1 Machine Learning and Neural Networks

1.1 Introduction

1.1.1 Grundlagen des Lernens

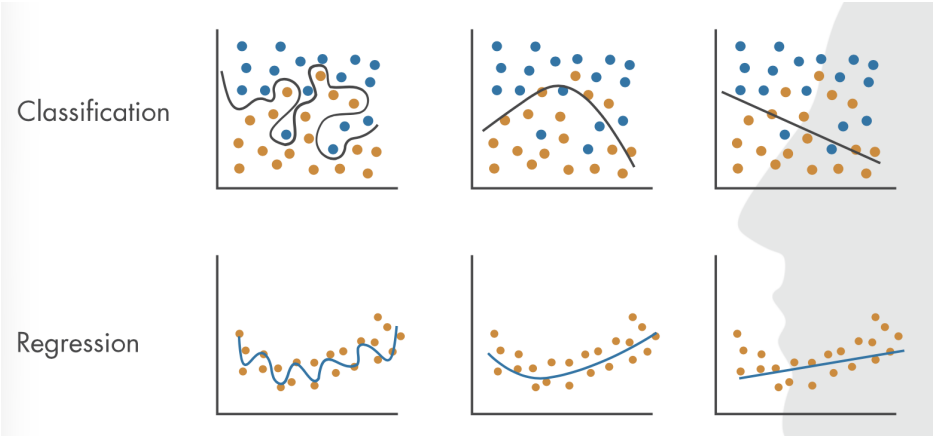
Lernen ist ein fundamentaler Prozess für intelligente Agenten, um in unbekannten Umgebungen zu agieren und die Leistung basierend auf Erfahrungen zu verbessern.

Definition: Machine Learning (Tom Mitchell, 1997)

Ein Computerprogramm lernt aus Erfahrung E im Hinblick auf eine Klasse von Aufgaben T und ein Leistungsmaß P, wenn sich seine Leistung bei Aufgaben in T, gemessen an P, mit der Erfahrung E verbessert.

Induktives Lernen Induktives Lernen ist die einfachste Form des Lernens, bei der eine Funktion aus Beispielen abgeleitet wird.

- **Ziel:** Finden einer Hypothese h, die die unbekannte Zielfunktion f approximiert ($h \approx f$).
- **Gegeben:** Ein Trainingsset von Beispielen $(x, f(x))$, wobei x der Input und $f(x)$ das Label (Target) ist.
- **Konsistenz:** Eine Hypothese h ist konsistent, wenn sie für alle Trainingsbeispiele mit f übereinstimmt.



Ockham's Razor

Die beste Erklärung ist die einfachste Erklärung, die zu den Daten passt. Im Kontext von Machine Learning bedeutet dies, dass bei gleicher Genauigkeit einfachere Modelle bevorzugt werden sollten, um Overfitting zu vermeiden.

1.1.2 Machine Learning vs. Traditionelle Programmierung

- **Traditionelle Programmierung:** Daten + Programm → Output. Der Mensch formuliert die Regeln explizit.
- **Machine Learning:** Daten + Output → Programm (Modell). Der Algorithmus lernt die Regeln ("Rule Set") aus den Daten.

1.1.3 Arten des Lernens

- **Supervised Learning (Überwachtes Lernen):** Das Modell lernt aus gelabelten Daten (Input-Output-Paare). Ziel ist es, eine Abbildung von Eingabe zu Ausgabe zu lernen.

  - **Regression:** Vorhersage eines kontinuierlichen Wertes (z.B. Temperatur).
  - **Klassifikation:** Vorhersage einer diskreten Klassenbezeichnung (z.B. "Regen" oder "Sonne").
- **Unsupervised Learning (Unüberwachtes Lernen)**: Das Modell lernt aus ungelabelten Daten. Ziel ist es, Muster, Strukturen oder Ähnlichkeiten in den Daten zu finden (z.B. Clustering).
- **Reinforcement Learning (Bestärkendes Lernen)**: Ein Agent lernt durch Interaktion mit der Umgebung und erhält positives oder negatives Feedback (Reward).
- **Semi-supervised Learning:** Eine Kombination, bei der nur ein kleiner Teil der Daten gelabelt ist.

1.1.4 Datenrepräsentation und Feature Engineering

Algorithmen benötigen Daten in einer verarbeitbaren Form, meist als **Feature-Vektoren** im $\mathbb{R}^n$.

- **Feature Engineering:** Der Prozess der Auswahl, Manipulation und Transformation von Rohdaten in Features, die für das Modell nutzbar sind.
- **Prinzip "Garbage In, Garbage Out":** Schlechte Datenqualität oder irrelevante Features führen zu schlechten Modellen, unabhängig von der Qualität des Algorithmus.
- **Preprocessing:** Wichtige Schritte umfassen den Umgang mit Ausreißern, fehlenden Werten und Bias in den Daten.

1.1.5 Modell-Evaluierung

Um die Qualität eines Modells zu messen, wird eine Metrik benötigt, die zum Ziel passt (z.B. Accuracy, Precision, Recall). Für Regressionsprobleme wird häufig der **Mean Squared Error (MSE)** verwendet.

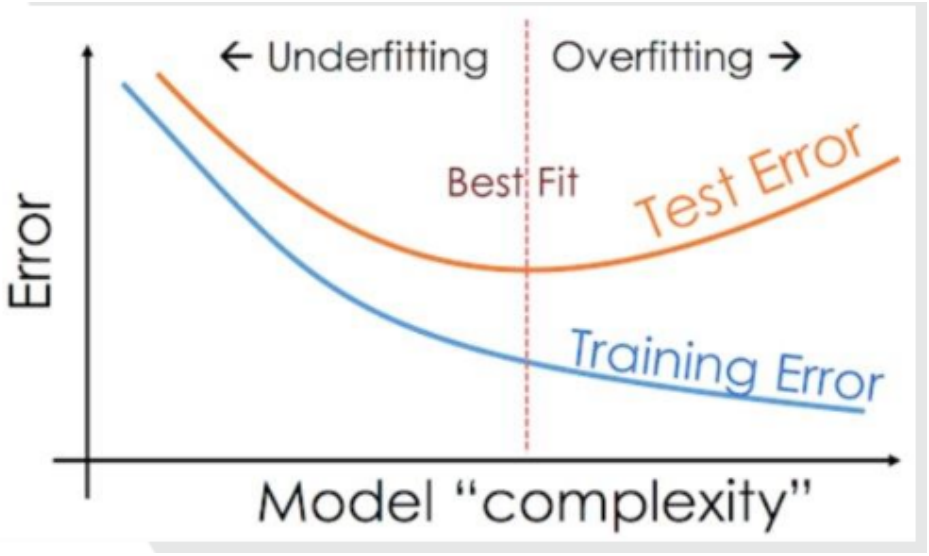
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

wobei y_i der wahre Wert und $\hat{y}_i$ der vorhergesagte Wert ist.

Generalisierung und Overfitting Das Ziel von Machine Learning ist **Generalisierung** (Leistung auf neuen, unbekannten Daten), nicht bloßes Auswendiglernen (**Memorization**).

- **Overfitting**: Das Modell hat die Trainingsdaten "auswendig gelernt" (inklusive Rauschen) und performt schlecht auf neuen Daten. Das Modell ist zu komplex.
- **Underfitting**: Das Modell ist zu einfach, um die zugrunde liegende Struktur der Daten zu erfassen.

Lösung (Train-Test-Split): Die Daten werden in ein **Training Set** (zum Lernen) und ein **Test Set** (zur Evaluation) unterteilt. Overfitting liegt vor, wenn der Trainingsfehler niedrig, der Testfehler aber hoch ist.



1.1.6 Künstliche Neuronale Netze (ANNs)

Neuronale Netze sind inspiriert von biologischen Gehirnen, versuchen diese aber nicht exakt nachzubilden. Sie sind besonders effektiv bei komplexen Inputs wie Bildern oder Sprache (Deep Learning).

Das Perzeptron (Künstliches Neuron) Das Perzeptron ist die Grundeinheit eines neuronalen Netzes. Es berechnet eine gewichtete Summe der Eingaben, addiert einen Bias und wendet eine Aktivierungsfunktion an.

$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i w_i\right)$$

- x_i: Inputs
- w_i: Gewichte (bestimmen Stärke und Vorzeichen der Verbindung)
- w_0: Bias (Verschiebung der Aktivierungsschwelle)
- Σ: Lineare Kombination (Summe)
- g: Nicht-lineare Aktivierungsfunktion
- $\hat{y}$: Output

Aktivierungsfunktionen Aktivierungsfunktionen entscheiden, ob ein Neuron "feuert". Sie sind essenziell, um **Nicht-Linearität** in das Netzwerk zu bringen. Ohne sie wäre jedes neuronale Netz, egal wie tief, mathematisch äquivalent zu einer einfachen linearen Regression.

Gängige Funktionen:

- **Sigmoid:** $\sigma(x) = \frac{1}{1+e^{-x}}$ (Wertebereich 0 bis 1, oft für Wahrscheinlichkeiten).
- **ReLU (Rectified Linear Unit):** $\max(0, x)$ (Löst das Problem verschwindender Gradienten, sehr verbreitet).
- **Tanh:** $\tanh(x)$ (Wertebereich -1 bis 1).

Multilayer Perceptron (MLP) Ein MLP besteht aus mehreren Schichten von Neuronen:

1. **Input Layer:** Nimmt die Feature-Vektoren auf.
2. **Hidden Layers:** Verarbeiten Informationen, extrahieren Features.
3. **Output Layer:** Liefert das Endergebnis.

Die Information fließt unidirektional von Input zu Output (**Forward Propagation**).

1.1.7 Training neuronaler Netze

Das Training ist ein Optimierungsprozess, um die Gewichte W so anzupassen, dass der Fehler (Loss) minimiert wird.

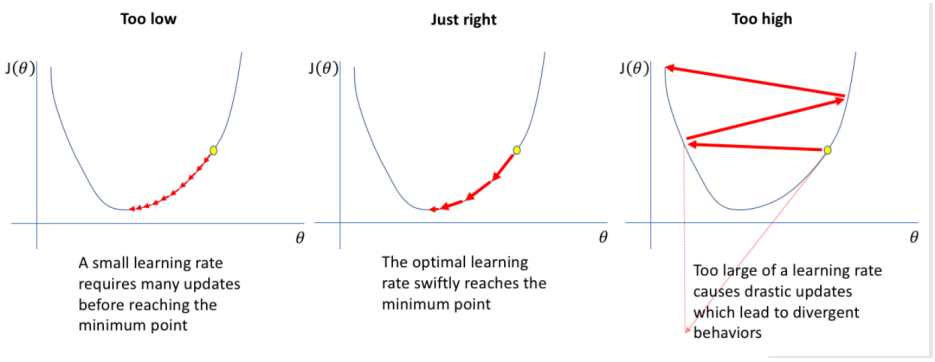
Loss Function (Verlustfunktion) Die Loss Function $J(W)$ misst die Diskrepanz zwischen Vorhersage $\hat{y}$ und wahrem Label y. Ziel ist:

$$W^* = \text{argmin}_W \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x^{(i)}; W), y^{(i)})$$

Gradient Descent (Gradientenabstieg) Ein iterativer Algorithmus zur Minimierung der Loss Function. Man bewegt sich im "Gewichtsraum" in Richtung des steilsten Abstiegs.

$$W \leftarrow W - \alpha \frac{\partial J(W)}{\partial W}$$

- $\frac{\partial J(W)}{\partial W}$: Der Gradient (Steigung) des Fehlers bezüglich der Gewichte.
- α (**Learning Rate**): Schrittweite.
  - Zu klein: Konvergenz dauert sehr lange.
  - Zu groß: Gefahr der Divergenz (man springt über das Minimum).



Backpropagation Backpropagation ist der Algorithmus zur effizienten Berechnung der Gradienten $\frac{\partial J(W)}{\partial W}$ durch Anwendung der **Kettenregel**. Der Fehler wird vom Output Layer rückwärts durch das Netz propagiert.

Beispielrechnung (Kettenregel): Gegeben sei ein einfaches Netz $x \to z \to \hat{y}$ und Loss $J(W) = (\hat{y} - y)^2$. Der Einfluss eines Gewichts w auf den Fehler ist:

$$\frac{\partial J(W)}{\partial w} = \frac{\partial J(W)}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w}$$

1. Wie ändert sich der Fehler mit dem Output? $(\frac{\partial J}{\partial \hat{y}})$
2. Wie ändert sich der Output mit der Aktivierung? $(\frac{\partial \hat{y}}{\partial z} = g'(z))$
3. Wie ändert sich die Aktivierung mit dem Gewicht? $(\frac{\partial z}{\partial w} = x)$

1.1.8 Regularisierung

Methoden zur Vermeidung von Overfitting in neuronalen Netzen:

- **Early Stopping:** Training beenden, sobald der Fehler auf dem Test-Set wieder ansteigt.
- **Dropout:** Zufälliges Deaktivieren von Neuronen während des Trainings, um Robustheit zu erzwingen.
- **Data Augmentation:** Künstliche Vergrößerung des Datensatzes (z.B. durch Rauschen oder Rotation bei Bildern).

1.1.9 Convolutional Neural Networks (CNNs)

CNNs sind spezialisierte Architekturen für grid-artige Daten (z.B. Bilder).

- **Convolutional Layers:** Verwenden Filter, um lokale Features (Kanten, Formen) zu extrahieren.
- **Pooling Layers:** Reduzieren die Dimensionalität (z.B. Max Pooling) und machen das Modell robuster gegenüber Verschiebungen.
- Tiefe Schichten erkennen komplexere Objekte (Hierarchie der Features).