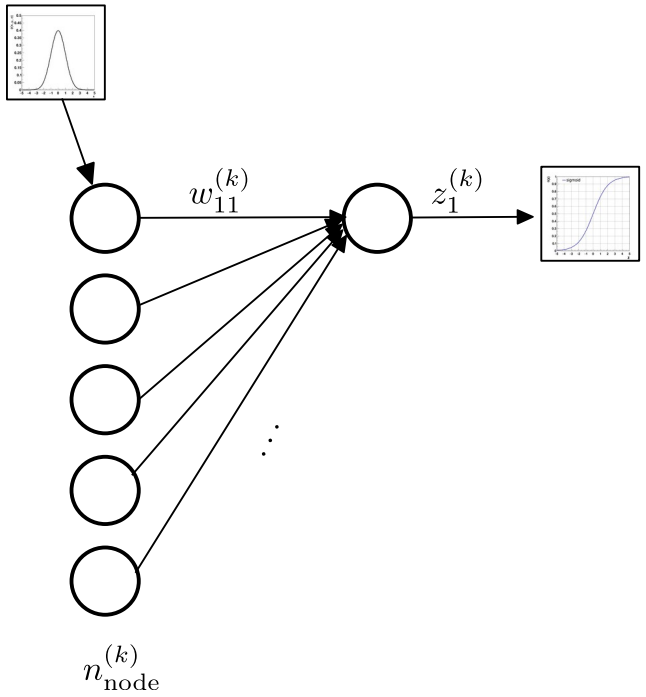
Initialisierung der trainierbaren MLP Parameter

- Die Initialisierung der Schwellen (*biases*) erfolgt üblicherweise auf 0.
- Die Initialisierung der Gewichte (*weights*) erfolgt zufällg normal- oder uniform verteilt.
- Naiver Ansatz:
  - Nehme an alle Gewicht seien standardnormal-verteilt initialisiert:
    $\mu(w_{ij}^{(k)}) = 0,\ \sigma(w_{ij}^{(k)}) = 1 \quad \forall\, i,\ j,\ k$
  - Dann ist $z_i^{(k)}$ auch normalverteilt, aber mit welcher Varianz? –



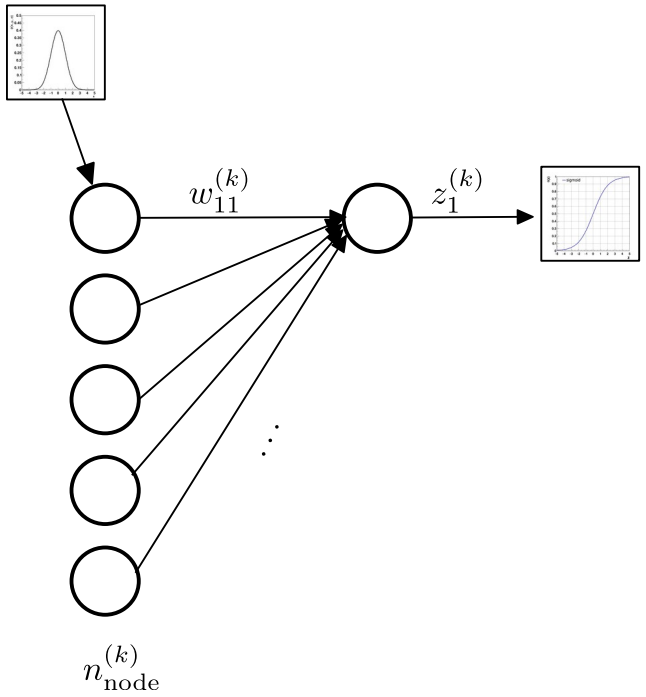
Initialisierung der trainierbaren MLP Parameter

- Die Initialisierung der Schwellen (*biases*) erfolgt üblicherweise auf 0.
- Die Initialisierung der Gewichte (*weights*) erfolgt zufällg normal- oder uniform verteilt.
- Naiver Ansatz:
  - Nehme an alle Gewicht seien standardnormal-verteilt initialisiert:
    $\mu(w_{ij}^{(k)}) = 0,\ \sigma(w_{ij}^{(k)}) = 1 \quad \forall\, i,\ j,\ k$
  - Dann ist $z_i^{(k)}$ auch normalverteilt, aber mit welcher Varianz? – $\sigma(z_i^{(k)}) = n_{\text{node}}^{(k)}$



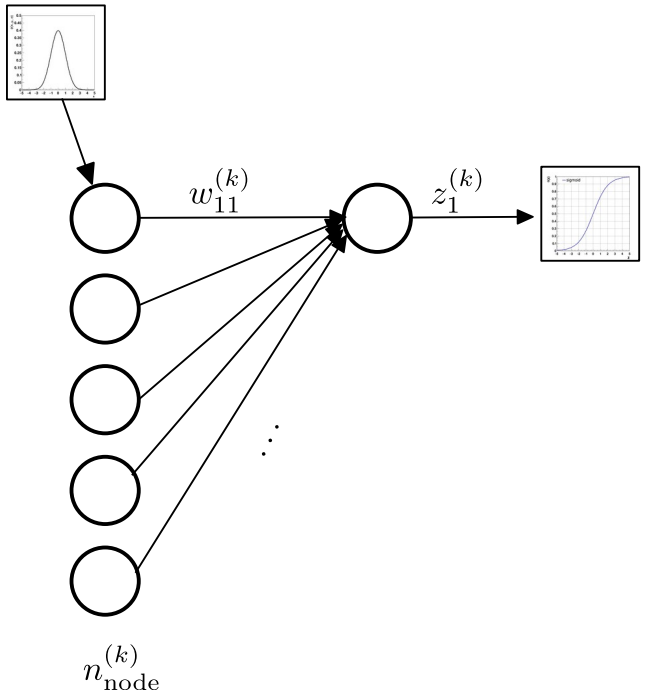
Initialisierung der trainierbaren MLP Parameter

- Die Initialisierung der Schwellen (*biases*) erfolgt üblicherweise auf 0.
- Die Initialisierung der Gewichte (*weights*) erfolgt zufällg normal- oder uniform verteilt.
- Naiver Ansatz:
  - Nehme an alle Gewicht seien standardnormal-verteilt initialisiert:
    $\mu(w_{ij}^{(k)}) = 0,\ \sigma(w_{ij}^{(k)}) = 1 \quad \forall\, i,\ j,\ k$
  - Dann ist $z_i^{(k)}$ auch normalverteilt, aber mit welcher Varianz? – $\sigma(z_i^{(k)}) = n_{\text{node}}^{(k)}$
  - Erhöhte Wahrscheinlichkeit für $|z_i^{(k)}| \gg 0$.



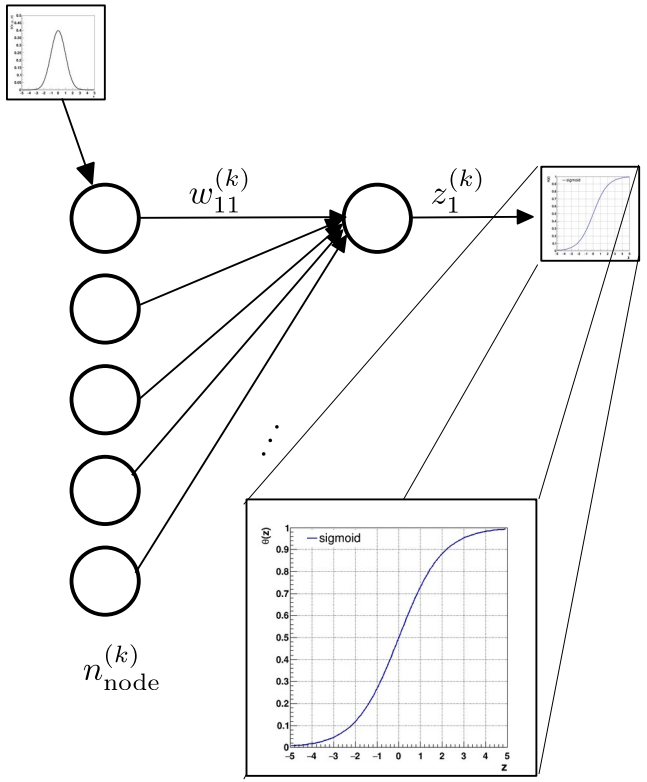
Initialisierung der trainierbaren MLP Parameter

- Die Initialisierung der Schwellen (*biases*) erfolgt üblicherweise auf 0.
- Die Initialisierung der Gewichte (*weights*) erfolgt zufällg normal- oder uniform verteilt.
- Naiver Ansatz:
  - Nehme an alle Gewicht seien standardnormal-verteilt initialisiert:
    $\mu(w_{ij}^{(k)}) = 0,\ \sigma(w_{ij}^{(k)}) = 1 \quad \forall\, i,\ j,\ k$
  - Dann ist $z_i^{(k)}$ auch normalverteilt, aber mit welcher Varianz? – $\sigma(z_i^{(k)}) = n_{\text{node}}^{(k)}$
  - Erhöhte Wahrscheinlichkeit für $|z_i^{(k)}| \gg 0$.
  - Konsequenz für $y_i^{(k)}$? –



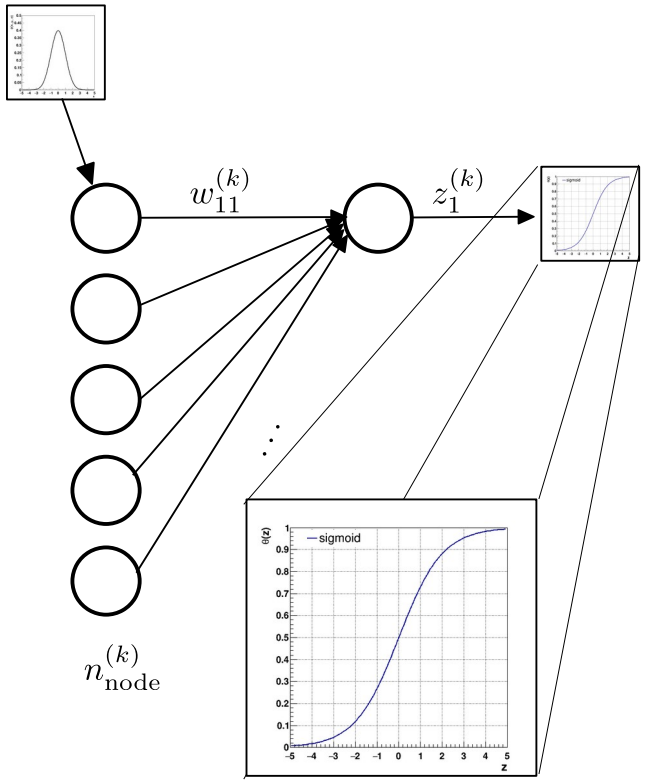
Initialisierung der trainierbaren MLP Parameter

- Die Initialisierung der Schwellen (*biases*) erfolgt üblicherweise auf 0.
- Die Initialisierung der Gewichte (*weights*) erfolgt zufällg normal- oder uniform verteilt.
- Naiver Ansatz:
  - Nehme an alle Gewicht seien standardnormal-verteilt initialisiert:
    $\mu(w_{ij}^{(k)}) = 0,\ \sigma(w_{ij}^{(k)}) = 1 \quad \forall\, i,\ j,\ k$
  - Dann ist $z_i^{(k)}$ auch normalverteilt, aber mit welcher Varianz? – $\sigma(z_i^{(k)}) = n_{\text{node}}^{(k)}$
  - Erhöhte Wahrscheinlichkeit für $|z_i^{(k)}| \gg 0$.
  - Konsequenz für $y_i^{(k)}$? – $y_i^{(k)} \to 0, 1$.



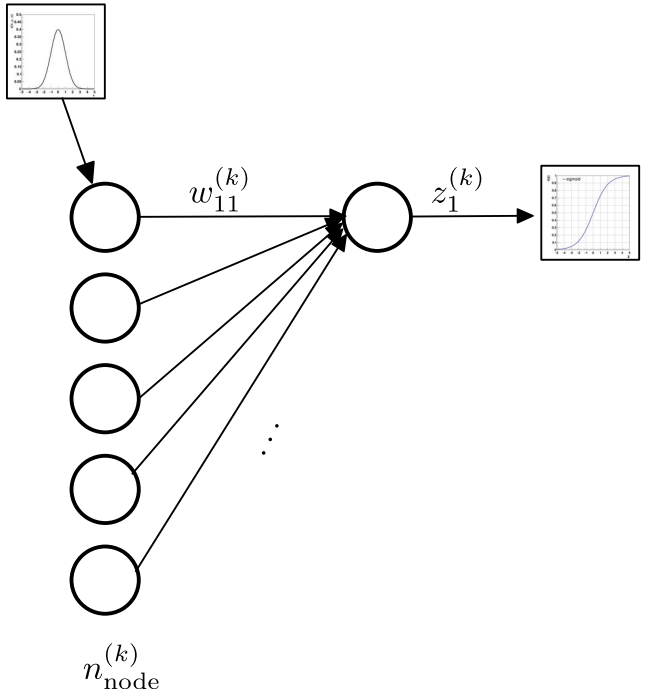
Initialisierung der trainierbaren MLP Parameter

- Die Initialisierung der Schwellen (*biases*) erfolgt üblicherweise auf 0.
- Die Initialisierung der Gewichte (*weights*) erfolgt zufällg normal- oder uniform verteilt.
- Naiver Ansatz:
  - Nehme an alle Gewicht seien standardnormal-verteilt initialisiert:
    $\mu(w_{ij}^{(k)}) = 0,\ \sigma(w_{ij}^{(k)}) = 1 \quad \forall\, i,\ j,\ k$
  - Dann ist $z_i^{(k)}$ auch normalverteilt, aber mit welcher Varianz? – $\sigma(z_i^{(k)}) = n_{\text{node}}^{(k)}$
  - Erhöhte Wahrscheinlichkeit für $|z_i^{(k)}| \gg 0$.
  - Konsequenz für $y_i^{(k)}$? – $y_i^{(k)} \to 0, 1$.
  - D.h. Knoten in folgenden Lagen tragen nicht mehr zum Informationsgewinn bei.



Glorot Initialisierung

- Diese Situation kann vermieden werden, wenn man die Gewichte wie folgt initialisiert:
  - Initialisiere Gewichte nach:
    $\mu(w_{ij}^{(k)}) = 0,\ \sigma(w_{ij}^{(k)}) = 1 \quad \forall\, i,\ j,\ k$
  - Skaliere alle Gewichte mit:
    $$w_{ij}^{(k)} \rightarrow w_{ij}^{(k)\prime} = \frac{1}{\sqrt{n_{\text{node}}^{(k)}}} w_{ij}^{(k)}$$
  - Das führt zu: $\sigma(z_i^{(k)}) = 1$
- Diese Art der Initialisierung bezeichnet man als **Glorot** oder **Xavier Initialisierung**.



Gradientenabstieg in der Praxis

- Für den Gradientenabstieg (*gradient descent, GD*) diskutieren wir im folgenden drei gängige praktische Vorgehensweisen:
  - Batch GD (*batch gradient descent, BGD*).
  - Stochastischer GD (*stochastic gradient descent, SGD*).
  - Mini-batch GD (*mini-batch gradient descent, mBGD*).

Batch gradient descent (BGD)

- Erinnerung Aktualisierungsregel:

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i)$$

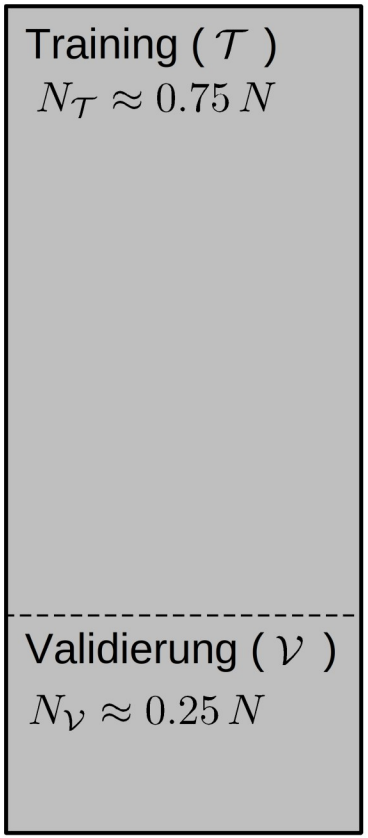
n : Länge der Stichprobe

y_i : MLP output für Beispiel i

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$



Batch gradient descent (BGD)

- Erinnerung Aktualisierungsregel:

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i)$$

n : Länge der Stichprobe

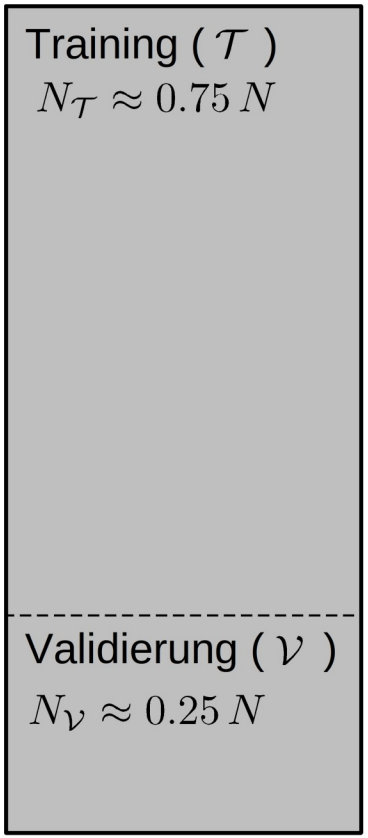
y_i : MLP output für Beispiel i

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$

- Für die Aktualisierung der Gewichte $\vec{w}^{(k)}$ evaluiere L auf $\mathcal{T}$ ($n = N_{\mathcal{T}}$).



Batch gradient descent (BGD)

- Erinnerung Aktualisierungsregel:

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i)$$

n : Länge der Stichprobe

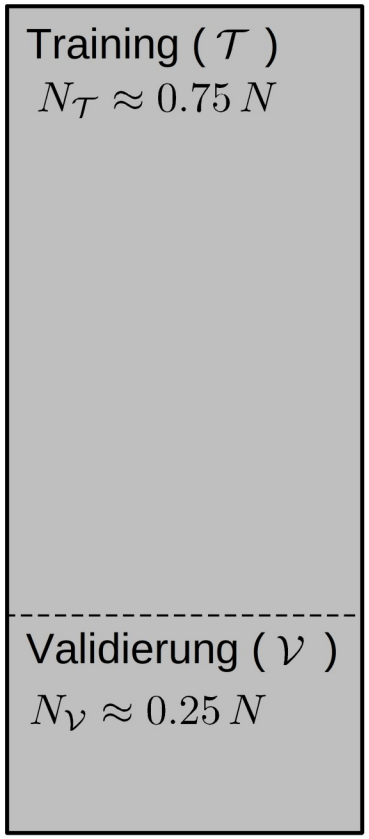
y_i : MLP output für Beispiel i

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$

- Für die Aktualisierung der Gewichte $\vec{w}^{(k)}$ evaluiere L auf $\mathcal{T}$ ($n = N_{\mathcal{T}}$).
- Nach Aktualisierung evaluiere L auf $\mathcal{V}$ ($n = N_{\mathcal{V}}$).



Stochastic gradient descent (SGD)

- Erinnerung Aktualisierungsregel:

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i)$$

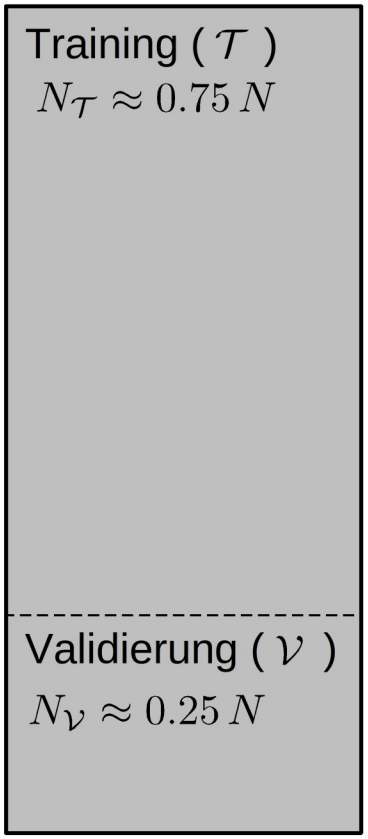
n : Länge der Stichprobe

y_i : MLP output für Beispiel i

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$



Stochastic gradient descent (SGD)

- Erinnerung Aktualisierungsregel:

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i)$$

n : Länge der Stichprobe

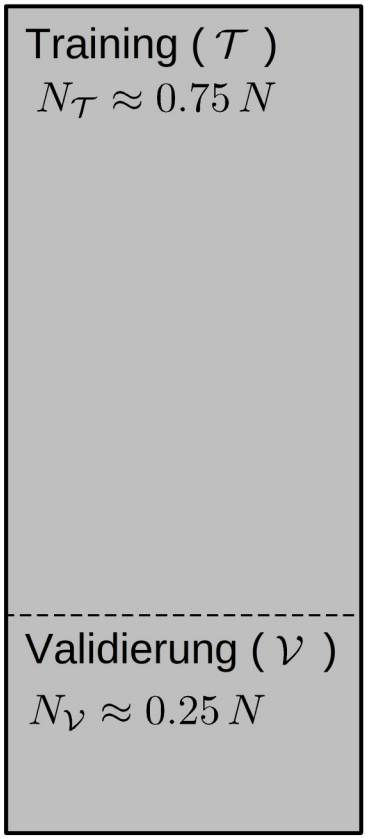
y_i : MLP output für Beispiel i

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$

- Für die Aktualisierung der Gewichte $\vec{w}^{(k)}$ evaluiere L auf einem einzelnen Beispiel aus $\mathcal{T}$ ($n = 1$).



Stochastic gradient descent (SGD)

- Erinnerung Aktualisierungsregel:

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i)$$

n : Länge der Stichprobe

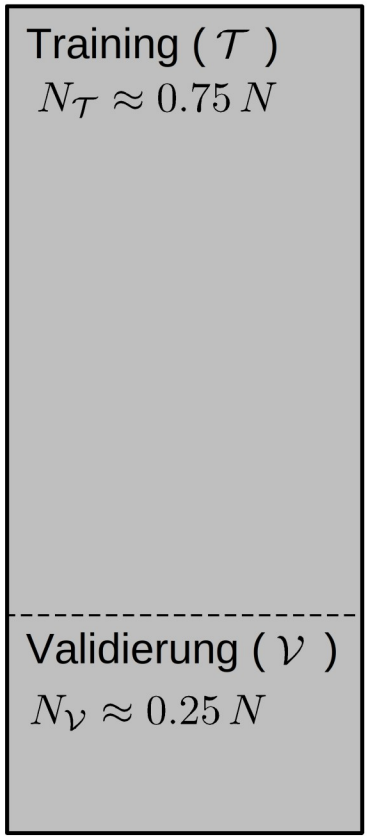
y_i : MLP output für Beispiel i

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$

- Für die Aktualisierung der Gewichte $\vec{w}^{(k)}$ evaluiere L auf einem einzelnen Beispiel aus $\mathcal{T}$ ($n = 1$).
- Nach Aktualisierung randomisiere $\mathcal{T}$.



Stochastic gradient descent (SGD)

- Erinnerung Aktualisierungsregel:

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i)$$

n : Länge der Stichprobe

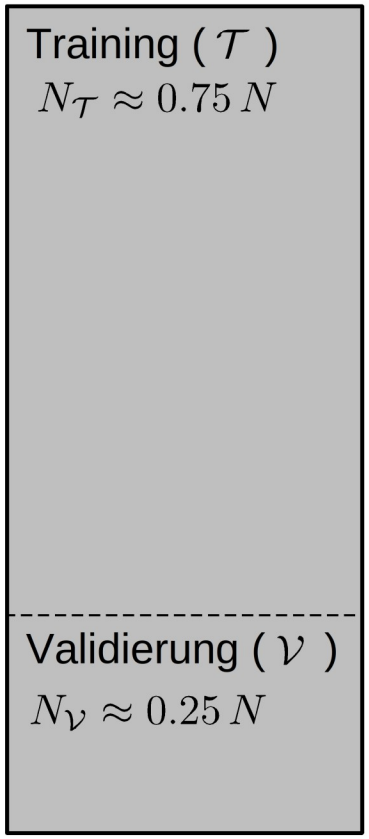
y_i : MLP output für Beispiel i

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$

- Für die Aktualisierung der Gewichte $\vec{w}^{(k)}$ evaluiere L auf einem einzelnen Beispiel aus $\mathcal{T}$ ($n = 1$).
- Nach Aktualisierung randomisiere $\mathcal{T}$.
- Evaluiere L auf $\mathcal{V}$ ($n = N_{\mathcal{V}}$).



Mini-batch gradient descent (mBGD)

- Erinnerung Aktualisierungsregel:

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i)$$

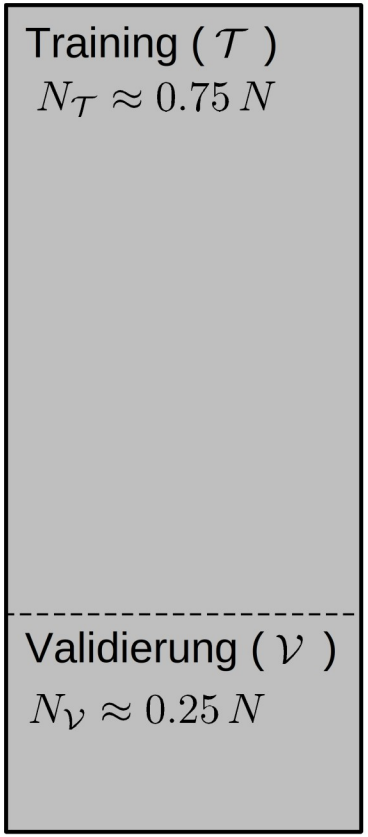
n : Länge der Stichprobe

y_i : MLP output für Beispiel i

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$



Mini-batch gradent descent (mBGD)

- Erinnerung Aktualisierungsregel:

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i) \qquad \vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L, \quad \eta > 0$$

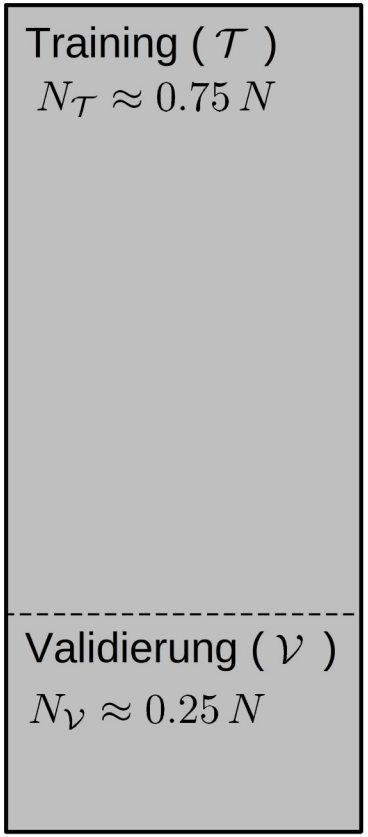
n : Länge der Stichprobe

y_i : MLP output für Beispiel i

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$

- Für die Aktualisierung der Gewichte $\vec{w}^{(k)}$ evaluiere L auf einem mini-batch aus $\mathcal{T}$ ($n = N_{\text{batch}} < N_{\mathcal{T}}$).



Mini-batch gradent descent (mBGD)

- Erinnerung Aktualisierungsregel:

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i) \qquad \vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L, \quad \eta > 0$$

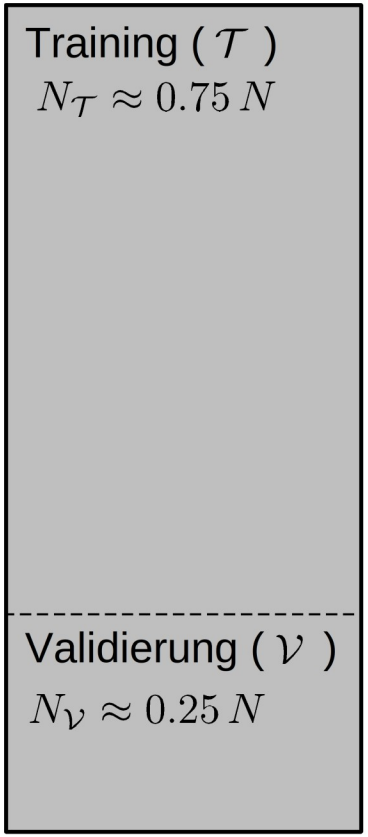
n : Länge der Stichprobe

y_i : MLP output für Beispiel i

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$

- Für die Aktualisierung der Gewichte $\vec{w}^{(k)}$ evaluiere L auf einem mini-batch aus $\mathcal{T}$ ($n = N_{\text{batch}} < N_{\mathcal{T}}$).
- Nach Aktualisierung randomisiere $\mathcal{T}$.



Mini-batch gradent descent (mBGD)

- Erinnerung Aktualisierungsregel:

$$L(\{p_i\},\{y_i\}) = -\sum_{i=1}^{n} p_i \log(y_i)$$

n : Länge der Stichprobe

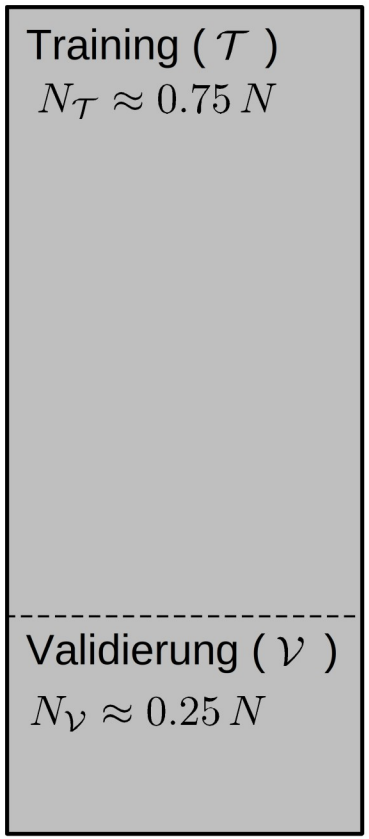
y_i : MLP output für Beispiel i

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$

- Für die Aktualisierung der Gewichte $\vec{w}^{(k)}$ evaluiere L auf einem mini-batch aus $\mathcal{T}$ ($n = N_{\text{batch}} < N_{\mathcal{T}}$).
- Nach Aktualisierung randomisiere $\mathcal{T}$.
- Evaluiere L auf $\mathcal{V}$ ($n = N_{\mathcal{V}}$).



Diskussion Gradientenabstieg

- Jedes mal, wenn L auf $\mathcal{V}$ evaluiert wird bezeichnet man das als **Epoche** (*epoche*).

Diskussion Gradientenabstieg

- Jedes mal, wenn L auf $\mathcal{V}$ evaluiert wird bezeichnet man das als **Epoche** (*epoche*).

- Dies geschieht üblicherweise nachdem $\mathcal{T}$ (grundsätzlich) 1x durchlaufen wurde. Man kann eine Epoche aber auch durch eine fest vorgegebene Anzahl an Gradientenabstiegen definieren.

Diskussion Gradientenabstieg

- Jedes mal, wenn L auf $\mathcal{V}$ evaluiert wird bezeichnet man das als **Epoche** (*epoche*).

- Dies geschieht üblicherweise nachdem $\mathcal{T}$ (grundsätzlich) 1x durchlaufen wurde. Man kann eine Epoche aber auch durch eine fest vorgegebene Anzahl an Gradientenabstiegen definieren.

- SGD und mBGD sind klassische Boostrap Verfahren. Sie sind ferner auf „on the fly“ wachsende Trainingsdatensätze anwendbar.

Diskussion Gradientenabstieg

- Jedes mal, wenn L auf $\mathcal{V}$ evaluiert wird bezeichnet man das als **Epoche** (*epoche*).

- Dies geschieht üblicherweise nachdem $\mathcal{T}$ (grundsätzlich) 1x durchlaufen wurde. Man kann eine Epoche aber auch durch eine fest vorgegebene Anzahl an Gradientenabstiegen definieren.

- SGD und mBGD sind klassische Boostrap Verfahren. Sie sind ferner auf „on the fly“ wachsende Trainingsdatensätze anwendbar.

- Das heutzutage beliebteste praktische Verfahren zum Gradientenabstieg ist der mBGD.

Diskussion Gradientenabstieg

- Jedes mal, wenn L auf $\mathcal{V}$ evaluiert wird bezeichnet man das als **Epoche** (*epoche*).

- Dies geschieht üblicherweise nachdem $\mathcal{T}$ (grundsätzlich) 1x durchlaufen wurde. Man kann eine Epoche aber auch durch eine fest vorgegebene Anzahl an Gradientenabstiegen definieren.

- SGD und mBGD sind klassische Boostrap Verfahren. Sie sind ferner auf „on the fly“ wachsende Trainingsdatensätze anwendbar.

- Das heutzutage beliebteste praktische Verfahren zum Gradientenabstieg ist der mBGD.

  - Die *batch*-Größen variieren zwischen 50 und einigen 100 Beispielen abhängig davon, was man sich unter dem Askpekt der CPU/GPU *performance* leisten kann.

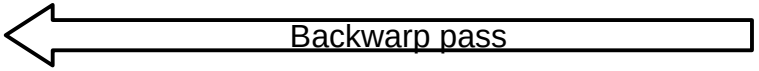
Herausforderungen während des Trainings

- Im folgenden werden wir einige Herausforderungen beim MLP training diskutieren und wie man ihnen i.a. begegnet:
  - Exploding/vanishing gadient
  - Batch normalization
  - Generalisierbarkeit des Trainings
  - Regularisierungsmethoden (*early stopping, dropout*, L1/L2 Regularisierung)

Exploding/Vanishing gradient

- $\nabla_{\vec{w}^{(k)}} L$ ergibt sich aus dem Produkt von Ableitungen nachfolgender Lagen (*backward pass*).

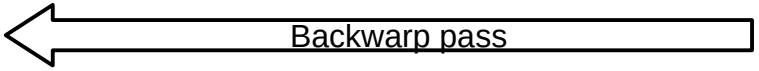
$$\frac{\mathrm{d}L}{\mathrm{d}w_{ij}^{(1)}} = \underbrace{\sum_i \frac{\mathrm{d}z_j^{(1)}}{\mathrm{d}w_{ij}^{(1)}} \cdot \frac{\mathrm{d}y_1^{(1)}}{\mathrm{d}z_j^{(1)}}}_{\text{Lage 1}} \cdot \underbrace{\sum_i \frac{\mathrm{d}z_j^{(2)}}{\mathrm{d}w_{ij}^{(2)}} \cdot \frac{\mathrm{d}y_j^{(2)}}{\mathrm{d}z_j^{(2)}}}_{\text{Lage 2}} \cdot \ldots \cdot \underbrace{\sum_i \frac{\mathrm{d}z_j^{(k)}}{\mathrm{d}w_{ij}^{(k)}} \cdot \frac{\mathrm{d}y_i^{(k)}}{\mathrm{d}z_i^{(k)}}}_{\text{Lage } k} \cdot \frac{\mathrm{d}L}{\mathrm{d}y_i^{(k)}}$$



Exploding/Vanishing gradient

- $\nabla_{\vec{w}^{(k)}} L$ ergibt sich aus dem Produkt von Ableitungen nachfolgender Lagen (*backward pass*).

$$\frac{\mathrm{d}L}{\mathrm{d}w_{ij}^{(1)}} = \underbrace{\sum_i \frac{\mathrm{d}z_j^{(1)}}{\mathrm{d}w_{ij}^{(1)}} \cdot \frac{\mathrm{d}y_1^{(1)}}{\mathrm{d}z_j^{(1)}}}_{\text{Lage 1}} \cdot \underbrace{\sum_i \frac{\mathrm{d}z_j^{(2)}}{\mathrm{d}w_{ij}^{(2)}} \cdot \frac{\mathrm{d}y_j^{(2)}}{\mathrm{d}z_j^{(2)}}}_{\text{Lage 2}} \cdot \ldots \cdot \underbrace{\sum_i \frac{\mathrm{d}z_j^{(k)}}{\mathrm{d}w_{ij}^{(k)}} \cdot \frac{\mathrm{d}y_i^{(k)}}{\mathrm{d}z_i^{(k)}}}_{\text{Lage } k} \cdot \frac{\mathrm{d}L}{\mathrm{d}y_i^{(k)}}$$

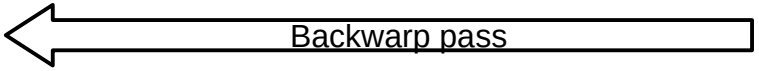


- Insbesondere für frühe Lagen (wie z.B. die erste) kann dieses Produkt sehr lang werden.

Exploding/Vanishing gradient

- $\nabla_{\vec{w}^{(k)}} L$ ergibt sich aus dem Produkt von Ableitungen nachfolgender Lagen (*backward pass*).

$$\frac{\mathrm{d}L}{\mathrm{d}w_{ij}^{(1)}} = \underbrace{\sum_i \frac{\mathrm{d}z_j^{(1)}}{\mathrm{d}w_{ij}^{(1)}} \cdot \frac{\mathrm{d}y_1^{(1)}}{\mathrm{d}z_j^{(1)}}}_{\text{Lage 1}} \cdot \underbrace{\sum_i \frac{\mathrm{d}z_j^{(2)}}{\mathrm{d}w_{ij}^{(2)}} \cdot \frac{\mathrm{d}y_j^{(2)}}{\mathrm{d}z_j^{(2)}}}_{\text{Lage 2}} \cdot \ldots \cdot \underbrace{\sum_i \frac{\mathrm{d}z_j^{(k)}}{\mathrm{d}w_{ij}^{(k)}} \cdot \frac{\mathrm{d}y_i^{(k)}}{\mathrm{d}z_i^{(k)}}}_{\text{Lage } k} \cdot \frac{\mathrm{d}L}{\mathrm{d}y_i^{(k)}}$$

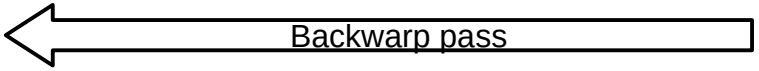


- Insbesondere für frühe Lagen (wie z.B. die erste) kann dieses Produkt sehr lang werden.
  - Es kann passieren, dass jeder der Faktoren < 1 ist $\rightarrow \nabla_{\vec{w}^{(k)}} L \rightarrow 0$, d.h. $w_{ij}^{(k)}$ wird nicht mehr aktualisiert, schlechte Konvergenz (*vanishing gradient*).

Exploding/Vanishing gradient

- $\nabla_{\vec{w}^{(k)}} L$ ergibt sich aus dem Produkt von Ableitungen nachfolgender Lagen (*backward pass*).

$$\frac{\mathrm{d}L}{\mathrm{d}w_{ij}^{(1)}} = \underbrace{\sum_i \frac{\mathrm{d}z_j^{(1)}}{\mathrm{d}w_{ij}^{(1)}} \cdot \frac{\mathrm{d}y_1^{(1)}}{\mathrm{d}z_j^{(1)}}}_{\text{Lage 1}} \cdot \underbrace{\sum_i \frac{\mathrm{d}z_j^{(2)}}{\mathrm{d}w_{ij}^{(2)}} \cdot \frac{\mathrm{d}y_j^{(2)}}{\mathrm{d}z_j^{(2)}}}_{\text{Lage 2}} \cdot \ldots \cdot \underbrace{\sum_i \frac{\mathrm{d}z_j^{(k)}}{\mathrm{d}w_{ij}^{(k)}} \cdot \frac{\mathrm{d}y_i^{(k)}}{\mathrm{d}z_i^{(k)}}}_{\text{Lage } k} \cdot \frac{\mathrm{d}L}{\mathrm{d}y_i^{(k)}}$$

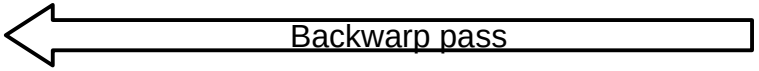


- Insbesondere für frühe Lagen (wie z.B. die erste) kann dieses Produkt sehr lang werden.
  - Es kann passieren, dass jeder der Faktoren < 1 ist $\rightarrow \nabla_{\vec{w}^{(k)}} L \rightarrow 0$, d.h. $w_{ij}^{(k)}$ wird nicht mehr aktualisiert, schlechte Konvergenz (*vanishing gradient*).
  - Es kann passieren, dass jeder der Faktoren $\gg 1$ ist $\rightarrow \nabla_{\vec{w}^{(k)}} L \rightarrow \infty$, erratische Sprünge in $w_{ij}^{(k)}$(*exploiding gradient*).

Exploding/Vanishing gradient

- $\nabla_{\vec{w}^{(k)}} L$ ergibt sich aus dem Produkt von Ableitungen nachfolgender Lagen (*backward pass*).

$$\frac{\mathrm{d}L}{\mathrm{d}w_{ij}^{(1)}} = \underbrace{\sum_i \frac{\mathrm{d}z_j^{(1)}}{\mathrm{d}w_{ij}^{(1)}} \cdot \frac{\mathrm{d}y_1^{(1)}}{\mathrm{d}z_j^{(1)}}}_{\text{Lage 1}} \cdot \underbrace{\sum_i \frac{\mathrm{d}z_j^{(2)}}{\mathrm{d}w_{ij}^{(2)}} \cdot \frac{\mathrm{d}y_j^{(2)}}{\mathrm{d}z_j^{(2)}}}_{\text{Lage 2}} \cdot \ldots \cdot \underbrace{\sum_i \frac{\mathrm{d}z_j^{(k)}}{\mathrm{d}w_{ij}^{(k)}} \cdot \frac{\mathrm{d}y_i^{(k)}}{\mathrm{d}z_i^{(k)}}}_{\text{Lage } k} \cdot \frac{\mathrm{d}L}{\mathrm{d}y_i^{(k)}}$$



- Insbesondere für frühe Lagen (wie z.B. die erste) kann dieses Produkt sehr lang werden.
  - Es kann passieren, dass jeder der Faktoren < 1 ist $\rightarrow \nabla_{\vec{w}^{(k)}} L \rightarrow 0$, d.h. $w_{ij}^{(k)}$ wird nicht mehr aktualisiert, schlechte Konvergenz (*vanishing gradient*).
  - Es kann passieren, dass jeder der Faktoren $\gg 1$ ist $\rightarrow \nabla_{\vec{w}^{(k)}} L \rightarrow \infty$, erratische Sprünge in $w_{ij}^{(k)}$(*exploiding gradient*).
  - Man fasst diesen Problemkomplex unter dem Begriff instabile Gradienten (*unstable gradients*) zusammen.

Initialisierung und *feature* Standartisierung

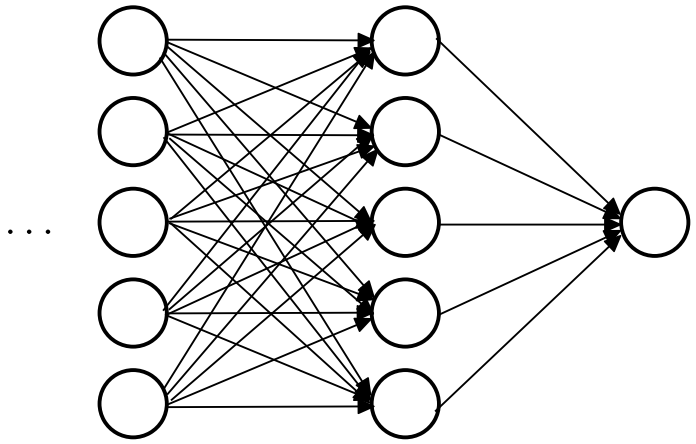
- Probleme mit instabilen Gradienten begegnet man durch Glorot Initialisierung und Standartisierung der *input features*:

$$z \to z' = \frac{z - \mu_z}{\sigma_z}$$

  - *Input features* mit beliebig und potentiell stark variierenden Skalen werden auf eine Standardskala abgebildet.

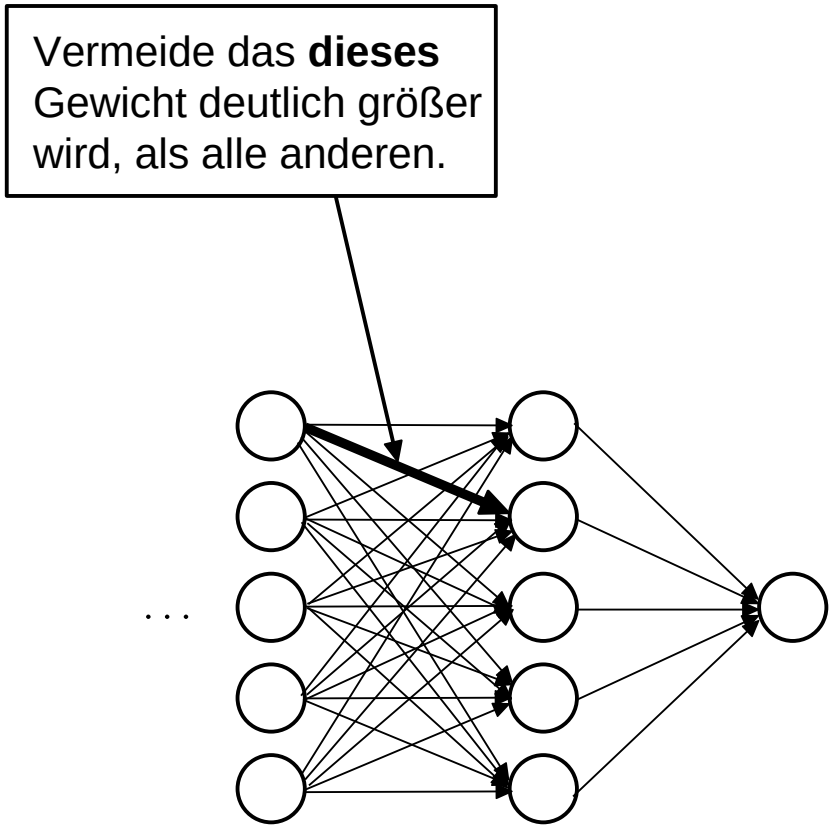
Batch Normierung

- Standardtisierung der outputs der einzelnen Lagen hilft auch während des Trainings Probleme mit unausgewogenen Gewichten zu vermeiden.



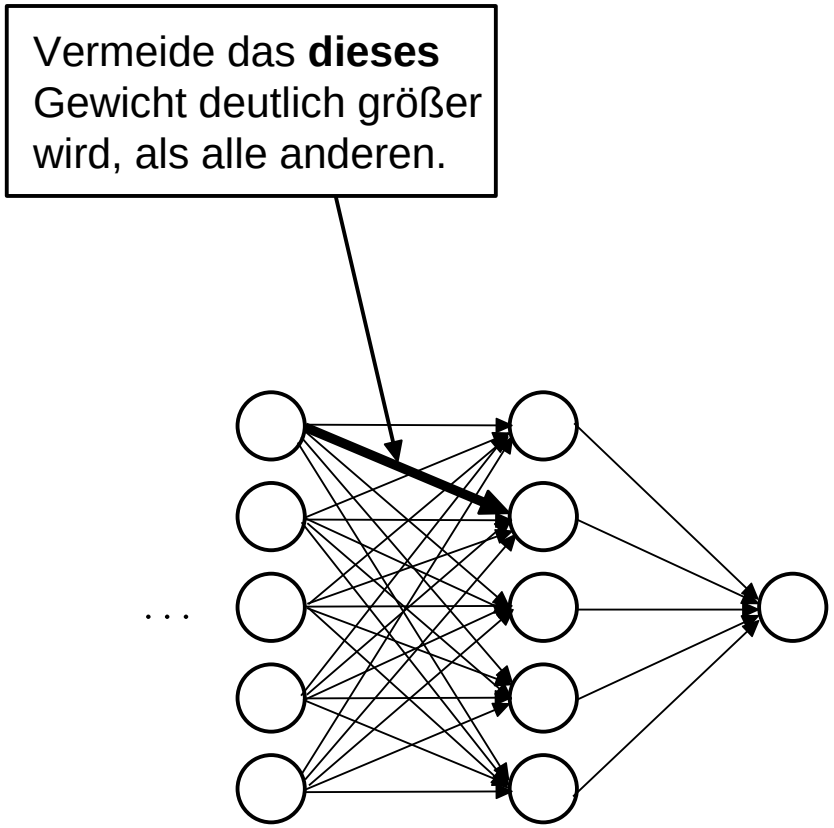
Batch Normierung

- Standardtisierung der outputs der einzelnen Lagen hilft auch während des Trainings Probleme mit unausgewogenen Gewichten zu vermeiden.



Batch Normierung

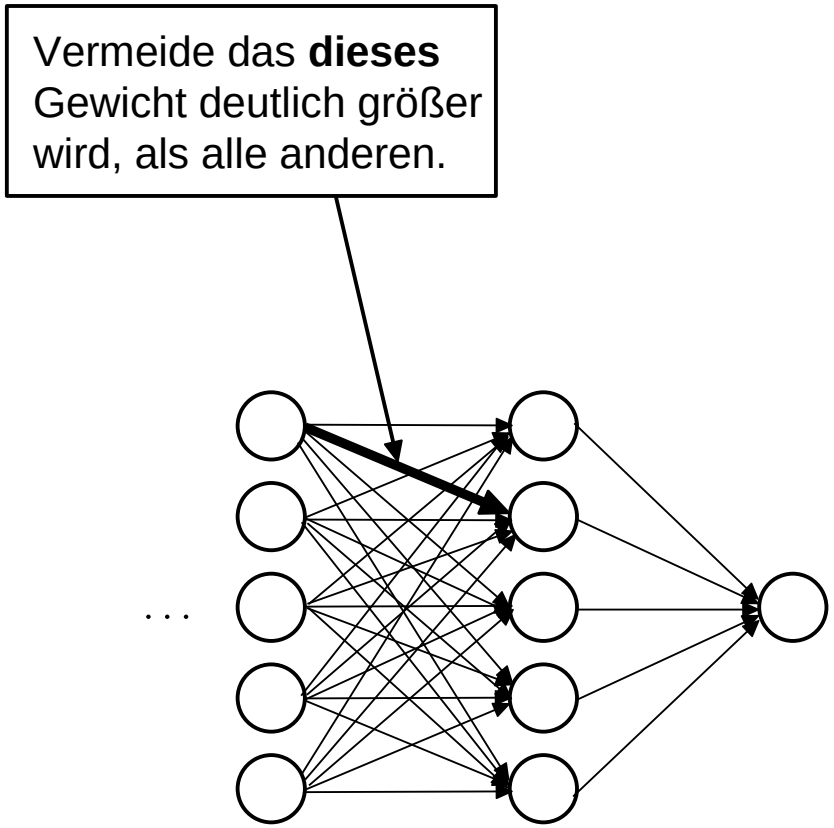
- Standardtisierung der outputs der einzelnen Lagen hilft auch während des Trainings Probleme mit unausgewogenen Gewichten zu vermeiden.



- Vorgehensweise:

Batch Normierung

- Standardtisierung der outputs der einzelnen Lagen hilft auch während des Trainings Probleme mit unausgewogenen Gewichten zu vermeiden.

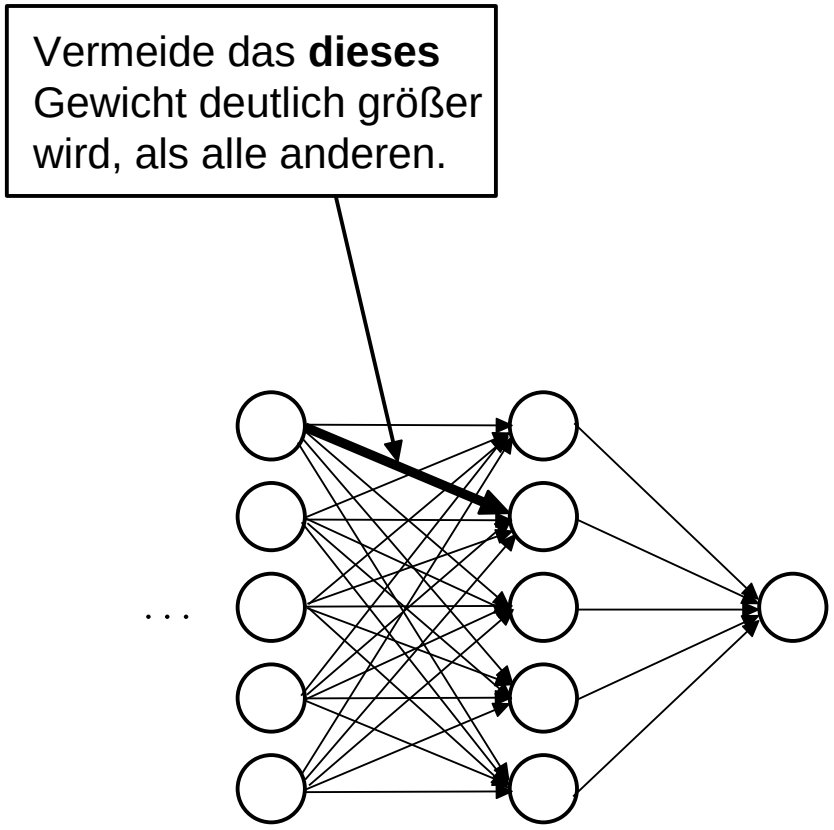


- Vorgehensweise:
  - Standardtisiere output der Lage (k):

$$y \rightarrow y' = \frac{y - \mu_y}{\sigma_y}$$

Batch Normierung

- Standardtisierung der outputs der einzelnen Lagen hilft auch während des Trainings Probleme mit unausgewogenen Gewichten zu vermeiden.



- Vorgehensweise:
  - Standardtisiere output der Lage (k):

$$y \rightarrow y' = \frac{y - \mu_y}{\sigma_y}$$

  - Strecke und schiebe das Resultat auf einen beliebigen Parameterunterraum:

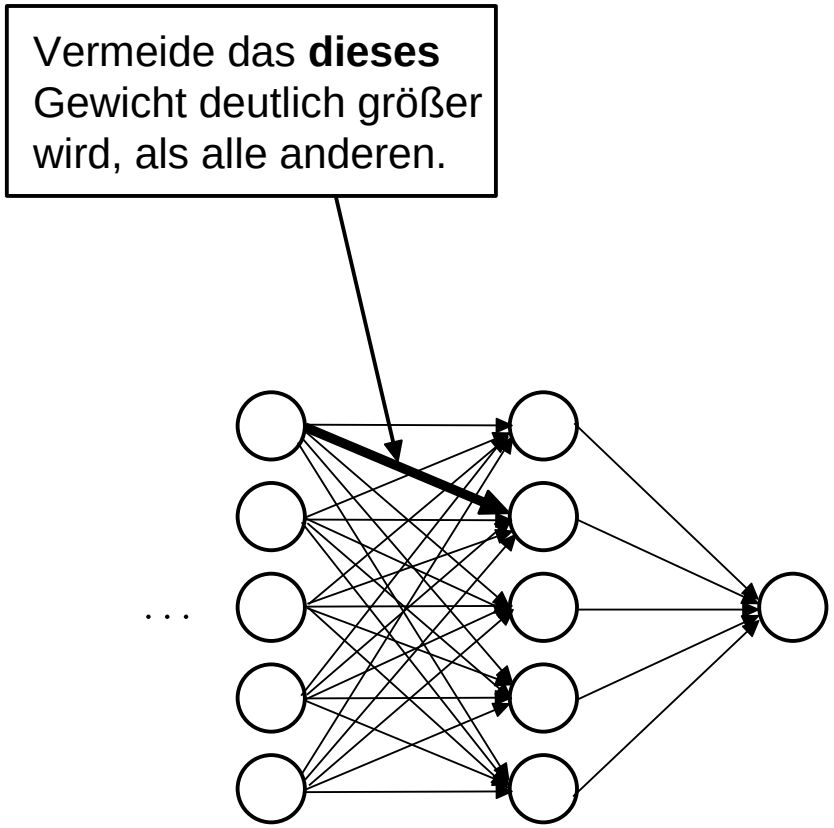
$$y' \rightarrow y'' = (g \cdot y') + b$$

g, b: beliebige trainierbare Parameter.

Die Bedeutung von g und b ist, es dem MLP zu ermöglichen immer auf der gleichen Untermanigfaltigkeit der $\{w_{ij}^{(k)}\}$ zu operieren.

Batch Normierung

- Standardtisierung der outputs der einzelnen Lagen hilft auch während des Trainings Probleme mit unausgewogenen Gewichten zu vermeiden.



- Vorgehensweise:
  - Standardtisiere output der Lage (k):

    $$y \to y' = \frac{y - \mu_y}{\sigma_y}$$

  - Strecke und schiebe das Resultat auf einen beliebigen Parameterunterraum:

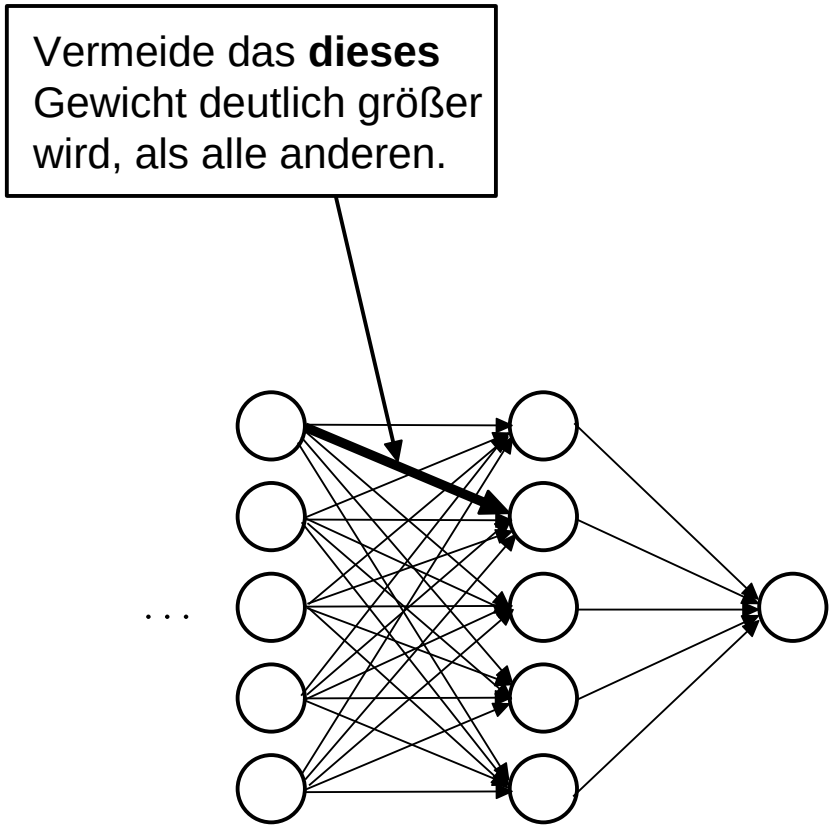
    $$y' \to y'' = (g \cdot y') + b$$

    g, b: beliebige trainierbare Parameter.

  - Woher erhalte ich μ_y und σ_y ? –

Batch Normierung

- Standardtisierung der outputs der einzelnen Lagen hilft auch während des Trainings Probleme mit unausgewogenen Gewichten zu vermeiden.



- Vorgehensweise:
  - Standardtisiere output der Lage (k):

    $$y \to y' = \frac{y - \mu_y}{\sigma_y}$$

  - Strecke und schiebe das Resultat auf einen beliebigen Parameterunterraum:

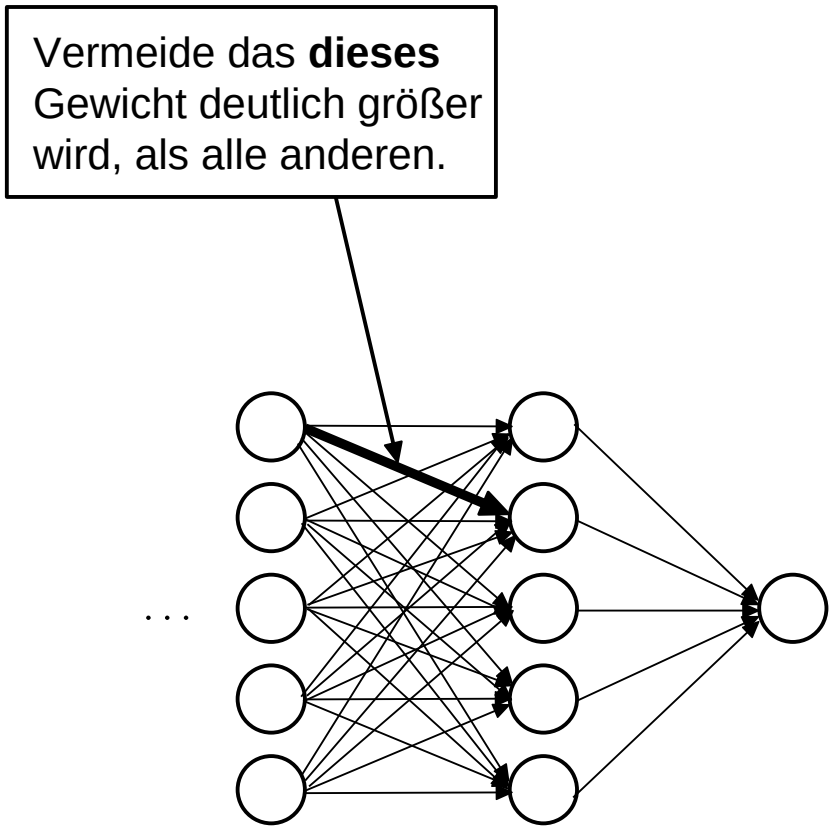
    $$y' \to y'' = (g \cdot y') + b$$

    g, b: beliebige trainierbare Parameter.

  - Woher erhalte ich μ_y und σ_y ? – Diese Parameter können z.B. aus einzelnen Aktualisierungsschritten der Gewichte innerhalb einer (mini-)batch bestimmt werden.

Batch Normierung

- Standardtisierung der outputs der einzelnen Lagen hilft auch während des Trainings Probleme mit unausgewogenen Gewichten zu vermeiden.



- Vorgehensweise:
  - Standardtisiere output der Lage (k):

    $$y \to y' = \frac{y - \mu_y}{\sigma_y}$$

  - Strecke und schiebe das Resultat auf einen beliebigen Parameterunterraum:

    $$y' \to y'' = (g \cdot y') + b$$

    g, b: beliebige trainierbare Parameter.

  - Woher erhalte ich μ_y und σ_y ? – Diese Parameter können z.B. aus einzelnen Aktualisierungsschritten der Gewichte innerhalb einer (mini-)batch bestimmt we

Um ein MLP mit batch Normierung auf einen Testdatensatz anwenden zu können halte μ_y und σ_y auf dem gesamten Testdatensatz vor.

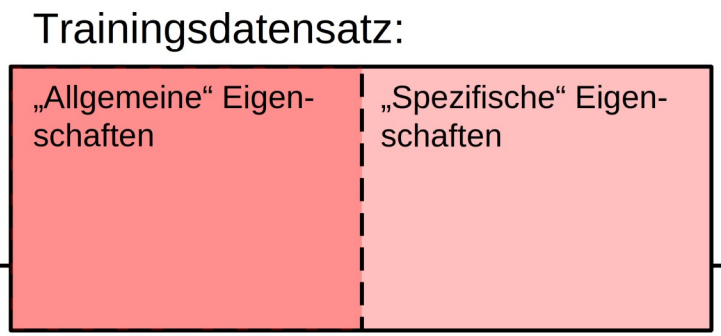
Was lernt das MLP?!?

- **Erinnerung**: Im Allgemeien nehmen wir an, dass unserem Klassifikations-/Regressionsproblem eine unbekannte Wahrheit unterliegt.

Was lernt das MLP?!?

- **Erinnerung**: Im Allgemeien nehmen wir an, dass unserem Klassifikations-/Regressionsproblem eine unbekannte Wahrheit unterliegt.

- Grundproblem der **Generalisierung**:

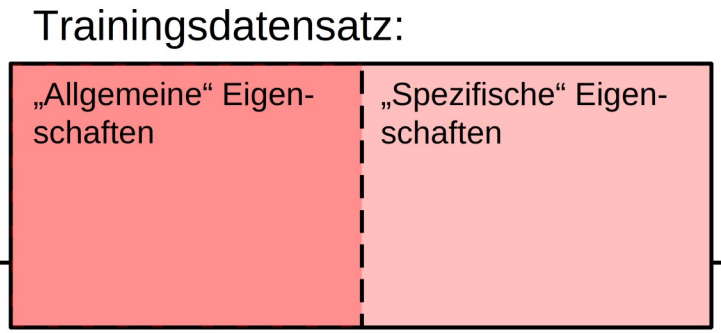


Was lernt das MLP?!?

- **Erinnerung**: Im Allgemeien nehmen wir an, dass unserem Klassifikations-/Regressionsproblem eine unbekannte Wahrheit unterliegt.

- Grundproblem der **Generalisierung**:

  - Die Wahrheit ist unbekannt! Auch der Trainingsdatensatz ist nur eine Stichprobe aus einer angenommenen wahren Grundgesamtheit.



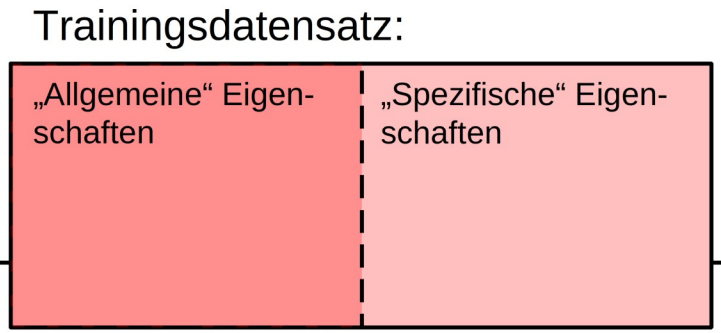
Was lernt das MLP?!?

- **Erinnerung**: Im Allgemeien nehmen wir an, dass unserem Klassifikations-/Regressionsproblem eine unbekannte Wahrheit unterliegt.

- Grundproblem der **Generalisierung**:

  - Die Wahrheit ist unbekannt! Auch der Trainingsdatensatz ist nur eine Stichprobe aus einer angenommenen wahren Grundgesamtheit.

  - Als Stichprobe unterliegt der Trainingsdatensatz statistichen Fluktuationen (→ Varianz).



Was lernt das MLP?!?

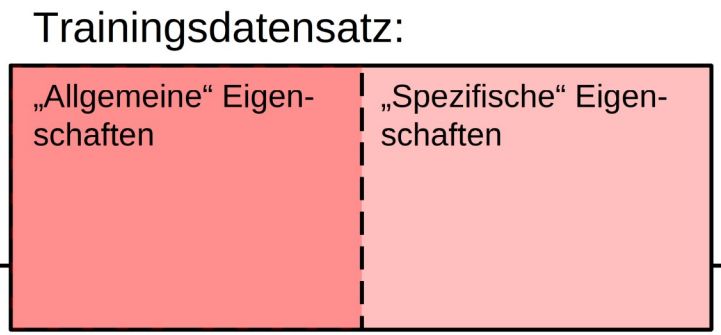
- **Erinnerung**: Im Allgemeien nehmen wir an, dass unserem Klassifikations-/Regressionsproblem eine unbekannte Wahrheit unterliegt.

- Grundproblem der **Generalisierung**:

  - Die Wahrheit ist unbekannt! Auch der Trainingsdatensatz ist nur eine Stichprobe aus einer angenommenen wahren Grundgesamtheit.

  - Als Stichprobe unterliegt der Trainingsdatensatz statistichen Fluktuationen (→ Varianz).

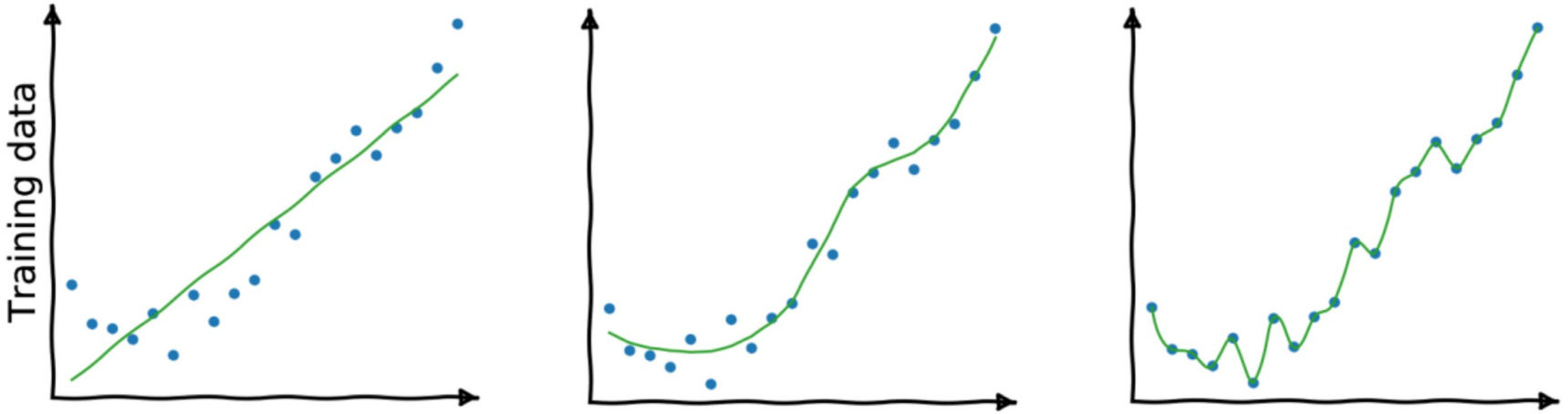
  - Zusätzlich kann der Trainingsdatensatz sich grundsätzlich von der Grundgesamtheit unterscheiden (→ Verzerrung/*bias*).



Identifikation allgemeiner Eigenschaften

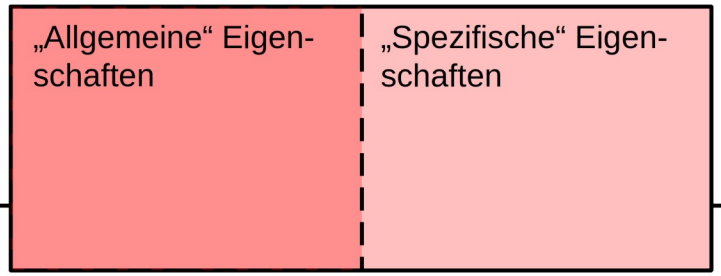
- (Wie) kann man allgemeine von spezifischen Eigenschaften des Trainingsdatensatzes unterscheiden?

- **Beispiel**: Dieser Trainingsdatensatz ist durch die blauen Punkte dargestellt. Was ist noch „allgemeiner Trend“ und was ist Fluktuation und damit spezifisch für den Trainingsdatensatz?



Offensichtliche Verbindung zum Problem des Overfitting/Overtraining.

Trainingsdatensatz:



Training und Validierung

- Dem Problem der Generalisierbarkeit des MLP Modells nach Abschluss des Trainings begegnet man durch Verwendung eines Validierungsdatensatzes (siehe Folie 4):

Training und Validierung

- Dem Problem der Generalisierbarkeit des MLP Modells nach Abschluss des Trainings begegnet man durch Verwendung eines Validierungsdatensatzes (siehe Folie 4):

- Wenn das MLP Modell überwiegend unverzerrte allgemeine Eigenschaften der Grundgesamtheit abbildet ist zu erwarten, dass es den (stat. unabhängigen) Validierungsdatensatz „ähnlich gut“ beschreibt.

Training und Validierung

- Dem Problem der Generalisierbarkeit des MLP Modells nach Abschluss des Trainings begegnet man durch Verwendung eines Validierungsdatensatzes (siehe Folie 4):

- Wenn das MLP Modell überwiegend unverzerrte allgemeine Eigenschaften der Grundgesamtheit abbildet ist zu erwarten, dass es den (stat. unabhängigen) Validierungsdatensatz „ähnlich gut“ beschreibt.

- Eine offensichtliche Art Konsistenz mit einem Referenzdatensatz zu quantifizieren ist über die Verlustfunktion.

Training und Validierung

- Dem Problem der Generalisierbarkeit des MLP Modells nach Abschluss des Trainings begegnet man durch Verwendung eines Validierungsdatensatzes (siehe Folie 4):

- Wenn das MLP Modell überwiegend unverzerrte allgemeine Eigenschaften der Grundgesamtheit abbildet ist zu erwarten, dass es den (stat. unabhängigen) Validierungsdatensatz „ähnlich gut“ beschreibt.

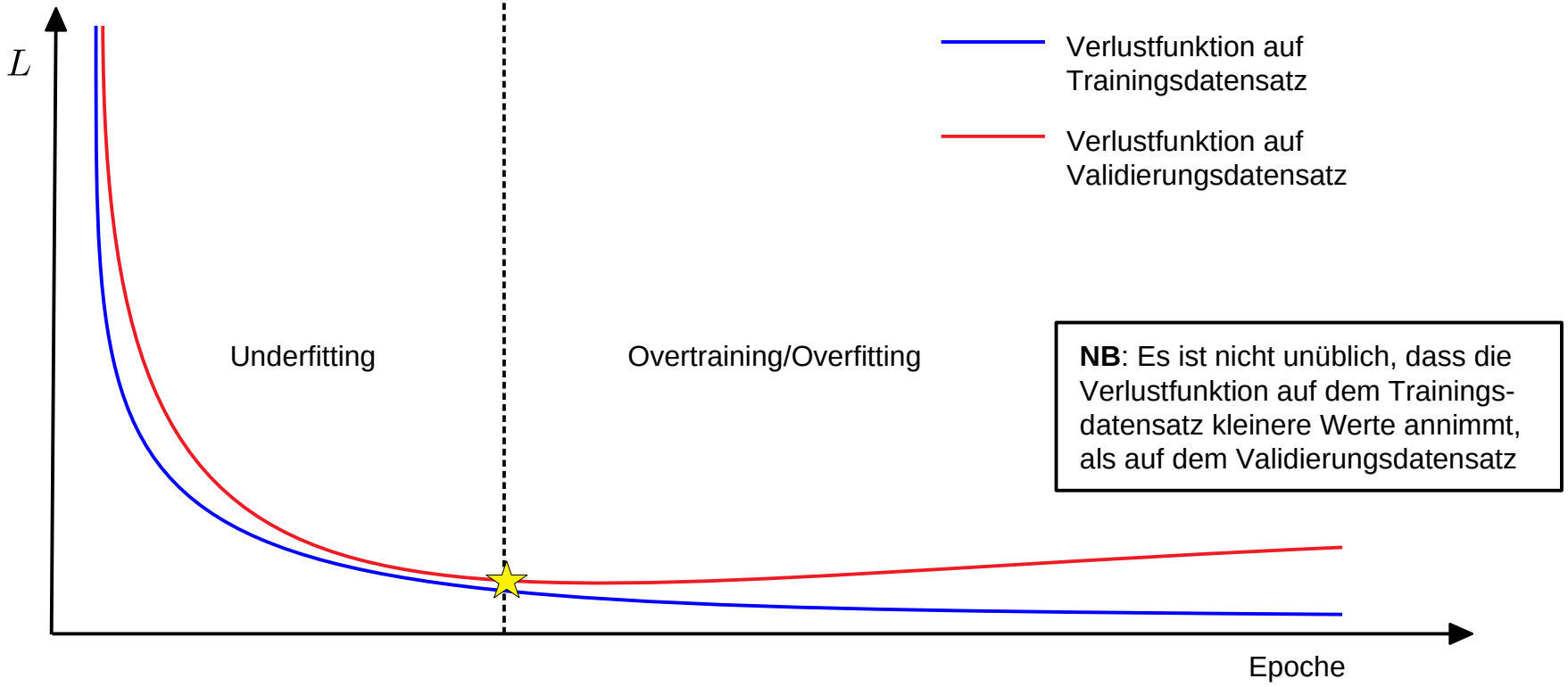
- Eine offensichtliche Art Konsistenz mit einem Referenzdatensatz zu quantifizieren ist über die Verlustfunktion.

- **NB**: Dies ist nur eine Art das zu tun. Was man früher ebenfalls oft gemacht hat, ist die MLP output Funktion $y(\{x_k\}, \{w_j\})$ ausgewertet auf dem Trainings- und einem Validierungsdatensatz mit Hilfe eines Kolmogorow-Smirnov Tests zu vergleichen.

Lernkurve

- Typischerweise sinkt die Verlustfunktion mit zunehmender Anzahl an Epochen mit typischem Konvergenzverhalten auf dem Trainingsdatensatz.

- Auf dem Validierungsdatensatz steigt die Verlustfunktion nach einer bestimmten Anzahl an Epochen wieder an.

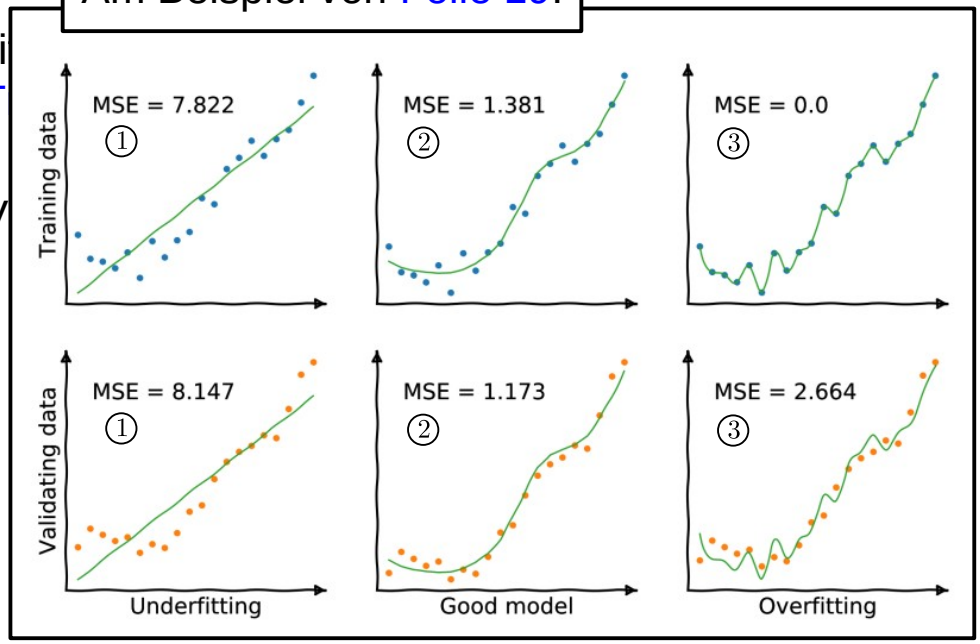


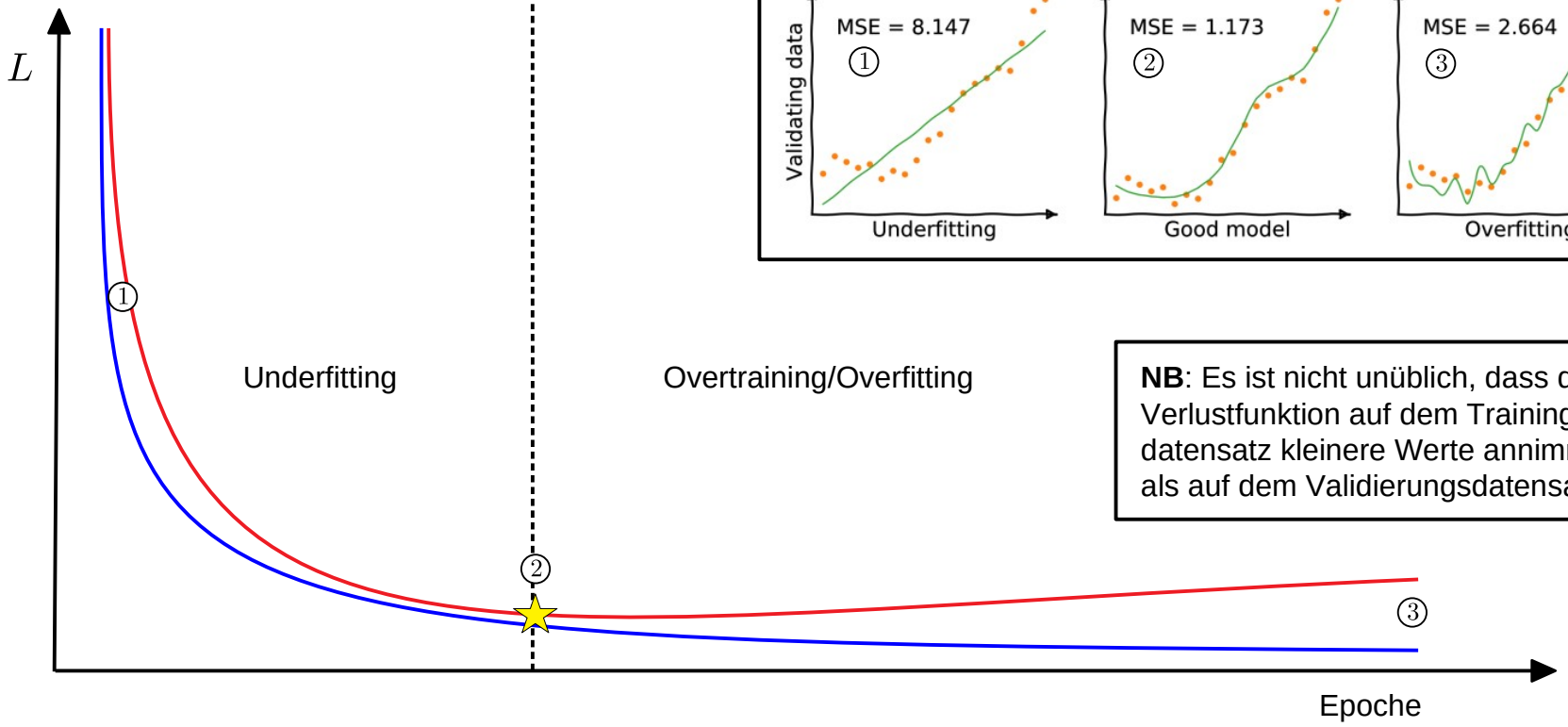
NB: Es ist nicht unüblich, dass die Verlustfunktion auf dem Trainings-datensatz kleinere Werte annimmt, als auf dem Validierungsdatensatz

Lernkurve

- Typischerweise sinkt die Verlustfunktion mi
  typischem Konvergenzverhalten auf dem T
- Auf dem Validierungsdatensatz steigt die V
  Epochen wieder an.

Am Beispiel von Folie 19:





NB: Es ist nicht unüblich, dass die Verlustfunktion auf dem Trainings-datensatz kleinere Werte annimmt, als auf dem Validierungsdatensatz

Early stopping

- Diese Betrachtungen motivieren eine Strategie, um auf sehr einfache Weise grundsätzliche Generalisierbarkeit des MLP Modells sicherzustellen:

Early stopping

- Diese Betrachtungen motivieren eine Strategie, um auf sehr einfache Weise grundsätzliche Generalisierbarkeit des MLP Modells sicherzustellen:
  - Evaluiere L nach jeder Epoche (siehe Folien 9–12) auf $\mathcal{V}$.

Early stopping

- Diese Betrachtungen motivieren eine Strategie, um auf sehr einfache Weise grundsätzliche Generalisierbarkeit des MLP Modells sicherzustellen:
  - Evaluiere L nach jeder Epoche (siehe Folien 9–12) auf $\mathcal{V}$.
  - Wenn L ausgewertet auf $\mathcal{V}$ nach einer vorgegebenen Verweildauer (*latency*) nicht mehr abnimmt beende das Training.

Early stopping

- Diese Betrachtungen motivieren eine Strategie, um auf sehr einfache Weise grundsätzliche Generalisierbarkeit des MLP Modells sicherzustellen:
  - Evaluiere L nach jeder Epoche (siehe Folien 9–12) auf $\mathcal{V}$.
  - Wenn L ausgewertet auf $\mathcal{V}$ nach einer vorgegebenen Verweildauer (*latency*) nicht mehr abnimmt beende das Training.
  - Ein solches Vorgehen bezeichnet man als *early stopping*. Hier ist es in seiner einfachsten Ausprägung beschrieben.

Diskussion der Generalisierbarkeit

- Da die Optimierung des MLP auf Stichproben basiert, ist das auf dem Trainingsdatensatz erreichte Minimum im Bestfall nur mit dem Minimum auf dem Validierungsdatensatz konsistent, d.h. die Erwartungswerte der Abschätzungen auf beiden Datensätzen stimmen im Rahmen ihrer Varianz überein.

Diskussion der Generalisierbarkeit

- Da die Optimierung des MLP auf Stichproben basiert, ist das auf dem Trainingsdatensatz erreichte Minimum im Bestfall nur mit dem Minimum auf dem Validierungsdatensatz konsistent, d.h. die Erwartungswerte der Abschätzungen auf beiden Datensätzen stimmen im Rahmen ihrer Varianz überein.

- Erreicht Ihr Trainingssetup Konsistenz mit dem Vaidierungsdatensatz spricht man von guter Generalisierbarkeit des Modells (*generalisation property*).

Diskussion der Generalisierbarkeit

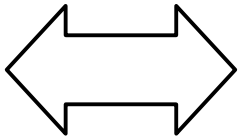
- Da die Optimierung des MLP auf Stichproben basiert, ist das auf dem Trainingsdatensatz erreichte Minimum im Bestfall nur mit dem Minimum auf dem Validierungsdatensatz konsistent, d.h. die Erwartungswerte der Abschätzungen auf beiden Datensätzen stimmen im Rahmen ihrer Varianz überein.

- Erreicht Ihr Trainingssetup Konsistenz mit dem Vaidierungsdatensatz spricht man von guter Generalisierbarkeit des Modells (*generalisation property*).

- Hat Ihr MLP schlechte Generalisierungseigenschaften ist es nutzlos.

Entfaltung vs. MLP Generalisierbarkeit

- Die Diskussion der Generalisierbarkeit im maschinellen Lernen hat eine offensichtliche Entsprechung zum „inversen Problem“ des Entfaltens:

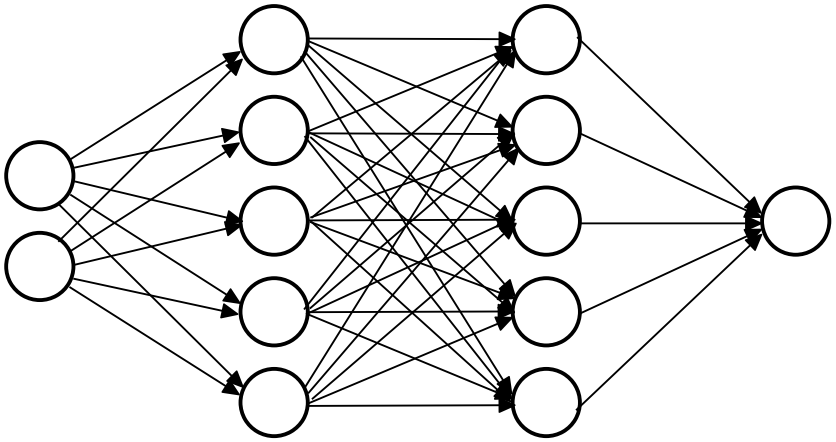
Entfaltung:		**Maschinelles Lernen**:
Zu entfaltende Wahrheit		Zu approximierende Grundgesamtheit
Entfaltungsmatrix		Trainingsdatensatz
Entfaltung		MLP Modell nach Training

- Wie bei der Entfaltung helfen **Regularisierungsmaßnahmen** dabei die „Korrespondenz des Modells mit der Wahrheit“ zu erhöhen.

Dropout

- Es hat regulierende Wirkung ein Ensembe variierender MLP Architekturen zu trainienren und über deren Ergebnisse zu mitteln.

- Eine einfache Realisierung besteht im sog. (*inverted*) **Dropout**:

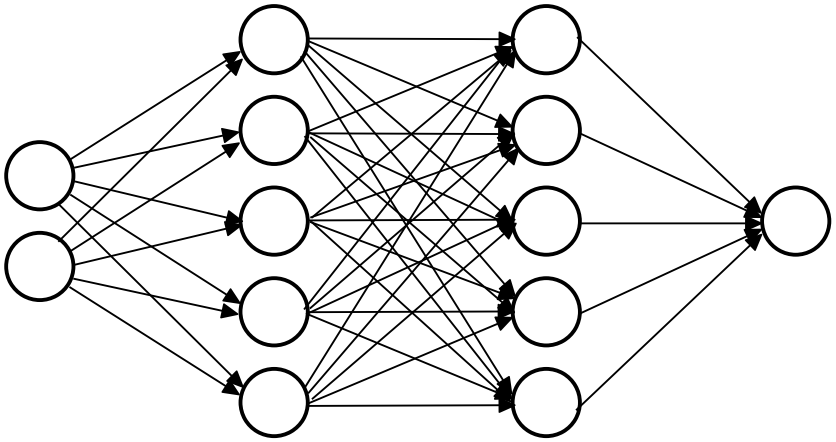


Dropout

- Es hat regulierende Wirkung ein Ensembe variierender MLP Architekturen zu trainienren und über deren Ergebnisse zu mitteln.

- Eine einfache Realisierung besteht im sog. (*inverted*) **Dropout**:

  - „Lösche“ vor jedem Gradientenabstieg zufällig Knoten (und alle damit bestehenden Verbindungen) mit einer Wahrscheinlichkeit d .



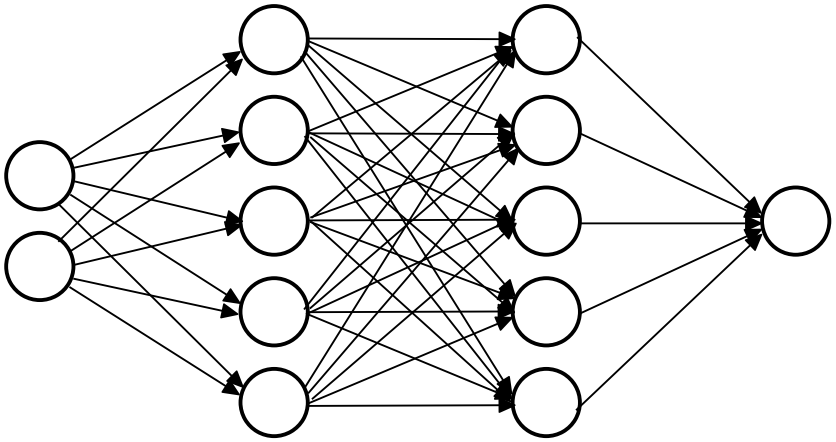
Dropout

- Es hat regulierende Wirkung ein Ensembe variierender MLP Architekturen zu trainienren und über deren Ergebnisse zu mitteln.

- Eine einfache Realisierung besteht im sog. (*inverted*) **Dropout**:

  - „Lösche“ vor jedem Gradientenabstieg zufällig Knoten (und alle damit bestehenden Verbindungen) mit einer Wahrscheinlichkeit d .

  - Damit entsteht für jeden Gradientenabstieg eine leicht variierte MLP Architektur.



Dropout

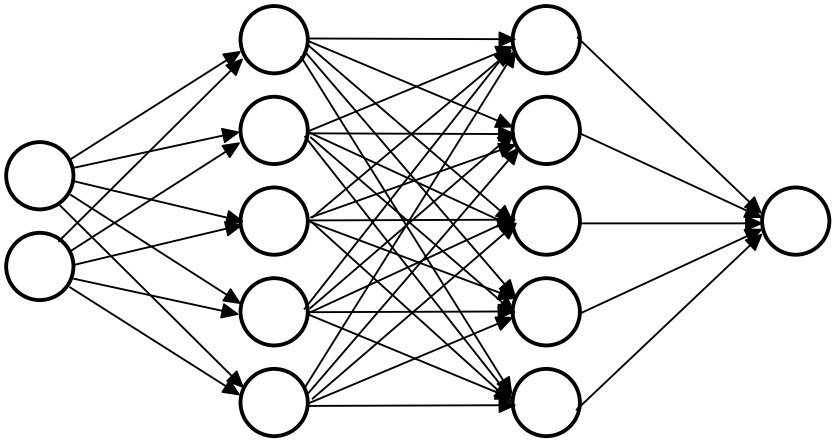
- Es hat regulierende Wirkung ein Ensembe variierender MLP Architekturen zu trainienren und über deren Ergebnisse zu mitteln.

- Eine einfache Realisierung besteht im sog. (*inverted*) **Dropout**:

  - „Lösche“ vor jedem Gradientenabstieg zufällig Knoten (und alle damit bestehenden Verbindungen) mit einer Wahrscheinlichkeit d .

  - Damit entsteht für jeden Gradientenabstieg eine leicht variierte MLP Architektur.

  - Renormiere die verbliebenen Gewichte mit $1/(1-d)$ Warum? –



Dropout

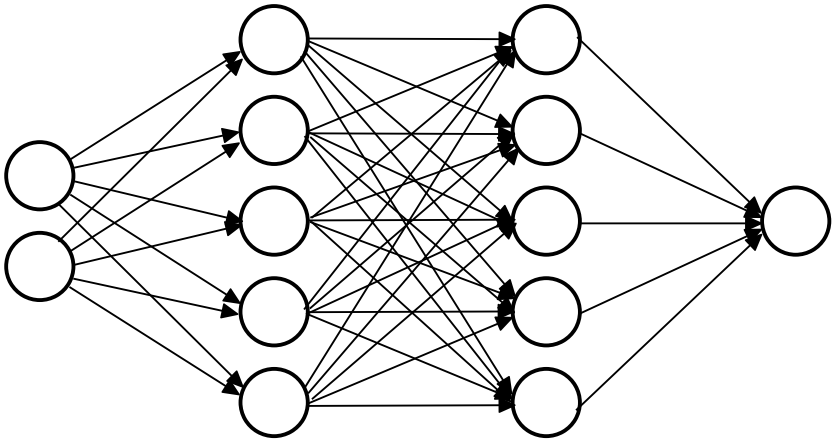
- Es hat regulierende Wirkung ein Ensembe variierender MLP Architekturen zu trainienren und über deren Ergebnisse zu mitteln.

- Eine einfache Realisierung besteht im sog. (*inverted*) **Dropout**:

  - „Lösche“ vor jedem Gradientenabstieg zufällig Knoten (und alle damit bestehenden Verbindungen) mit einer Wahrscheinlichkeit d .

  - Damit entsteht für jeden Gradientenabstieg eine leicht variierte MLP Architektur.

  - Renormiere die verbliebenen Gewichte mit $1/(1-d)$ Warum? – Stellen Sie sich vor, Sie hätten einen Anteil d der Knoten aus Lage (k) gelöscht. D.h.:



$$z_j^{(k)} = \sum_i y_i^{(k-1)} w_{ij}^{(k)}; \quad \langle z_j^{(k)} \rangle \Big|_p = (1-p) \langle z_j^{(k)} \rangle$$

Dropout

- Es hat regulierende Wirkung ein Ensembe variierender MLP Architekturen zu trainienren und über deren Ergebnisse zu mitteln.

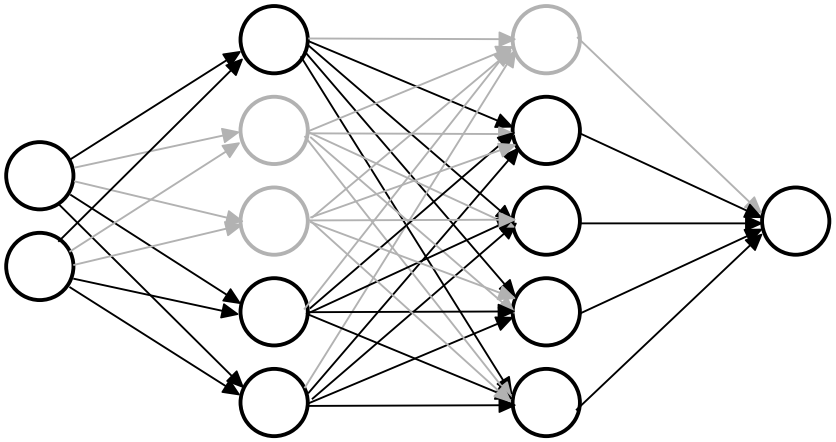
- Eine einfache Realisierung besteht im sog. (*inverted*) **Dropout**:

  - „Lösche“ vor jedem Gradientenabstieg zufällig Knoten (und alle damit bestehenden Verbindungen) mit einer Wahrscheinlichkeit d .

  - Damit entsteht für jeden Gradientenabstieg eine leicht variierte MLP Architektur.

  - Renormiere die verbliebenen Gewichte mit $1/(1-d)$ Warum? – Stellen Sie sich vor, Sie hätten einen Anteil d der Knoten aus Lage (k) gelöscht. D.h.:

Dropout:



$d = 0.3$

$$z_j^{(k)} = \sum_i y_i^{(k-1)} w_{ij}^{(k)}; \quad \langle z_j^{(k)} \rangle \Big|_p = (1-p) \langle z_j^{(k)} \rangle$$

Dropout

- Es hat regulierende Wirkung ein Ensembe variierender MLP Architekturen zu trainienren und über deren Ergebnisse zu mitteln.

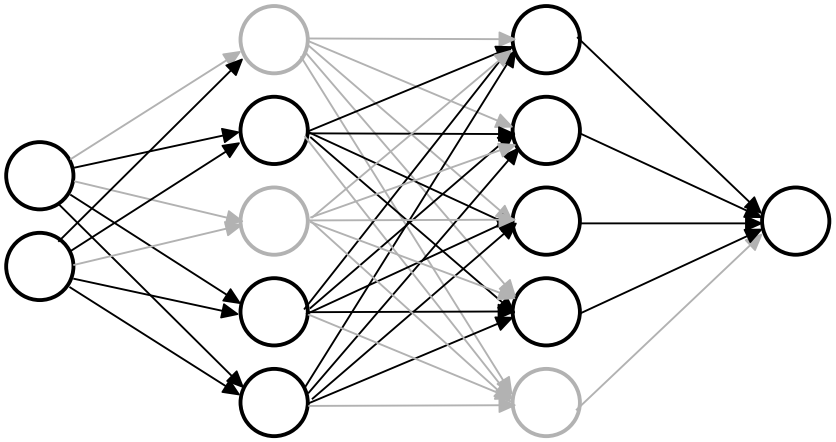
- Eine einfache Realisierung besteht im sog. (*inverted*) **Dropout**:

  - „Lösche“ vor jedem Gradientenabstieg zufällig Knoten (und alle damit bestehenden Verbindungen) mit einer Wahrscheinlichkeit d .

  - Damit entsteht für jeden Gradientenabstieg eine leicht variierte MLP Architektur.

  - Renormiere die verbliebenen Gewichte mit $1/(1-d)$ Warum? – Stellen Sie sich vor, Sie hätten einen Anteil d der Knoten aus Lage (k) gelöscht. D.h.:

Dropout:



$d = 0.3$

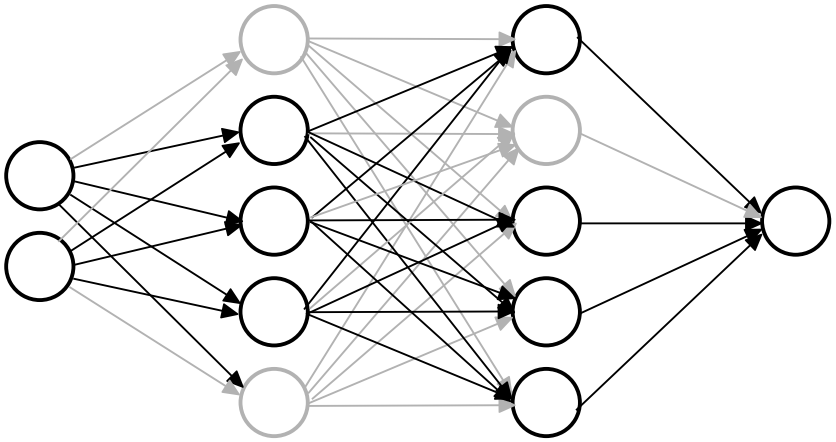
$$z_j^{(k)} = \sum_i y_i^{(k-1)} w_{ij}^{(k)}; \quad \langle z_j^{(k)} \rangle \Big|_p = (1-p) \langle z_j^{(k)} \rangle$$

Dropout

- Es hat regulierende Wirkung ein Ensembe variierender MLP Architekturen zu trainienren und über deren Ergebnisse zu mitteln.

- Eine einfache Realisierung besteht im sog. (*inverted*) **Dropout**:
  - „Lösche“ vor jedem Gradientenabstieg zufällig Knoten (und alle damit bestehenden Verbindungen) mit einer Wahrscheinlichkeit d .
  - Damit entsteht für jeden Gradientenabstieg eine leicht variierte MLP Architektur.
  - Renormiere die verbliebenen Gewichte mit $1/(1-d)$ Warum? – Stellen Sie sich vor, Sie hätten einen Anteil d der Knoten aus Lage (k) gelöscht. D.h.:

Dropout:



$d = 0.3$

$$z_j^{(k)} = \sum_i y_i^{(k-1)} w_{ij}^{(k)}; \quad \langle z_j^{(k)} \rangle \Big|_p = (1-p)\langle z_j^{(k)} \rangle$$

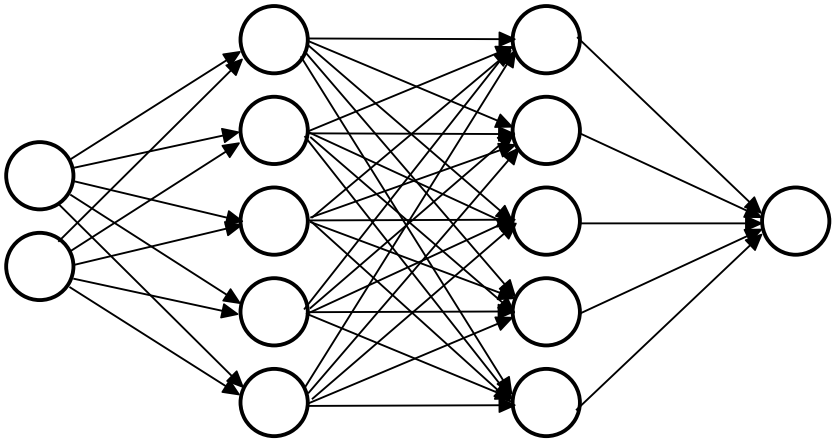
Dropout

- Es hat regulierende Wirkung ein Ensembe variierender MLP Architekturen zu trainienren und über deren Ergebnisse zu mitteln.

- Eine einfache Realisierung besteht im sog. (*inverted*) **Dropout**:
  - „Lösche“ vor jedem Gradientenabstieg zufällig Knoten (und alle damit bestehenden Verbindungen) mit einer Wahrscheinlichkeit d .
  - Damit entsteht für jeden Gradientenabstieg eine leicht variierte MLP Architektur.
  - Renormiere die verbliebenen Gewichte mit $1/(1-d)$ Warum? – Stellen Sie sich vor, Sie hätten einen Anteil d der Knoten aus Lage (k) gelöscht. D.h.:

$$z_j^{(k)} = \sum_i y_i^{(k-1)} w_{ij}^{(k)}; \quad \langle z_j^{(k)} \rangle \Big|_p = (1-p)\langle z_j^{(k)} \rangle$$

Dropout:

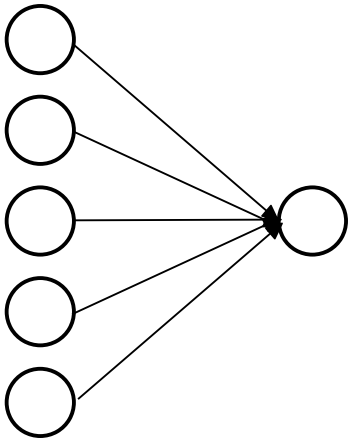


$d = 0.3$

Übliche Dropoutwahrscheinlichkeiten liegen bei $d = 0.3 \ldots 0.5$.

Wie und warum funktionert Dropout?

- Wie kann Dropout die Gewichte des MLP regularisieren?



- Der Knoten k erhält seine Information aus 5 Vorgängerknoten.
- Jeder Vorgängerknoten könnte im nächsten Gradientenabstieg „ausgelöscht“ werden.
- Der Konten k kann sich zur Entscheidungsfindung also nicht auf einen einzelnen Vorgängerknoten verlassen. Er muss die notwendige Information aus mehreren Vorgängerknoten verteilt beziehen.

- Dies führt zu einer gleichmäßigeren Verteilung der Gewichte.
- Man kann zeigen, das Dropout zu einer L2 Regularisierung äquivalent ist, bei der der Parameter λ in jedem Knoten separat bestimmt wird.

L1 und L2 Regularisierung

- Die letzte Form der Regularisierung, die wir diskutieren werden ist die L1 und/oder L2 Regularisierung.

- Diese sollte Ihnen aus der Optimierung mit Nebenbedingungen mit Hilfe der Implementierung von Straftermen (*penalty terms*) bekannt sein.

- Hierzu fügen Sie der Verlustfunktion einfach die Summe aller Gewichte in Form der L1 oder L2 Norm zu:

$$L(\{p_i\},\{y_i\},\{w_\alpha\}) = -\sum_{i=1}^{n} p_i \log(y_i) + \lambda\,||w_\alpha||_{L1/2}$$

n : Länge der Stichprobe

y_i : MLP output für Beispiel i

$$||w_\alpha||_{L1} = \sum_\alpha |w_\alpha| \qquad \text{(}\textit{least absolute shrinkage and selection operator, Lasso}\text{)}$$

$$||w_\alpha||_{L2} = \sum_\alpha w_\alpha^2 \qquad \text{(}\textit{ridge regularization}\text{)}$$

L1 Regularisierung

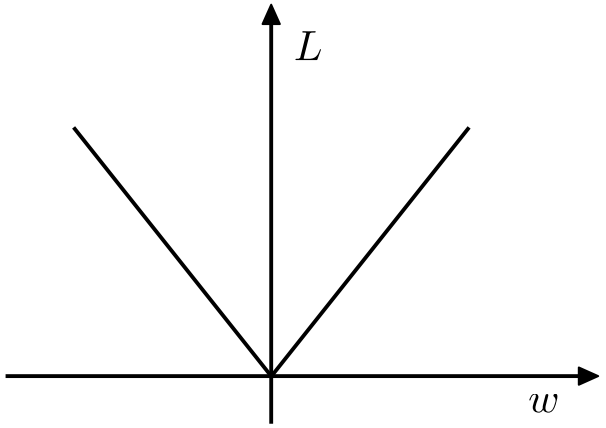
- Ableitung der L1-Norm:

$$\frac{\mathrm{d}|w|}{\mathrm{d}w} = \begin{cases} +1 & w > 0 \\ -1 & w < 0 \end{cases}$$

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L - \mathrm{sgn}(w)\,\lambda, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$



- Löscht einzelne Knoten aus.

L2 Regularisierung

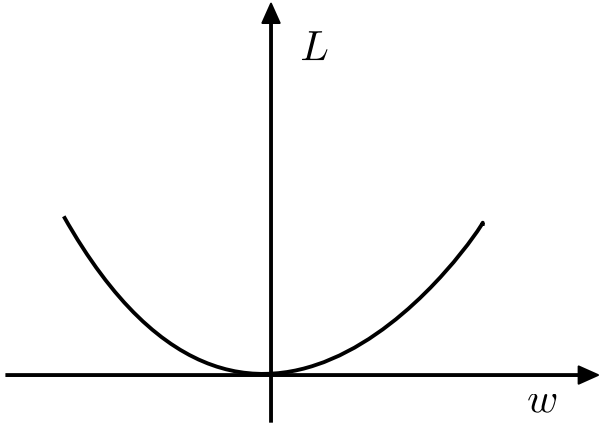
- Ableitung der L2-Norm:

$$\frac{\mathrm{d}w^2}{\mathrm{d}w} = 2\,w$$

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L - 2\,\lambda\,w, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$



- Reduziert Beträge einzelner Gewichte.

L1 Regularisierung

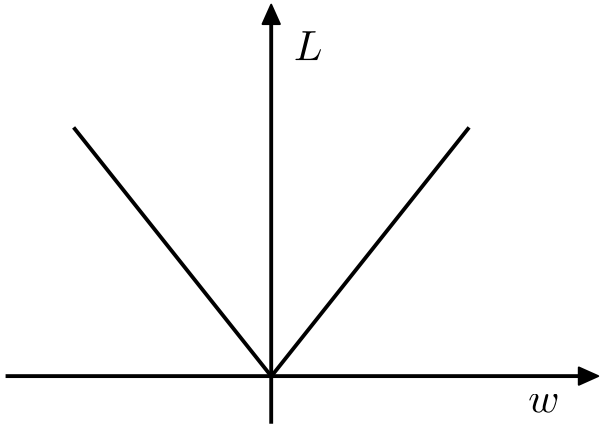
- Ableitung der L1-Norm:

$$\frac{\mathrm{d}|w|}{\mathrm{d}w} = \begin{cases} +1 & w > 0 \\ -1 & w < 0 \end{cases}$$

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L - \mathrm{sgn}(w)\,\lambda, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$



- Löscht einzelne Knoten aus.

L2 Regularisierung

- Ableitung der L2-Norm:

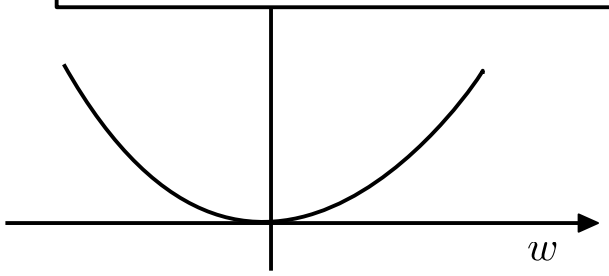
$$\frac{\mathrm{d}w^2}{\mathrm{d}w} = 2\,w$$

$$\vec{w}^{(k+1)} = \vec{w}^{(k)} - \eta \nabla_{\vec{w}^{(k)}} L - 2\,\lambda\,w, \quad \eta > 0$$

solange:

$$\left| L(w^{(k)}) - L(w^{(k-1)}) \right| > \epsilon$$

NB: Entspricht die Verlustfunktion der NLL dann entspricht die L2 Norm einem normalverteilten *constraint* mit Standardabweichung λ.



- Reduziert Beträge einzelner Gewichte.

Diskussion Regularisierung

- Im Allgemeinen gilt:
  - Je mehr trainierbare Parameter das MLP hat, desto schneller gerät man in den Bereich des overtraining.
  - Je größer der Trainingsdatensatz ist, desto geringer ist die Gefahr des overtraining.
- Es ist also auch immer möglich durch Erhöhung des Trainingsdatensatzes das Training zu regularisieren.
- Ein vorgehen, dass wir hier nicht diskutiert haben, weil es in der Teilchenphysik irrelevant ist bezeichnet man als *data augmentation*:
  - Dabei erweitert man den Trainingsdatensatz künstlich, z.B. durch Drehen, Verzerren oder Spiegeln der Beispiele.

Backup