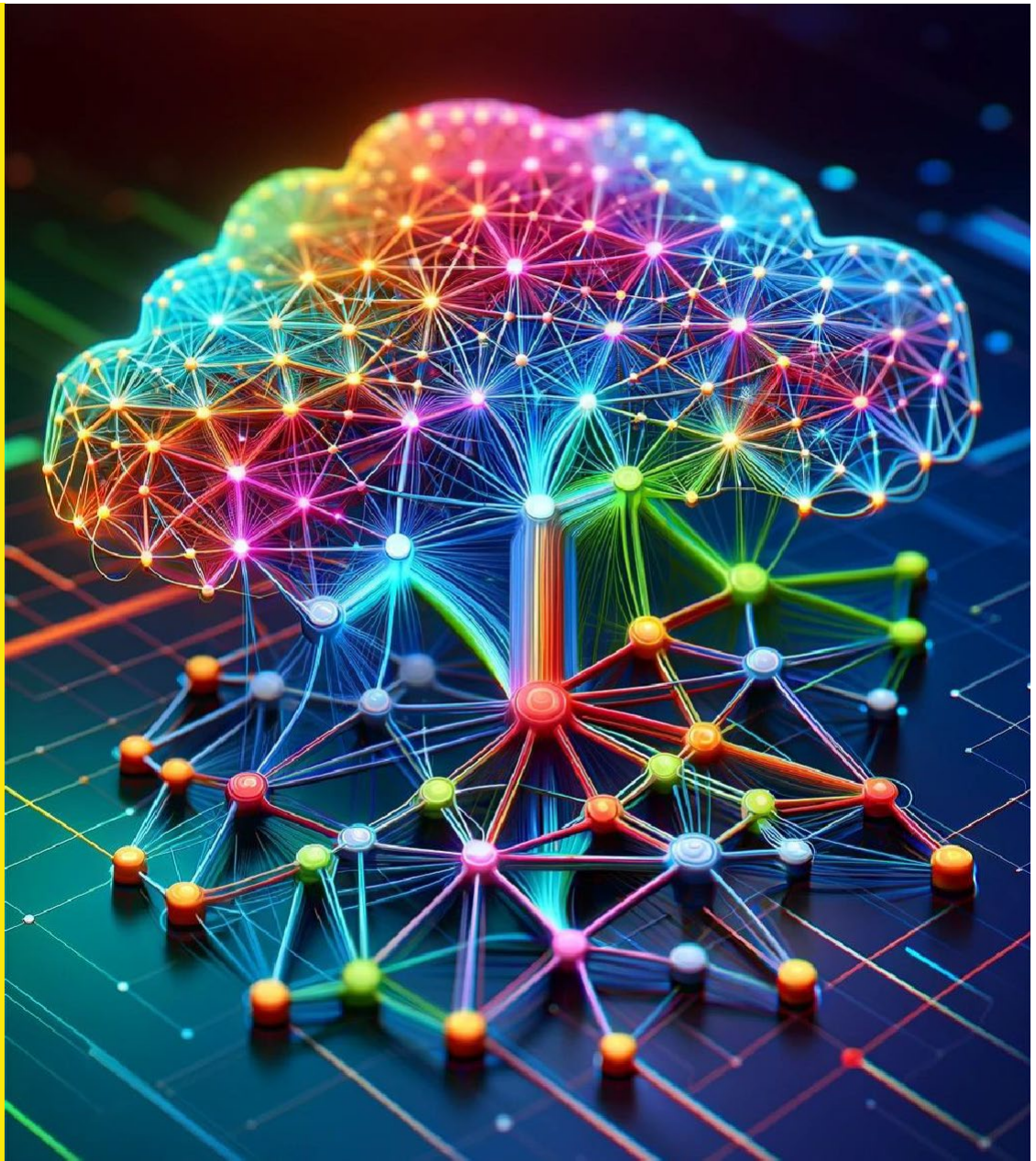
University of Applied Sciences and Arts Northwestern Switzerland
School of Business

swissuniversities

AACSB ACCREDITED

Deep Learning

Künstliche Intelligenz | BSc BAI

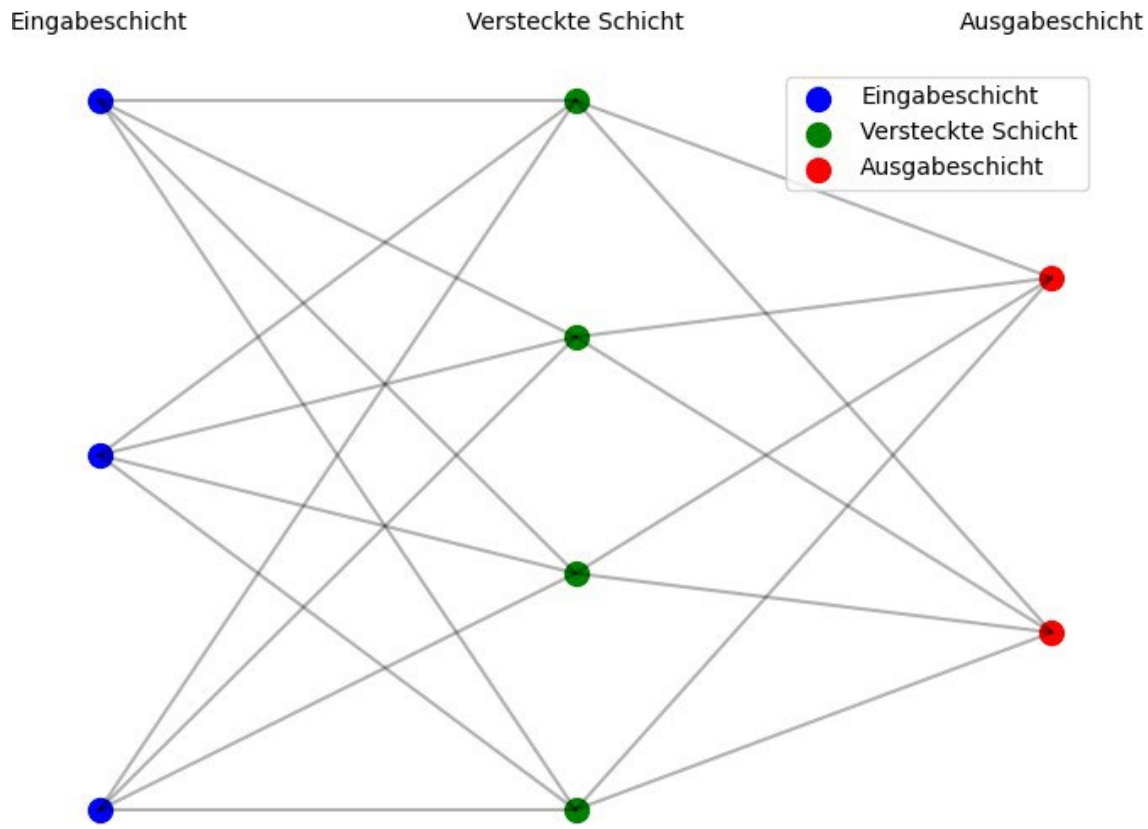


Prof. Dr. Andreas Martin & Prof. Dr. Manuel Renold

1. Neuronale Netzwerke

Neuronale Netzwerke

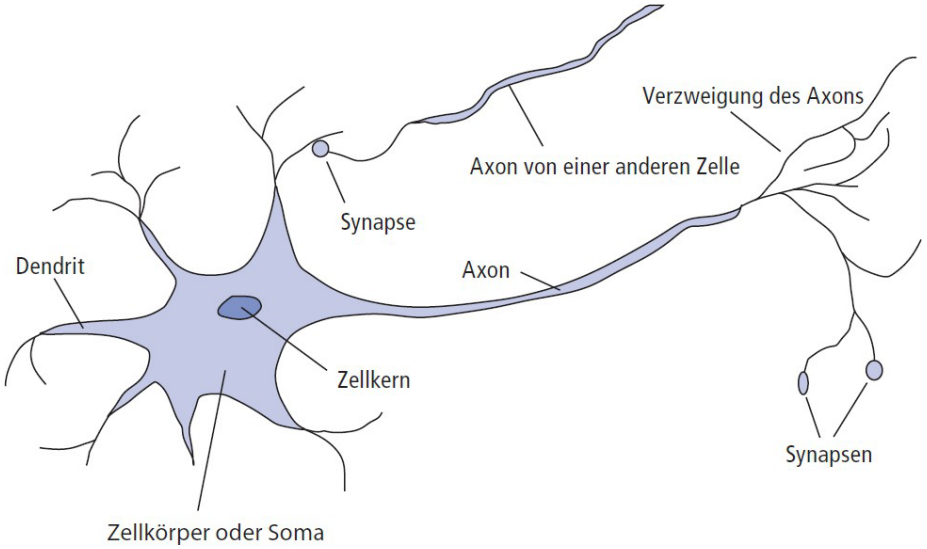
- Neuronale Netzwerke ahmen die Funktionsweise des menschlichen Gehirns nach und bestehen aus einer Reihe von künstlichen Neuronen in verschiedenen Schichten:
- Eingabe-, versteckte und Ausgabeschicht.
- Jedes Neuron verarbeitet Eingabesignale mittels Gewichte und einem Bias-Wert, ähnlich wie Koeffizienten in der logistischen Regression.
- Das Ergebnis wird dann durch eine Aktivierungsfunktion geleitet.
- Im Gegensatz zur logistischen Regression, die auf einer einzelnen linearen Entscheidungsebene operiert, können neuronale Netzwerke durch ihre Schichtenstruktur komplexere Muster in Daten erfassen.



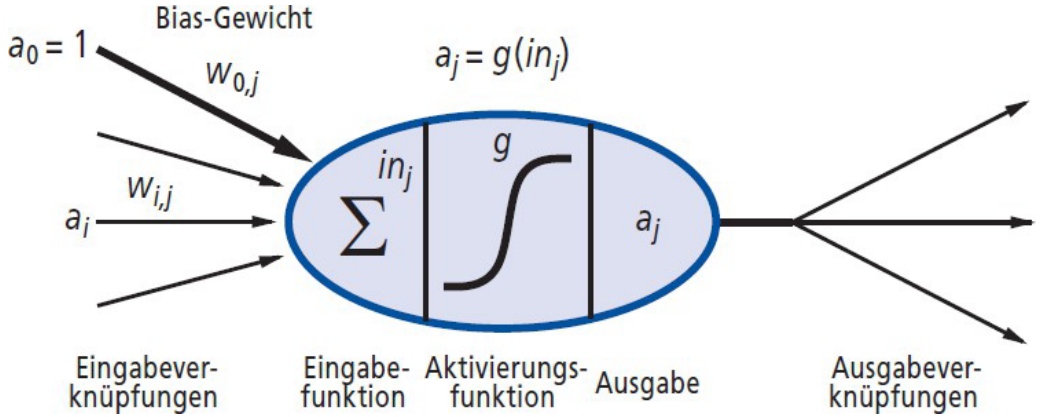
Wiederholung SW1!

Künstliche Neuronale Netze (1)

- Künstliche Neuronen sind mathematische Modelle, die ungefähr nachahmen, wie man sich vor langer Zeit vorgestellt hat, dass ein Neuron im Gehirn funktioniert.



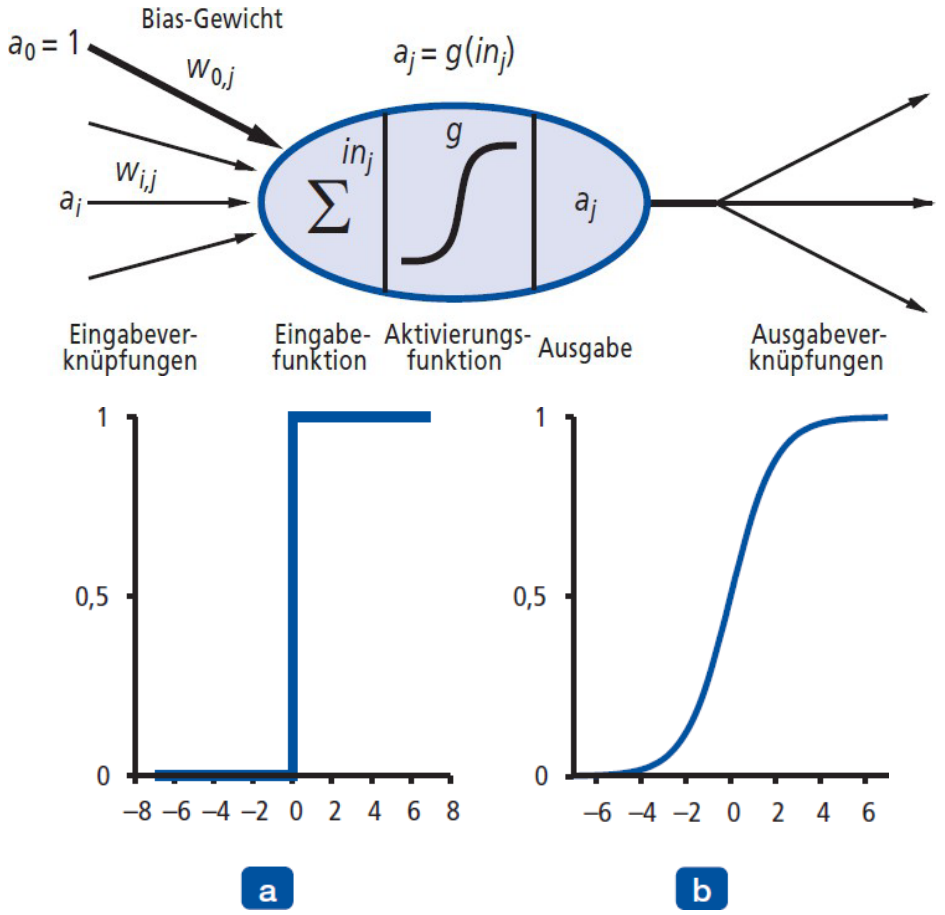
- Neuronale Netze bestehen aus Knoten Verknüpft durch gerichtete Verbindungen
- Jeder Verknüpfung hat ein Gewicht die Stärke der Verknüpfung bestimmt



Russell, S. J., & Norvig, P. (2012). Künstliche Intelligenz: Ein moderner Ansatz (F. Langenau, Übers.; 3., aktualisierte Auflage). Pearson, Higher Education.

Künstliche Neuronale Netze (2)

- Neuronale Netze:
  - Bestehen aus Knoten
  - Verknüpft durch gerichtete Verbindungen
  - Verknüpfungen propagieren Aktivierung von i nach j
- Numerische Gewichte:
  - Jeder Verknüpfung hat ein Gewicht
  - Bestimmt Stärke und Vorzeichen der Verknüpfung
  - Gewichte beeinflussen die Aktivierung der Einheiten
- Aktivierungsfunktionen:
  - Beeinflussen die Aktivierung der Einheiten
- Vorgehen
  - 1. Eingaben (ai) werden in das Neuron eingespeist
  - 2. Gewichte (wi,j) werden mit jeder Eingabe (ai) multipliziert
  - 3. Summierung (wi,j * ai) und Bias-Gewicht («Verschiebung») addieren.
  - 4. Aktivierungsfunktion ggf. aktiviert

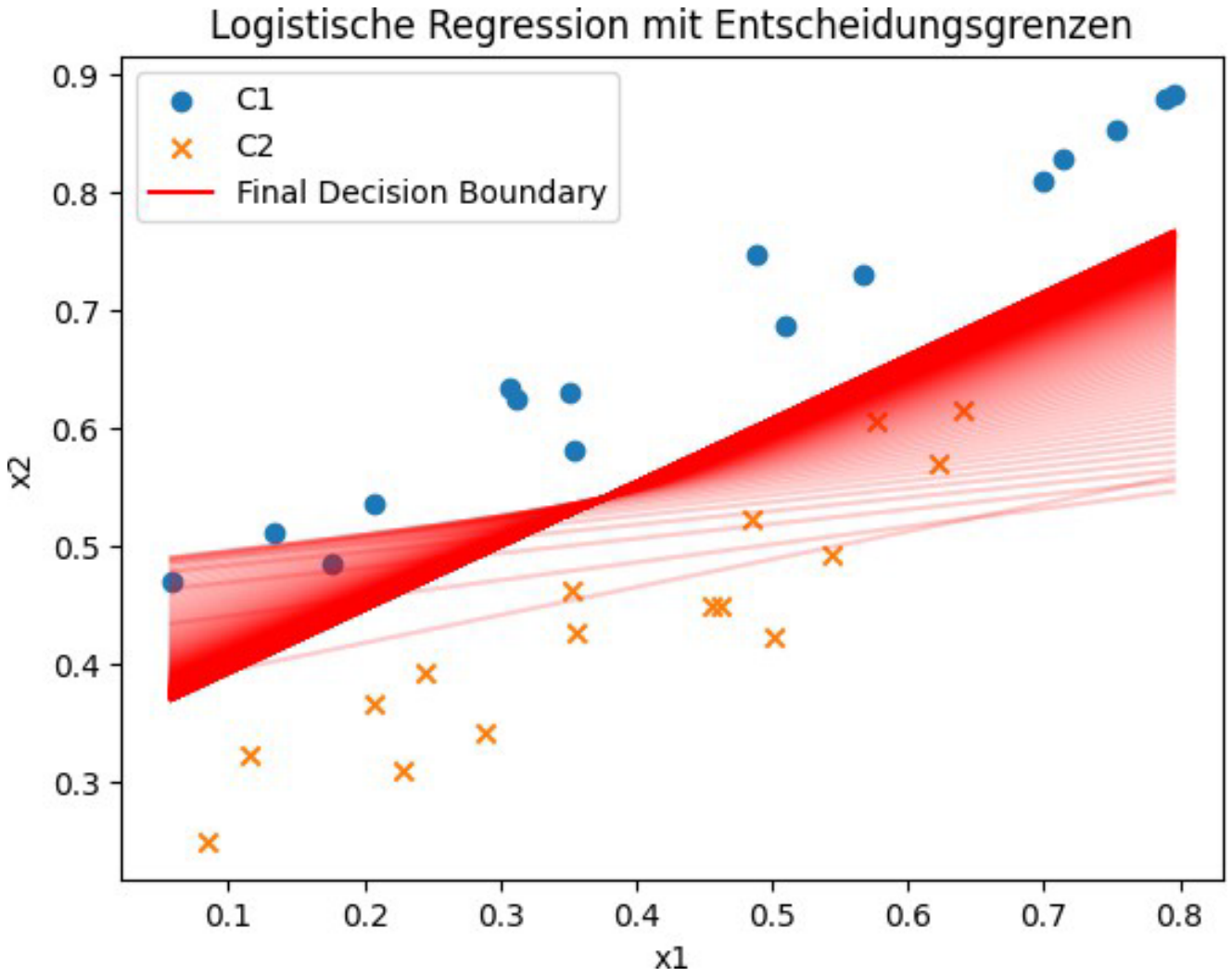


Russell, S. J., & Norvig, P. (2012). Künstliche Intelligenz: Ein moderner Ansatz (F. Langenau, Übers.; 3., aktualisierte Auflage). Pearson, Higher Education.

2. Logistische Regression und Neuron

Ähnlichkeiten zur Logistischen Regression: Entscheidungsgrenze

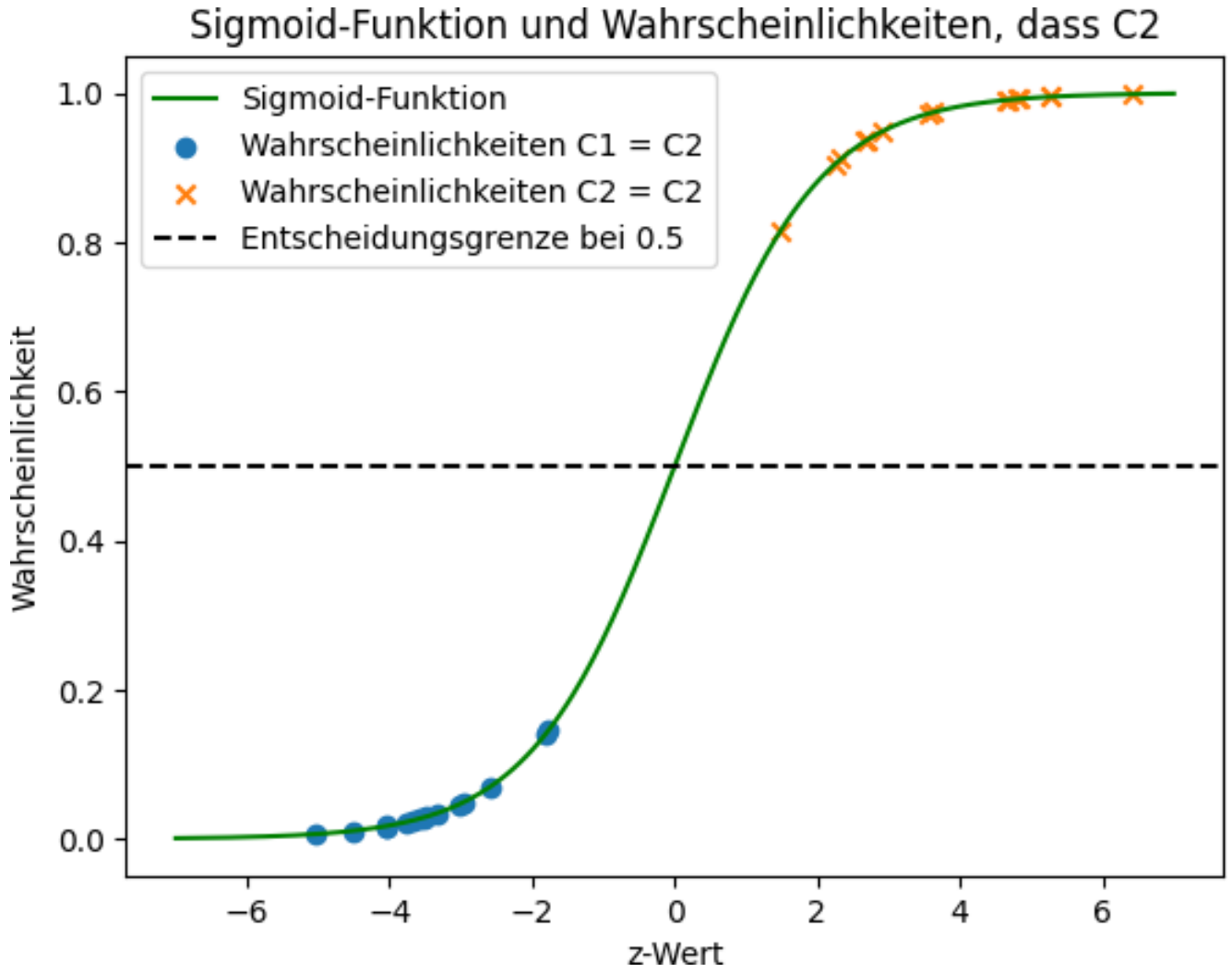
- Entscheidungsgrenze (Decision Boundary): In der logistischen Regression definiert durch die lineare Funktion $z = w_1x_1 + w_2x_2 + b$.
- z bildet eine Grenze zwischen den Klassen, basierend auf den **«trainierten» Merkmalsgewichtungen** w_1, w_2 oder **Koeffizienten** und den Merkmalswerten x_1, x_2.
- Der **«trainierte» Bias** b verschiebt die Entscheidungsgrenze, auf der Y-Achse (Achsenabschnitt; Intercept).
- Wenn z einen bestimmten Schwellenwert überschreitet, wird die Beobachtung einer Klasse zugeordnet.
- Liegt z unter diesem Schwellenwert, gehört die Beobachtung zur anderen Klasse.



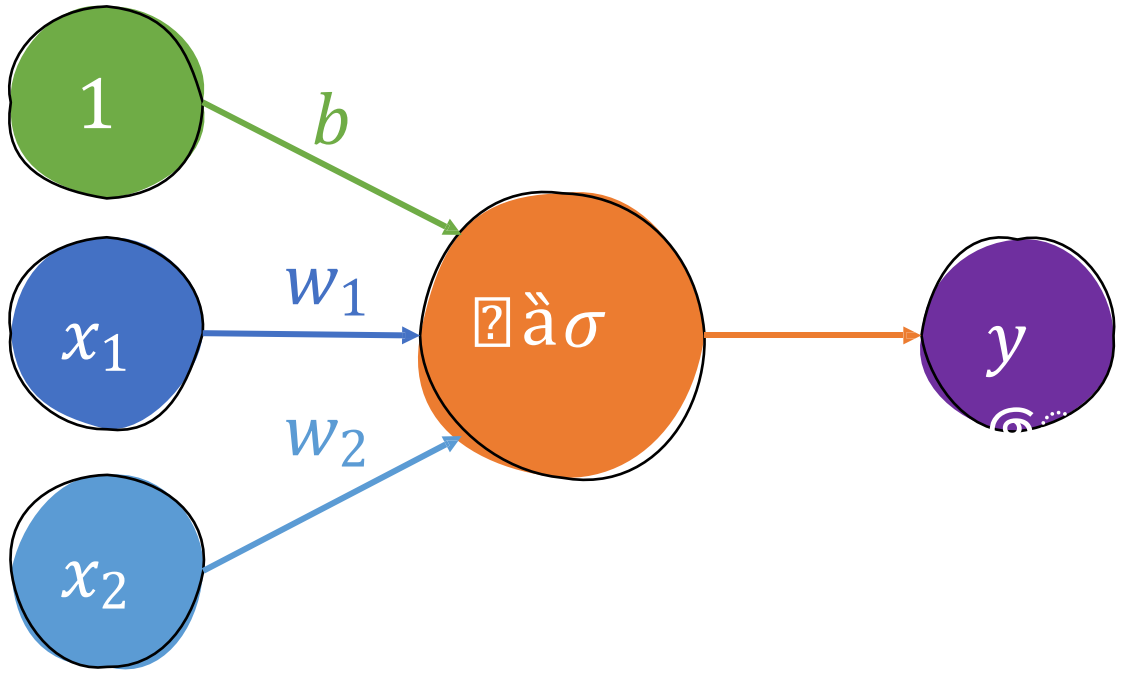
Ähnlichkeiten zur Logistischen Regression: Wahrscheinlichkeit & Sigmoid-Funktion

Sigmoid-Funktion zur Umwandlung der linearen Ausgabe z in einen Wahrscheinlichkeitswert zwischen 0 und 1.

- $g(z) = \frac{1}{1+e^{-z}}$
  - e: Euler'sche Zahl, Basis des natürlichen Logarithmus, ungefähr 2.71828.
  - z: Ergebnis der linearen Gleichung $z = w_1x_1 + w_2x_2 + b$.
  - $g(z)$: Gibt die Wahrscheinlichkeit an, dass eine Beobachtung zur positiven Klasse gehört (z.B. Klasse 2 in einem binären Klassifikationsproblem).



Das «Neuron» der Logistischen Regression



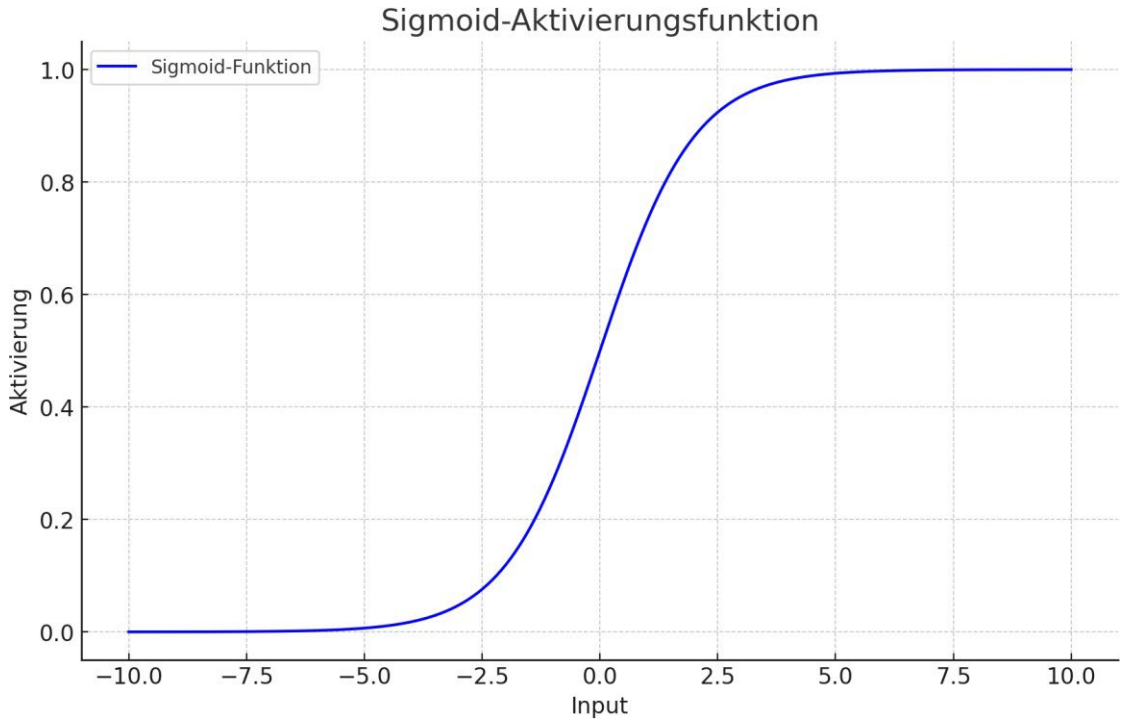
Ein einfaches künstliches Neuron mit der logistischen Regression als Aktivierungsfunktion für eine binäre Klassifikation.

- x_1, x_2: Die **Eingabemerkmale** oder Variablen, die in das Modell eingespeist werden.
- 1: Stellt den **Bias** b dar, ein Wert, der zu der Linearkombination hinzugefügt wird, um die Entscheidungsgrenze des Modells zu verschieben.
- σàσ: Symbolisiert das Neuron, das eine **Linearkombination** durchführt und dann die **Aktivierungs-Funktion** anwendet.
  - σ à steht für die Summation (die lineare Kombination der gewichteten Eingaben und des Bias)
  - σ bezieht sich auf die logistische (Sigmoid-) Funktion
- y Die **Ausgabe** des Neurons, das ist die geschätzte Wahrscheinlichkeit $P(C1)$, dass die Eingabedaten zur Klasse $C1$ gehören.

3. Aktivierungsfunktionen

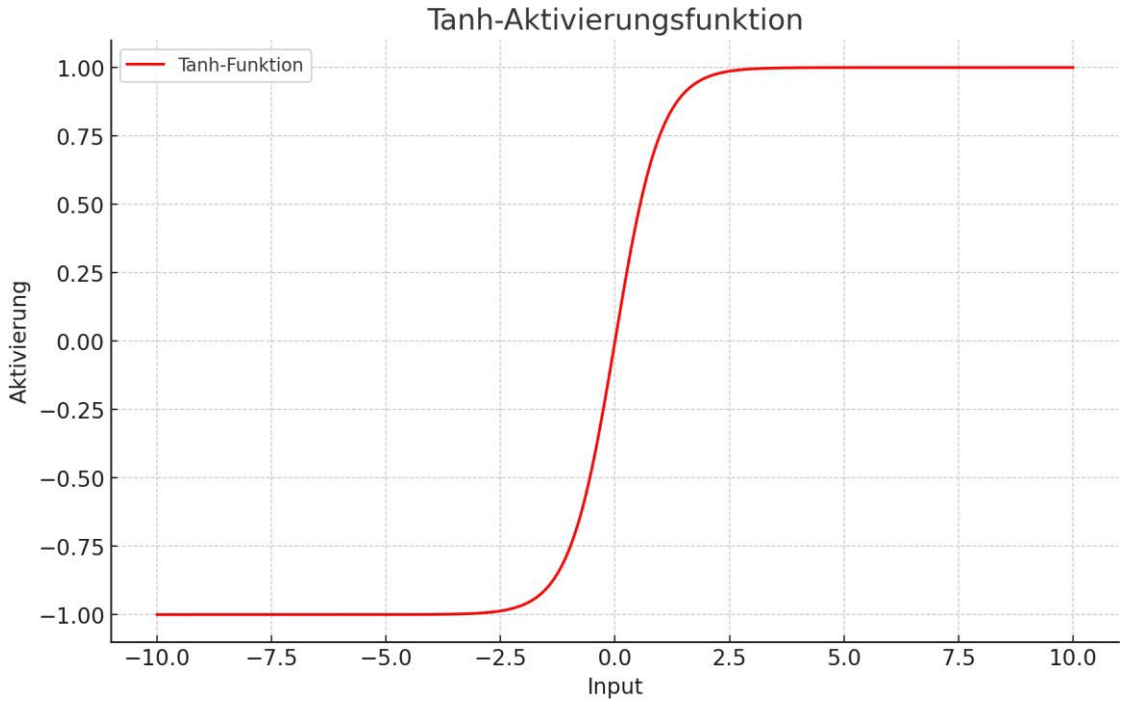
Aktivierungsfunktion: Sigmoid-Funktion

- Eigenschaften: Abbildung von Eingaben in einen Bereich zwischen 0 und 1, S-förmige Kurve. Empfindlich bei Eingaben nahe Null, aber Sättigung bei hohen positiven oder negativen Werten.
- Verwendung in Logistischer Regression: Ideal für binäre Klassifizierung, da Wahrscheinlichkeiten ausgegeben werden.
- Verwendung in MLPs (Mehrschichtige Perzeptronen): Weniger häufig aufgrund des Problems des verschwindenden Gradienten, bei dem Gradienten sehr klein werden und das Lernen verlangsamen.



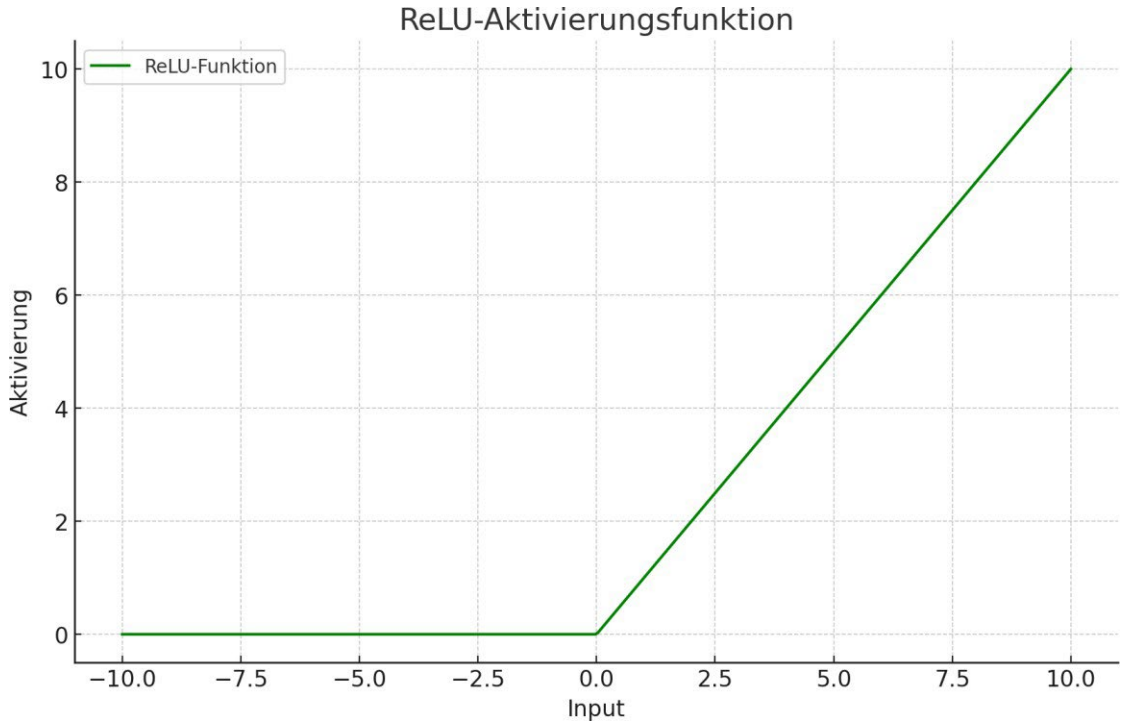
Aktivierungsfunktion: Tanh (Hyperbolischer Tangens)

- Eigenschaften: Ähnlich wie Sigmoid, bildet Eingaben aber in einen Bereich zwischen -1 und 1 ab. Wie Sigmoid hat sie eine S-Form und sättigt bei Eingaben mit grossen Beträgen.
- Verwendung in Logistischer Regression: Nicht typisch verwendet, da der Ausgabebereich nicht mit der Wahrscheinlichkeitsinterpretation übereinstimmt.
- Verwendung in MLPs: Häufiger als Sigmoid in versteckten Schichten verwendet, da sie nullzentriert ist und das Training in einigen Fällen erleichtert, aber immer noch mit dem Problem des verschwindenden Gradienten konfrontiert ist.



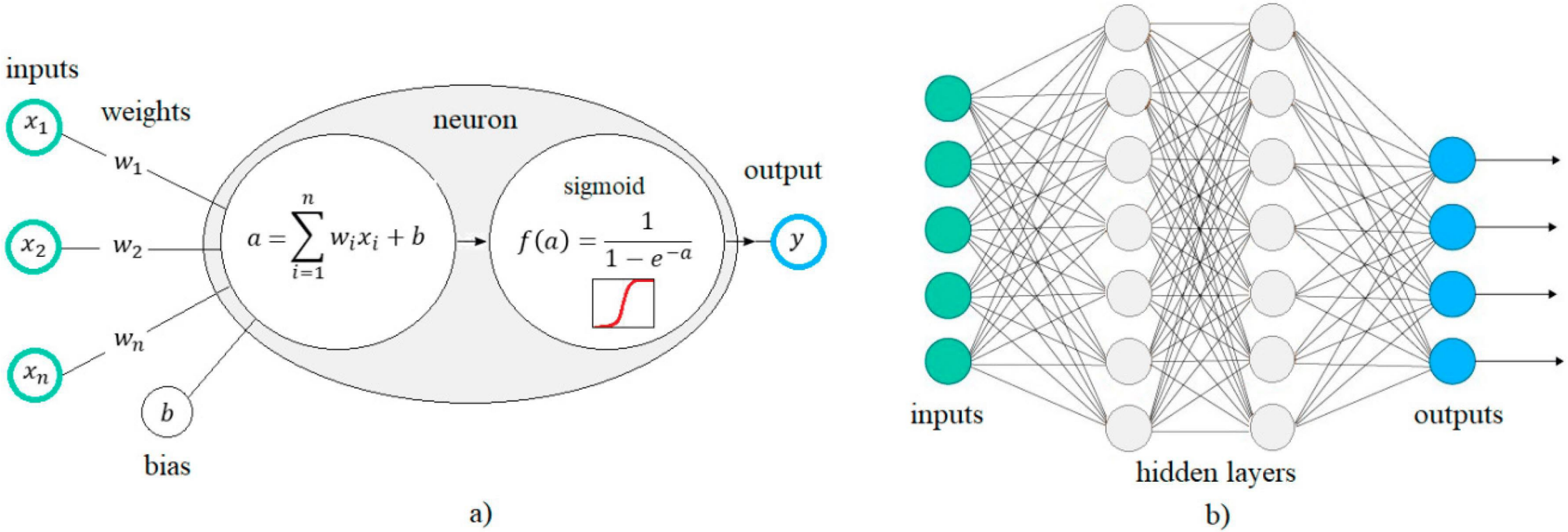
Aktivierungsfunktion: ReLU (Rectified Linear Unit)

- Eigenschaften: ReLU gibt den Eingabewert aus, wenn er positiv ist; andernfalls gibt sie Null aus. Sie ist nicht begrenzt, und Gradienten verschwinden nicht bei positiven Eingaben.
- Verwendung in Logistischer Regression: Nicht geeignet, da sie keine Ausgaben im Wahrscheinlichkeitsbereich (0 bis 1) produziert.
- Verwendung in MLPs: Sehr beliebt in versteckten Schichten wegen der rechentechnischen Effizienz und Lösung des Problems des verschwindenden Gradienten bei positiven Eingaben.



4. Deep Learning

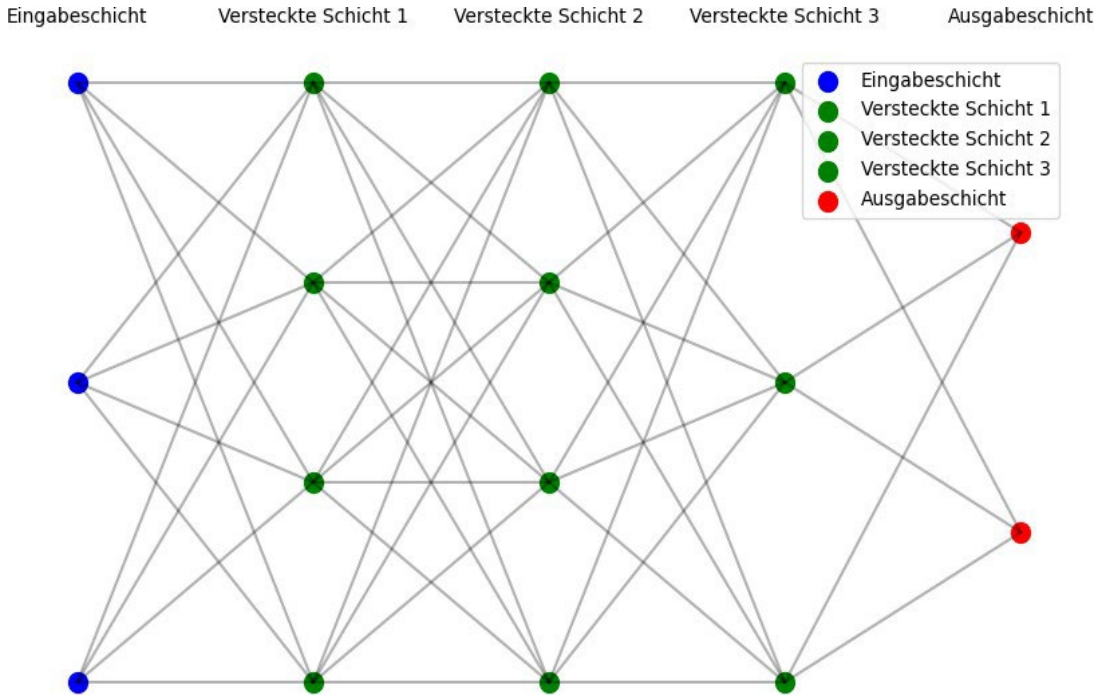
Neuronal ML zu Deep Learning



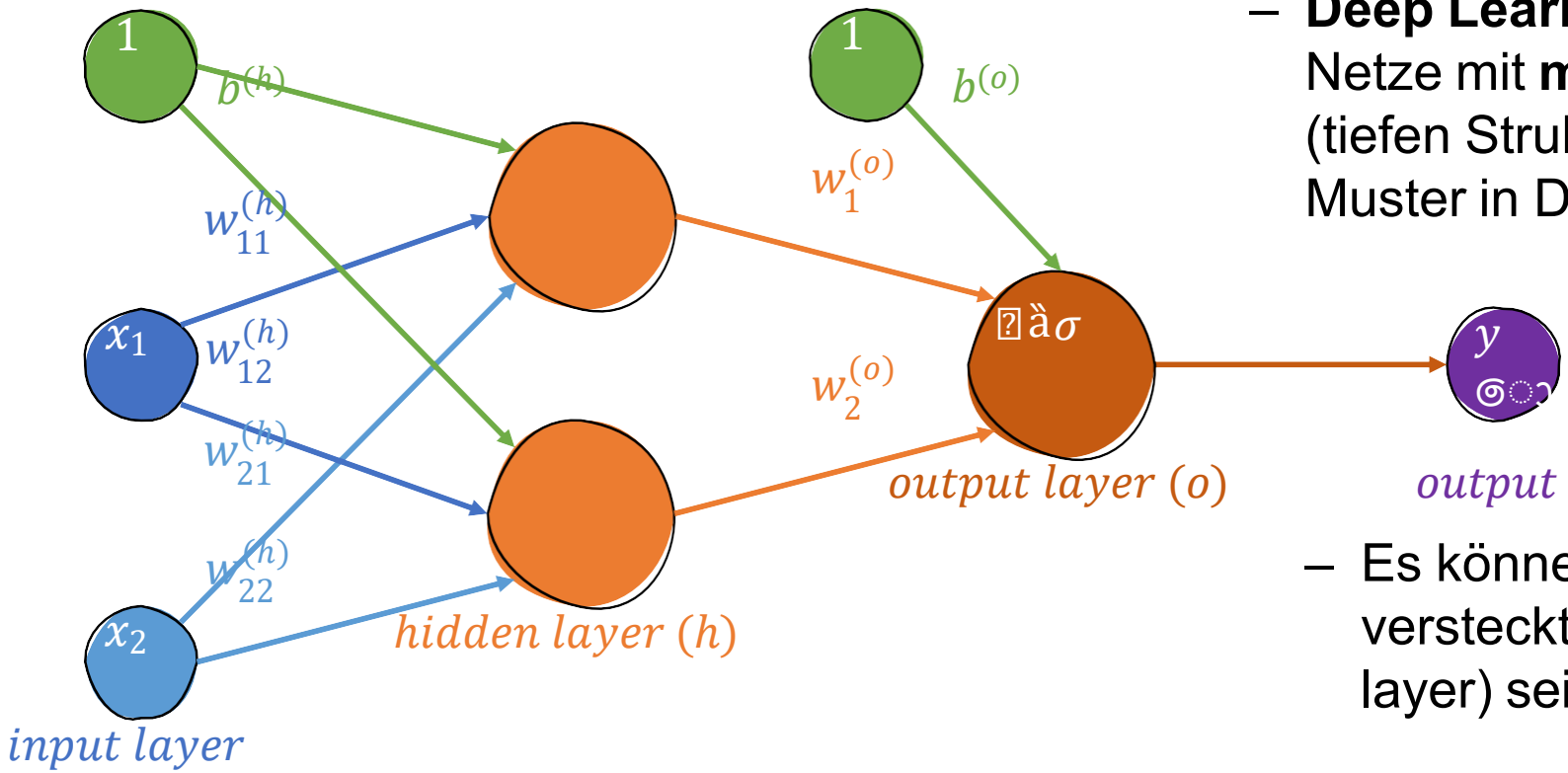
Dropka, N., & Holena, M. (2020). Application of Artificial Neural Networks in Crystal Growth of Electronic and Opto-Electronic Materials. Crystals, 10(8), 663. https://doi.org/10.3390/cryst10080663

Deep Learning

- Deep Learning bezieht sich auf Netzwerke mit mehreren (oft vielen) Schichten, die es ermöglichen, komplexere Muster und Beziehungen in Daten zu erkennen.
- Tiefe Netzwerke können Merkmale auf verschiedenen Ebenen lernen, von einfachen Mustern in frühen Schichten bis hin zu hochkomplexen Strukturen in tieferen Schichten.
- Der Gradientenabstieg spielt eine zentrale Rolle im Deep Learning.
- Er wird verwendet, um die Gewichte des Netzwerks zu optimieren, indem der Fehler zwischen den Vorhersagen des Netzwerks und den tatsächlichen Werten minimiert wird.



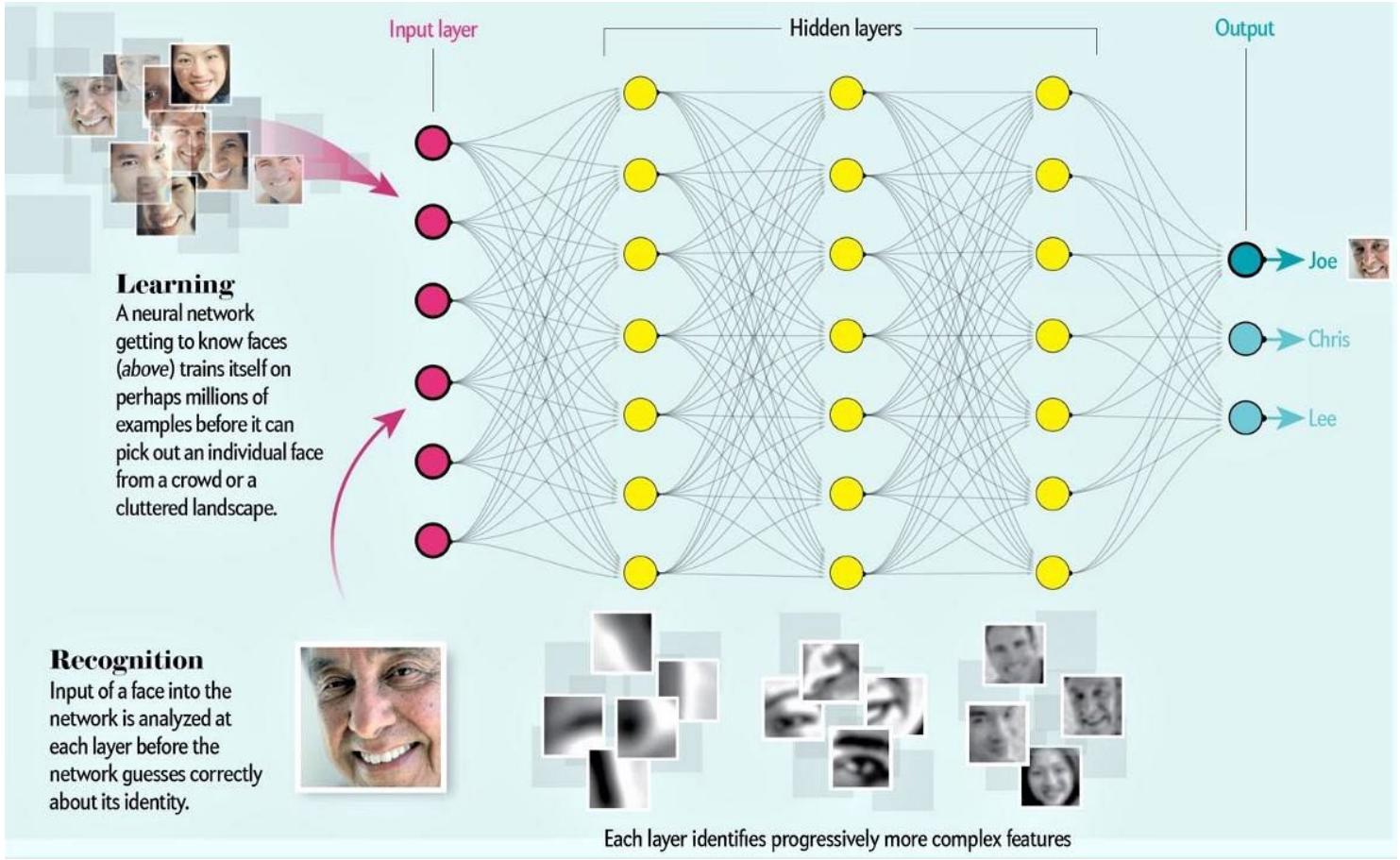
Deep Learning mit Hidden Layer(s)



- **Deep Learning** nutzt neuronale Netze mit **mehreren Schichten** (tiefen Strukturen), um komplexe Muster in Daten zu lernen.
- Es können mehr als eine versteckte Schicht (hidden layer) sein.

Wiederholung SW1!

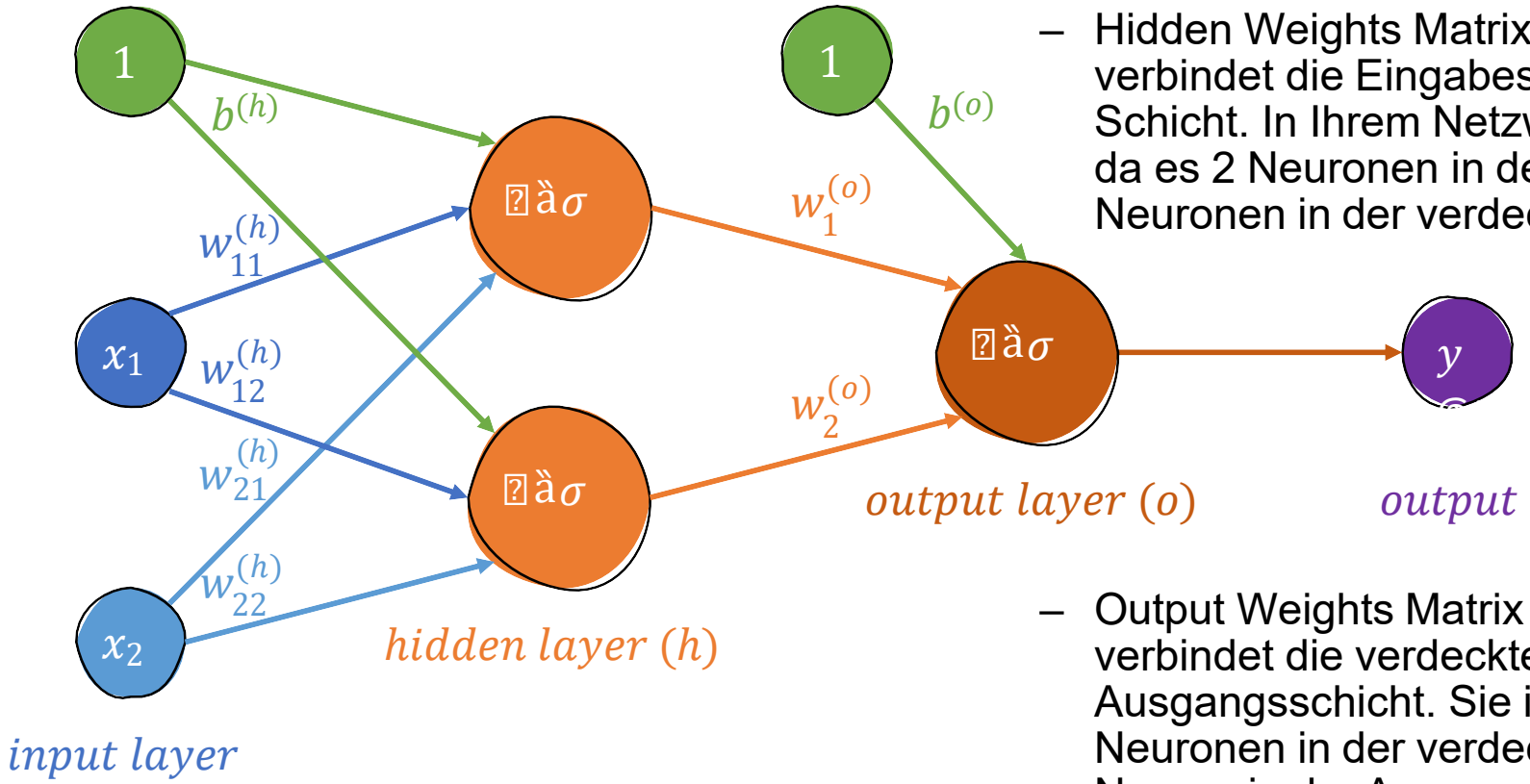
Jede versteckte Ebene identifiziert komplexere Merkmale



https://www.scientificamerican.com/article/springtime-for-ai-the-rise-of-deep-learning/

Gewichtsmatrizen im MLP

$$W^{(h)} = \begin{bmatrix} w_{11}^{(h)} & w_{12}^{(h)} \\ w_{21}^{(h)} & w_{22}^{(h)} \end{bmatrix}$$



– Hidden Weights Matrix $W^{(h)}$: Diese Matrix verbindet die Eingabeschicht mit der verdeckten Schicht. In Ihrem Netzwerk ist es eine 2x2-Matrix, da es 2 Neuronen in der Eingabeschicht und 2 Neuronen in der verdeckten Schicht gibt.

$$W^{(o)} = \begin{bmatrix} w_1^{(o)} \\ w_2^{(o)} \end{bmatrix}$$

– Output Weights Matrix $W^{(o)}$: Diese Matrix verbindet die verdeckte Schicht mit der Ausgangsschicht. Sie ist eine 2x1-Matrix, was 2 Neuronen in der verdeckten Schicht und 1 Neuron in der Ausgangsschicht widerspiegelt.

5. Forward- und Backpropagation

Wiederholung SW1!

Deep Learning Backpropagation

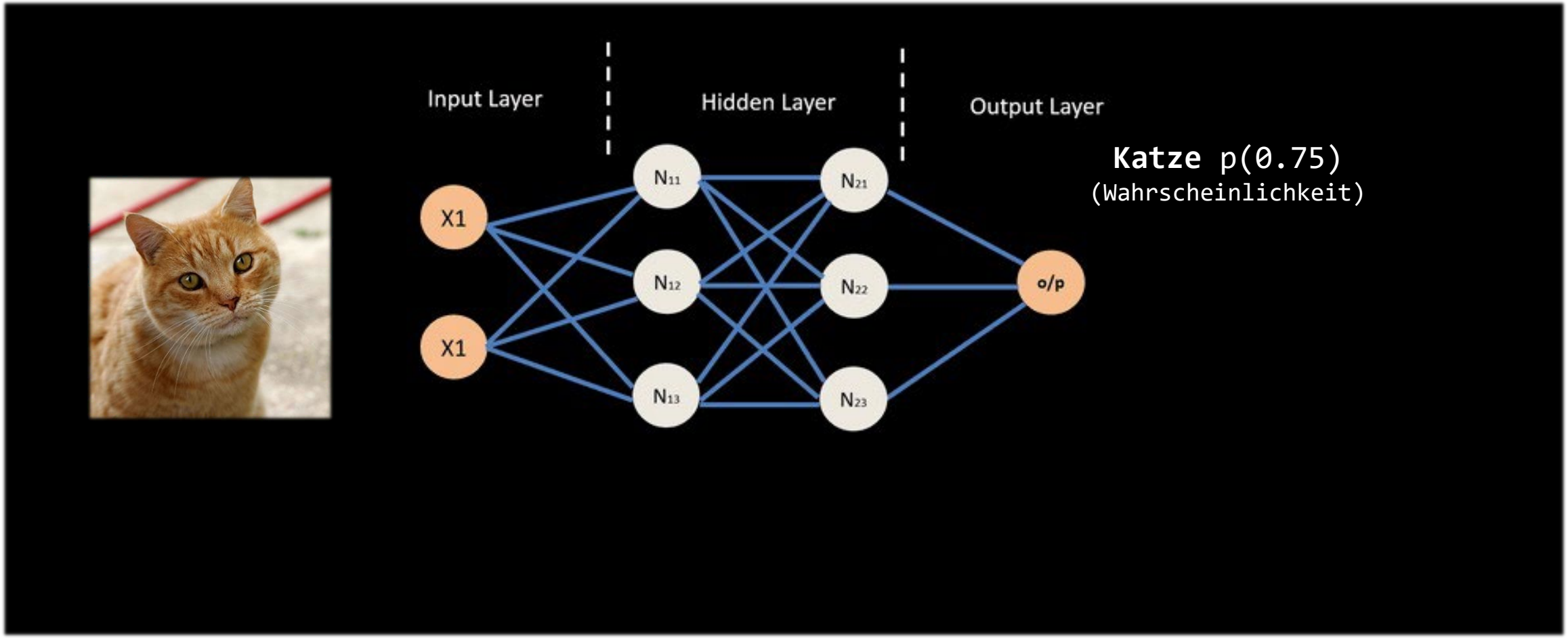
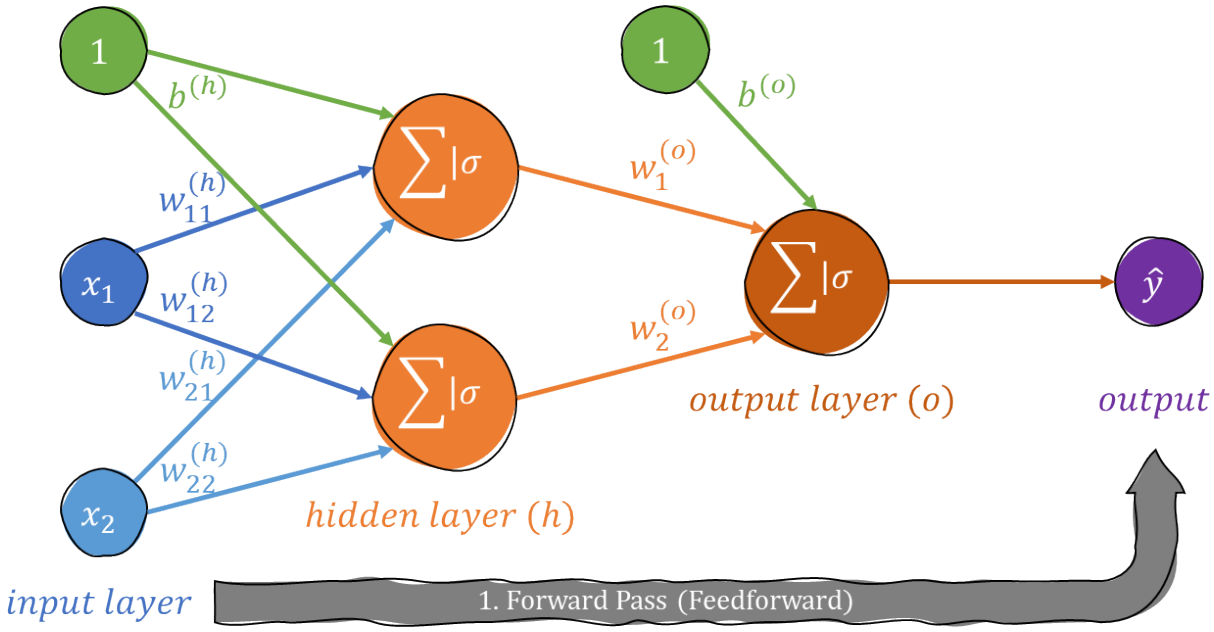


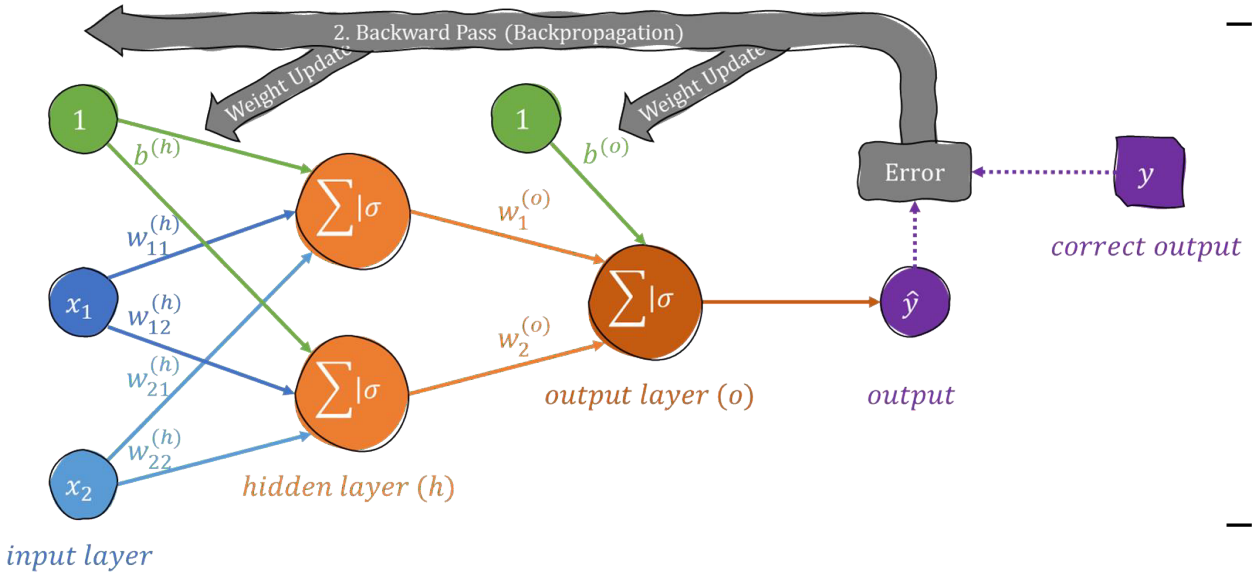
Image: https://machinelearningknowledge.ai/animated-explanation-of-feed-forward-neural-network-architecture/

1. Forward Pass (Feedforward)



- Gewichte und Bias:
  - Gewichte: Bestimmen die Stärke der Verbindung zwischen Neuronen in aufeinanderfolgenden Schichten.
  - Bias: Zusätzliche Parameter, die unabhängig von den Eingabewerten die Aktivierung der Neuronen beeinflussen.
- Forward Pass:
  - Jedes Neuron empfängt Eingabedaten, multipliziert diese mit seinen Gewichten und addiert den Bias.
  - Diese Summe wird dann durch eine Aktivierungsfunktion (z.B. Sigmoid) verarbeitet, um die Ausgabe des Neurons zu generieren.
  - Der Prozess wird wiederholt, bis der Output Layer erreicht ist, der die endgültige Vorhersage des Netzwerks liefert.

2. Backward Pass (Backpropagation)

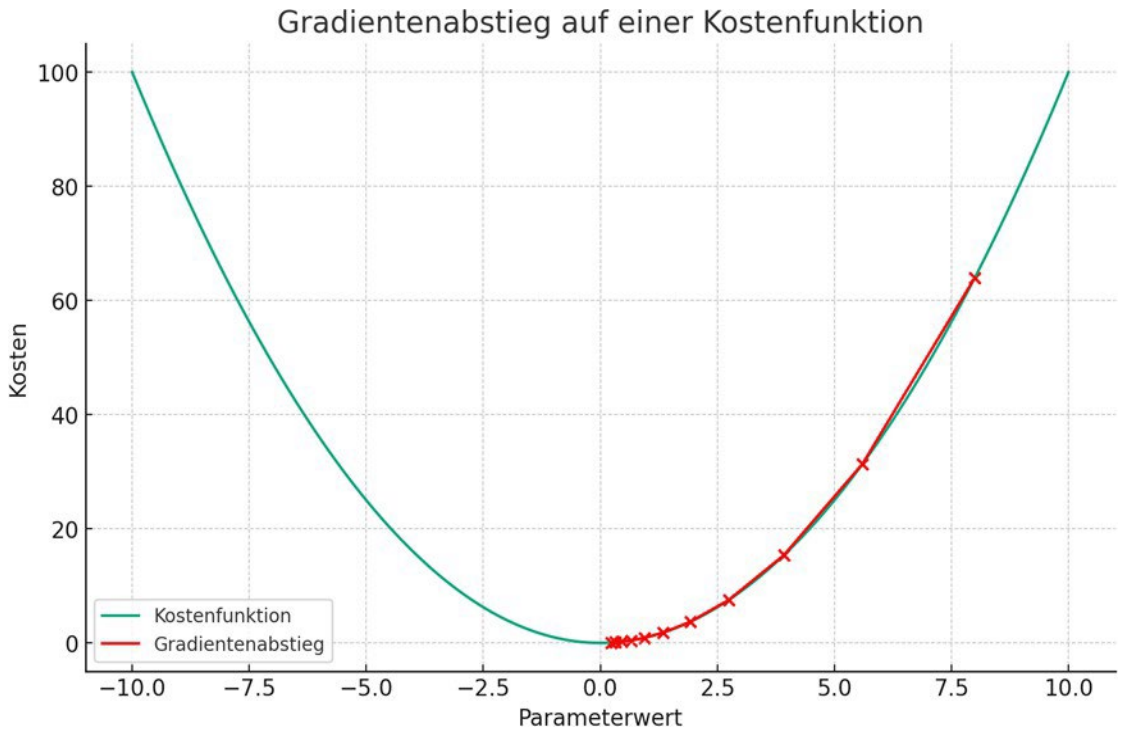


- **Fehlerbewertung**: Vergleich der Netzwerkvorhersagen mit tatsächlichen Werten zur Bestimmung des Vorhersagefehlers.
- **Backpropagation** und **Gradienten**:
  - Verwendung des Fehlers zur Anpassung der Gewichte; dies geschieht durch die Backpropagation, die den Fehler rückwärts durch das Netzwerk führt.
  - Berechnet Gradienten, die die Richtung und Grösse der erforderlichen Anpassungen der Gewichte anzeigen, basierend auf der Ableitung der Aktivierungsfunktion.
- **Gewichts-** und **Biasanpassung**:
  - Schrittweise Anpassung von Gewichten und Bias in Richtung des negativen Gradienten.

6. Gradientenabstieg als «Training» in ML

Gradientenabstieg als «Training» im ML

- Optimierung der Gewichte (w1, w2, ...) und des Bias in Modellen wie logistischer Regression und MLPs.
- Logistische Regression:
  - Gewichte (w1, w2, ...) und Bias bestimmen die Entscheidungsgrenze.
  - Der Gradientenabstieg passt diese Parameter an, um den Vorhersagefehler zu minimieren.
- Multilayer Perceptron (MLP):
  - MLPs haben mehrere Schichten mit Gewichten und Bias.
  - Der Gradientenabstieg wird in jedem Schritt genutzt, um diese Parameter schichtweise zu optimieren.



Error, Cost & Loss

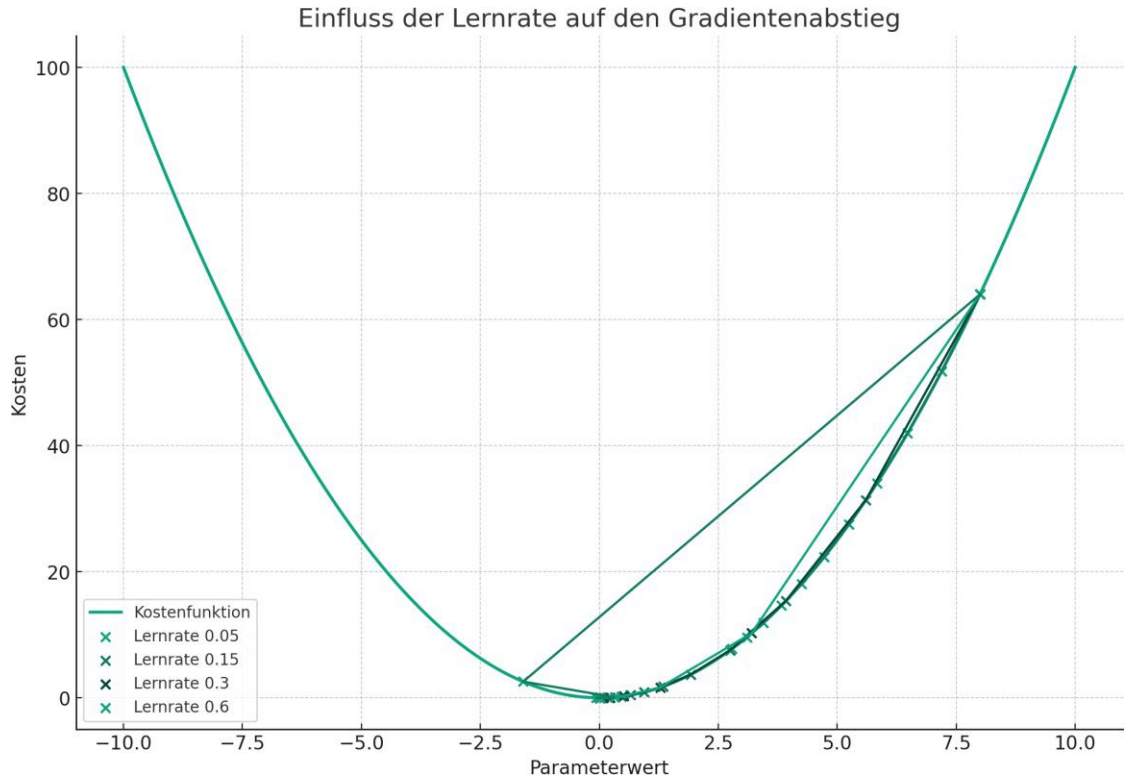
- **Error** (Fehler): Die Differenz zwischen dem vorhergesagten Wert und dem tatsächlichen Wert.
  - Kontext: In binären Vorhersagen (z.B. 0 oder 1) ist dies die Anzahl der falsch klassifizierten Fälle.
  - Beispielvorhersage = 0, tatsächlicher Wert = 1 → Error vorhanden.
- **Cost** (Kosten): Mass für die Gesamtleistung des Modells mit aktuellen Parametern.
  - Funktion: Verwendet wird oft die Log-Loss- oder Cross-Entropy-Funktion.
  - Ziel: Minimierung der Gesamtkosten, die sich aus der Summe der Einzelfehler ergeben.
- **Loss** (Verlust): Bewertet den Fehler oder die Kosten einer einzelnen Vorhersage.
  - Anwendung: Misst, wie gut das Modell bei einer Beobachtung abschneidet.
  - Gesamtbild: **Die Summe der Loss-Werte über alle Trainingsbeispiele ergibt die Gesamtkosten**.
  - Der "Loss" ist sehr ähnlich wie die Kosten, bezieht sich aber in der Regel auf den Fehler oder die Kosten einer einzelnen Vorhersage.
- Zusammenfassend ist der "Error" die Abweichung bei einzelnen Vorhersagen, der "Loss" ist eine Bewertung dieser Abweichung pro Vorhersage oder Datenpunkt, und die "Cost-Funktion" bewertet die Gesamtleistung des Modells über den gesamten Datensatz.

Gradientenabstieg – Kostenfunktion zur Fehlermessung

- Der Gradientenabstieg ist ein iteratives Optimierungsverfahren zum Auffinden des Minimums einer Kostenfunktion.
- Ziel ist es, die Kostenfunktion (oder den Fehler) zu minimieren, die misst, wie gut das Modell die Daten vorhersagt.
  - Die Kostenfunktion misst den Fehler zwischen den Vorhersagen des Modells und den tatsächlichen Werten.
  - Der Gradientenabstieg aktualisiert die Gewichte in Richtung, die den Gesamtfehler minimiert.
- Kreuzentropie-Verlust als Kostenfunktion
  - Die Skalenwerte der einer binären Kreuzentropie, können theoretisch in einem Bereich von 0 bis Unendlich liegen.
  - Die Kreuzentropie misst den Grad der Unterschiedlichkeit zwischen zwei Wahrscheinlichkeitsverteilungen, zwischen den tatsächlichen Labels und den von einem Modell vorhergesagten Labels.

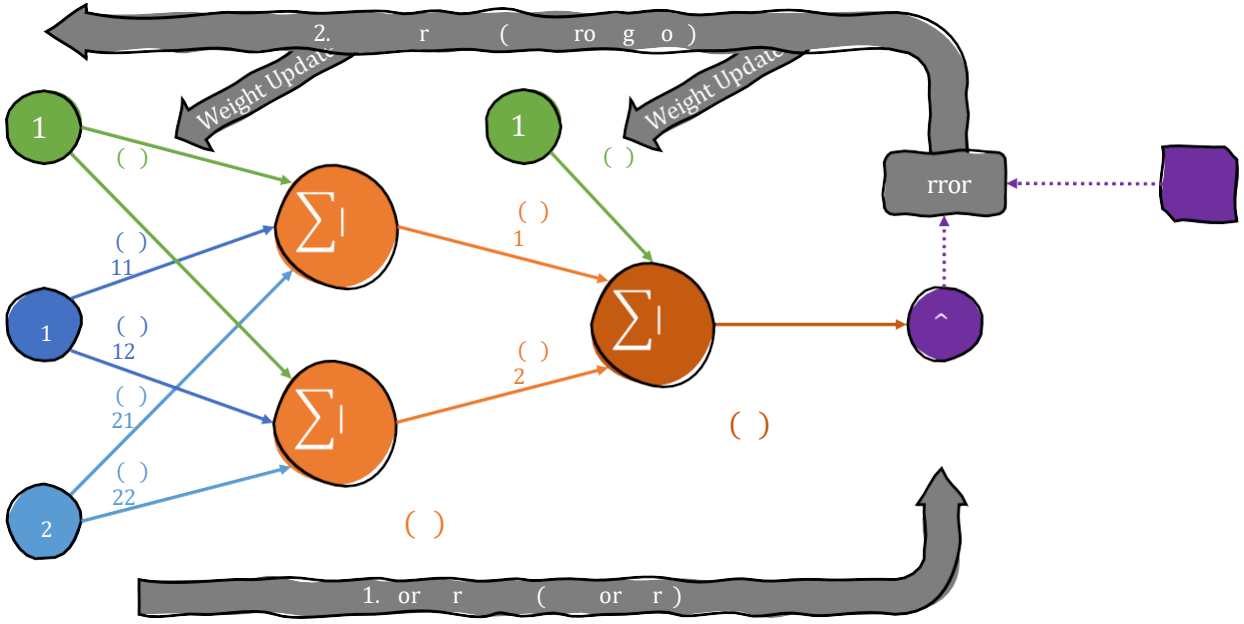
Gradientenabstieg – Iteration und Lernrate

- In jedem Iterationsschritt berechnet der Algorithmus den Gradienten der Fehlerfunktion bezüglich der Modellparameter.
  - Die Parameter werden dann in die entgegengesetzte Richtung des Gradienten angepasst (daher "Abstieg"), um den Fehler zu verringern.
- Die Lernrate bestimmt, wie gross die Schritte bei der Anpassung der Parameter sind.
  - Eine kleinere Lernrate (z.B. 0.05) führt zu kleineren Schritten und daher einer langsameren Konvergenz (Annäherung an das Minimum).
  - Eine moderate Lernrate (z.B. 0.15) ermöglicht eine effiziente Annäherung an das Minimum.
  - Zu hohe Lernraten (z.B. 0.3 und 0.6) können dazu führen, dass der Algorithmus über das Minimum hinausschiesst und eventuell sogar divergiert.
- Konvergenz: Der Prozess wird wiederholt, bis der Algorithmus konvergiert, d.h., bis die Änderungen in den Parametern minimal sind oder eine festgelegte Anzahl von Iterationen erreicht ist.



7. Deep Learning Trainingszyklus

Deep Learning Trainingszyklus

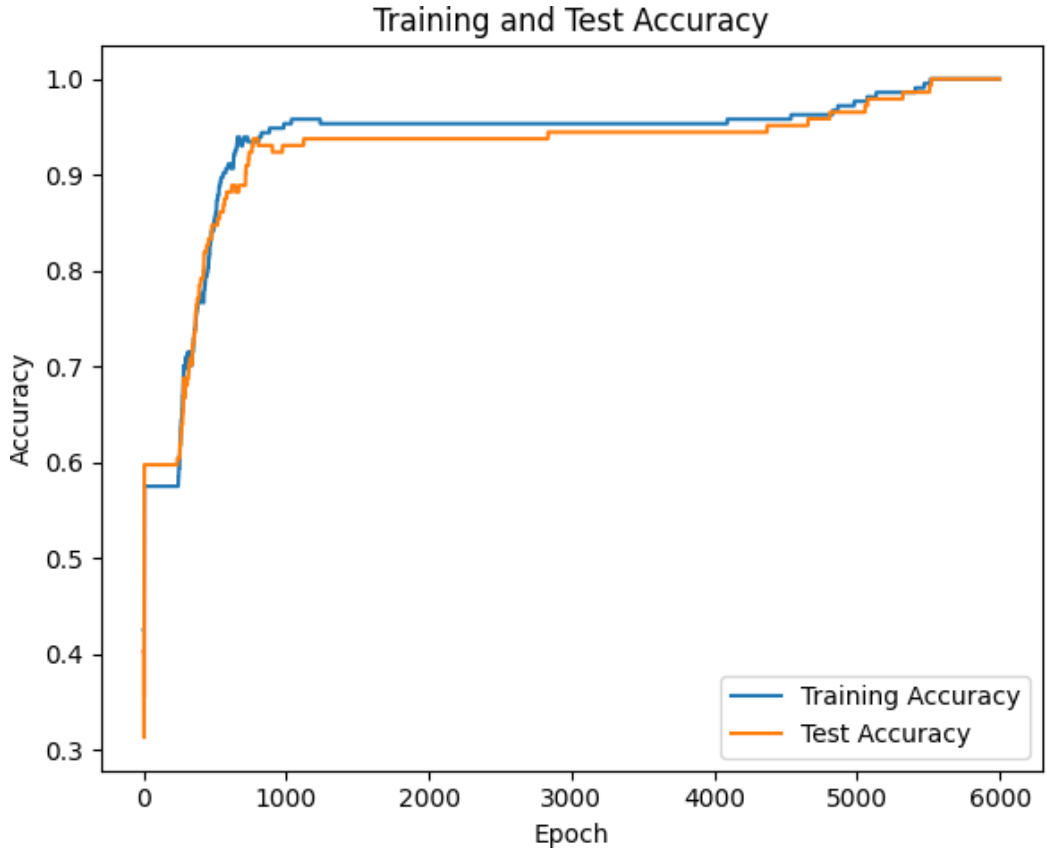


- Der Trainingszyklus umfasst
  - das Durchlaufen von Daten durch das Netzwerk (**Forward Pass**),
  - die Bewertung der Leistung (**Fehlerberechnung**) und
  - die Anpassung der Netzwerkparameter (**Backpropagation**).
- Iterationen, Epochen und Batch Size:
  - **Iterationen**: Jeder Durchlauf einer Teilmenge (Batch) der Trainingsdaten durch das Netzwerk.
  - **Epochen**: Ein vollständiger Durchlauf aller Trainingsdaten. Besteht aus mehreren Iterationen.
  - **Batch Size**: Anzahl der Datenbeispiele, die pro Iteration verarbeitet werden. Beeinflusst die Geschwindigkeit und Stabilität des Trainings.
- Zyklischer Prozess:
  - Das Netzwerk durchläuft mehrfach den gesamten Datensatz (mehrere Epochen), um die Genauigkeit schrittweise zu verbessern und die Fähigkeit zu entwickeln, Muster und Beziehungen in den Daten zu erkennen.

8. Evaluation im MLP

Metriken im MLP (1)

- Genauigkeit (Accuracy): Misst den Prozentsatz der korrekt klassifizierten Instanzen.
  - Genauigkeit (Accuracy) Plot: Stellt die Trainings- und Testgenauigkeit dar.
  - Hier erhöhen sich die Genauigkeitswerte über die Zeit, was zeigt, dass das Modell besser in der Lage ist, Vorhersagen zu treffen.
- Präzision und Recall (Precision and Recall): Wichtig für unausgewogene Datensätze. Präzision misst die Genauigkeit der positiven Vorhersagen, während Recall die Fähigkeit misst, tatsächlich positive Fälle zu identifizieren.
- F1-Score: Kombiniert Präzision und Recall in einer einzigen Metrik. Besonders nützlich, wenn ein Gleichgewicht zwischen Präzision und Recall erforderlich ist.
- Konfusionsmatrix (Confusion Matrix) und ROC-Kurve (Receiver Operating Characteristic Curve



Metriken im MLP (2)

- Verlustfunktion (Loss Function): Zeigt, wie gut das MLP während des Trainings abschneidet. Das Ziel ist ein Loss = 0.
  - Häufig verwendet sind Mean Squared Error (MSE) für Regression und Cross-Entropy (Kreuzentropie) für Klassifikation.
- Lernkurven (Learning Curves): Zeigen die Veränderung der Trainings- und Validierungsverluste über die Zeit.
  - Verlust (Loss) Plot: Zeigt den Trainings- und Testverlust über die Epochen.
    - In diesem Plot sinken beide Verlustkurven im Laufe der Zeit, was ein Indikator für das Lernen des Modells ist.
    - Hilfreich, um Über- oder Unteranpassung (Overfitting/Underfitting) zu identifizieren.