1 Einleitung

Spricht man mit Personen aus anderen Fachbereichen, betont fast jeder die Relevanz seines Gebiets. Biologen erforschen das Leben, Geologen die Erde, und auch Wirtschaftswissenschaftler, Mathematiker oder Sozialwissenschaftler sind oft überzeugt, dass ihre Disziplin zu den wichtigsten zählt.
In der Seminarreihe „Mathematics of Machine Learning“ behandeln wir ein Thema, das diesen Namen nicht nur verdient, sondern dessen Bedeutung in Wissenschaft und Gesellschaft gleichermaßen anerkannt ist.
In den letzten Jahren war ihr Einfluss konkurrenzlos. Nicht nur Large Language Modelle wie ChatGPT, sondern unzählige Anwendungen in wissenschaftlichen Studien, selbstfahrende Autos oder Optimierungen in der Wirtschaft verändert unser Dasein nachhaltig. Dass diese Revolution erst vergleichsweise spät begann, zeigt, wie komplex sie ist. Im Folgenden wird ein Fundament geschaffen, dass es ermöglichen soll, eine mathematische Theorie über Machine Learning zu entwickeln.

2 Formaler Rahmen

Da maschinelles Lernen den menschlichen Lernprozess nachahmen soll, lohnt es sich zunächst, diesen zu analysieren und zu reflektieren. Das wird uns ermöglichen, diese Konzepte auf Maschinen zu übertragen. Ein wichtiger Schritt ist das Differenzieren von verschiedenen Problemarten.

Definition 1.1. Wir unterscheiden zwischen drei unterschiedlichen Problemarten.

- *Supervised Learning* bezeichnet das Lösen von Problemen mit *labelled examples*, also gekennzeichneten Beispielen. Ein Beispiel hierfür ist Zeichenerkennung.
- *Unsupervised Learning* bezeichnet das Lösen von Problemen mit *unlabelled examples*, also Beispiele und Kennzeichnung. Ein Beispiel hierfür wäre das *Clustering* also die Clusterbildung von Musik in verschiedene Stimmungen wie z.B. „entspannt“.
- Beim *Reinforcement Learning* werden die Beispiele durch Interaktion mit der Umwelt generiert. Das geschieht beispielsweise beim Autofahren.

Für jede dieser Problemarten existiert eine ausgereifte mathematische Theorie, die weiterhin Gegenstand aktueller Forschung ist. In dieser Ausarbeitung konzentrieren wir uns auf *Supervised Learning*, ohne dies künftig stets ausdrücklich zu betonen.

Definition 1.2. Es gibt zwei verschiedene Arten von Problemen im Supervised Learning, die später noch rigoroser definiert werden.

- *Classification problems* beschäftigen sich mit der Vorhersage eine *categorical variable*. Ein bereits bekanntes Beispiel ist die Zeichenerkennung.
- *Regression Tasks* beschäftigen sich mit der Vorhersage von numerischen Werten, bei denen Abstände minimiert werden sollen. Ein Beispiel hiervon wäre die Vorhersage des Wetters.

Definition 1.2.1. Supervised Learning besteht aus

1. Einem *Input Space* $\mathcal{X} \subseteq \mathbb{R}^{n_{\text{inp}}}$ und einen *Output Space* $\mathcal{Y} \subseteq \mathbb{R}^{n_{\text{out}}}$ mit $n_{\text{inp}}, n_{\text{out}} \in \mathbb{N}$.
2. Eine unbekannte Abbildung $f : \mathcal{X} \to \mathcal{Y}$, die wir approximieren möchten.
3. Eine Wahrscheinlichkeitsverteilung D auf $\mathcal{X}$.
4. Ein *dataset* $\mathcal{S} = \{(x_i, y_i) \mid i = 1, \dots, m\}$ wobei $f(x_i) = y_i$ und die x_i u.i.v. aus $\mathcal{X}$ vermittels D gezogen werden. S bezeichnet wir als Trainingsdaten.
5. Eine *Hypothesis Class* $\mathcal{H}$, die eine Menge von Funktionen ist, die voraussichtlich f oder eine gute Approximation von f enthält. Ein Element $h \in \mathcal{H}$ wird *Predictor* genannt.
6. Eine *Loss Function* $\ell : Y \times Y \to \mathbb{R}_+$, mit deren Hilfe Abstände auf Y berechnen können. ℓ soll positiv definit und symmetrisch sein.
7. Ein *Training Algorithm* (Siehe **Definition 1.2.9**).

Bemerkung 1.2.3. Wir nehmen an, das *dataset* sei perfekt. Das ist aber aufgrund von Messungenauigkeiten oder fehlenden Daten oftmals falsch. Es gilt dann viel mehr $y_i = f(x_i) + \epsilon_i$, wobei ϵ_i ein zufälliger Vektor ist. Sehr häufig ist ϵ_i normalverteilt mit Mittelwert 0.

Bemerkung 1.2.4. Eine *Hypothesis Class* $\mathcal{H} \subseteq \{h : \mathcal{X} \to \mathcal{Y}\}$ ist eine Menge von Funktionen. Wir verwenden oftmals den Begriff *Parametric model* $\mathcal{W} \subseteq \mathbb{R}^p$, um $\mathcal{H}$ vollständig zu charakterisieren. Das heißt, dass jede Abbildung $h \in \mathcal{H}$ durch die Parameter $w = (w_1, \ldots, w_p) \in \mathcal{W}$ bestimmt wird. Formal gibt es eine Abbildung

$$\Phi : \mathcal{W} \to \mathcal{H} \quad w \mapsto f_w.$$

Hierbei ist $f_w : \mathcal{X} \to \mathcal{Y}$ die durch die Parameter w bestimmte Funktion. Deshalb schreibt man auch

$$\mathcal{H} = \{f_w \mid w \in \mathcal{W}\} = \Phi(\mathcal{W}).$$

Die Abbildung Φ ist normalerweise intuitiv klar und wird nicht explizit erwähnt.
Beispielsweise lässt sich die Menge

$$\mathcal{H}_{\text{lin}} := \{h : x \mapsto w_1 + w_2 x \mid w_1, w_2 \in \mathbb{R}\}$$

durch das *Parametric model* $\mathcal{W} = \mathbb{R}^2$ vollständig charakterisiern.

Definition 1.2.6. Ein Problem heißt *Classification problem* (Vgl. **Definition 1.2**) wenn $\mathcal{Y}$ abzählbar ist und ℓ die diskrete Metrik ist. Wenn $\mathcal{Y}$ kontinuierlich ist spricht man von einem *Regression Task* (Vgl. **Definition 1.2**).

Definition 1.2.8. Sei $h \in \mathcal{H}$ ein *Predictor*, dann ist der *Generalized Error* definiert als

$$R_D(h) := \mathbb{E}_{x \sim D}[\ell(h(x), f(x))].$$

Definition 1.2.8.1. Die Notation $\mathbb{E}_{x \sim D}$ suggeriert, dass nach x mit dem Maß D integriert wird. $R_d(h)$ bezeichnet also den erwarteten Fehler, den ein *Predictor* mit der unbekannten Funktion f haben wird, wenn wir zufällig einen *Input* ziehen. Das Ziel von *Supervised Learning* ist das Lösen von

$$\operatorname*{argmin}_{h \in \mathcal{H}} R_D(h).$$

Da f und D unbekannt sind, ist dieses Minimierungsproblem leider nicht lösbar. Wir versuchen im Folgenden dennoch mit Hilfe von $\mathcal{S}$ geeignete $h \in \mathcal{H}$ zu finden, indem wir $R_D(h)$ approximieren.

Definition 1.2.8.2. Unter *Empirical Risk Minimization* versteht man die Optimierung von

$$\operatorname*{argmin}_{h \in \mathcal{H}} L(h) := \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \ell(h(x_i), y_i).$$

Die Abbildung $L : \mathcal{H} \to \mathbb{R}_+$ wird *Empirical Error* genannt.

Definition 1.2.8.3. $L(h)$ ist der durchschnittliche Fehler des *Predictors* $h \in H$ verglichen mit den Testdaten $\mathcal{S}$. Die Hoffnung ist, dass folgende, leider im Allgemeinen falsche, Implikation hält:

$$\text{„}L(h) \text{ klein} \implies R_D(h) \text{ klein“}.$$

Wir werden nun untersuchen, warum die Implikation nicht korrekt ist und wie wir das Problem angehen können.

Definition 1.2.9. Sei $\mathcal{S}$ ein *dataset* und $\mathcal{H}$ eine *Hypothesis Class* mit zugehörigen *Parameter Space* $\mathcal{W}$, dann nennen wir eine Abbildung

$$\mathcal{A} = \mathcal{A}_{\mathcal{H},\mathcal{S}} : \mathcal{W} \to \mathcal{W}$$

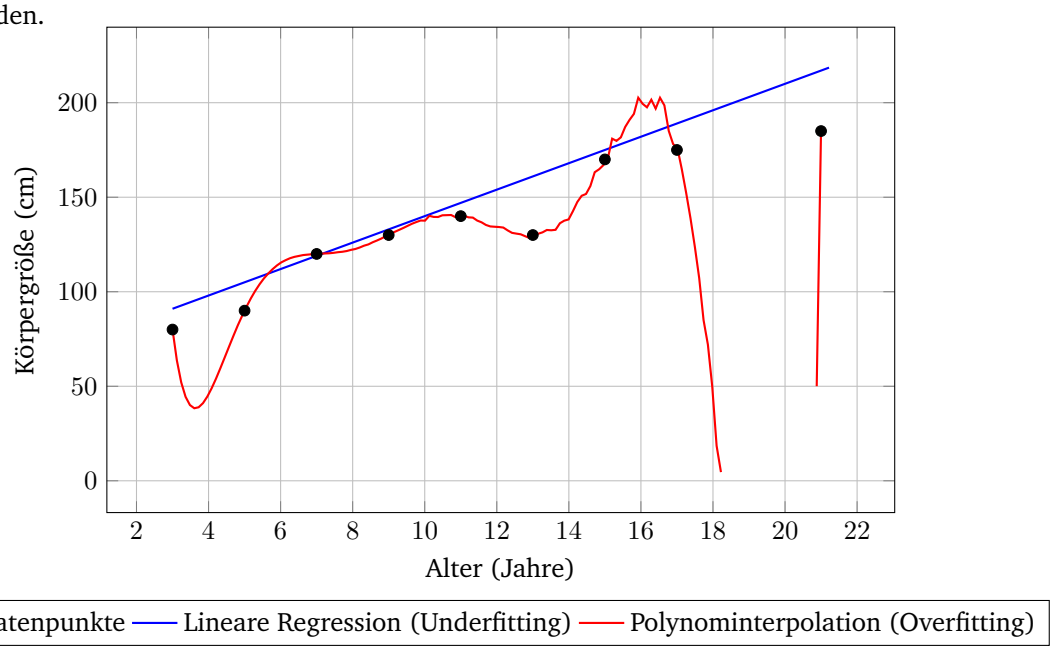
einen *Training Algorithm*.

Bemerkung 1.2.9.1. Die Idee hinter dem *Training Algorithm* ist es, einen initialen Parameter $w \in \mathcal{W}$ zu wählen und anschließend trainierte Parameter $\mathcal{A}(w)$ zu erhalten, sodass $f_{\mathcal{A}(w)}$ einen kleinen *Empirical Error* hat.

Definition 1.2.10. Parameter von $\mathcal{H}$ und $\mathcal{A}$, die invariant unter $\mathcal{A}$ sind heißen *hyperparameter*.

3 Die Wahl von $\mathcal{H}$

Es ist eine Kunst, $\mathcal{H}$ so geschickt zu wählen, dass man gute Ergebnisse erzielt. Angenommen, wir möchten den Zusammenhang zwischen Alter und Körpergröße analysieren und verfügen dafür über ein kleines *dataset* $\mathcal{S}$. Unterstellen wir einen linearen Zusammenhang, können wir eine einfache *Hypothesis Class* $\mathcal{H}_{\text{lin}}$ (Vgl. **Bem 1.2.4**) wählen. Dann wird unser Modell vermutlich nicht komplex genug, um die Beziehung adäquat zu erfassen. Erlauben wir dagegen Polynome in $\mathcal{H}$, deren Grad größer als $|\mathcal{S}|$ ist, werden die Trainingsdaten lediglich interpoliert – eine Verallgemeinerung findet dann nicht statt. Im ersten Fall spricht man von *underfitting* und im zweiten Fall spricht man von *overfitting*. Der folgende Graph zeigt eine Demonstration dieses Phänomens mit ausgedachten Daten. Das „Zittern“ vom Overfitting liegt an numerischen Ungenauigkeiten und kann ignoriert werden.



Definition 1.2.10. Sei $h \in \mathcal{H}$ ein *Predictor*, dann definiere den *Generalized Gap* als

$$R_D(h) - L(h).$$

Wir versuchen den *Generalized Gap* für einen *Predictor* $h \in \mathcal{H}$ zu schätzen. Hierfür teilen wir $\mathcal{S} = \mathcal{S}_{\text{train}} \dot{\cup} \mathcal{S}_{\text{test}}$. O.B.d.A ist

$$\mathcal{S}_{\text{train}} = \{(x_i, y_i) \mid i = 1, \dots, m'\}$$

mit $m' < m$. Nun kann man auf $\mathcal{S}_{\text{train}}$ trainieren und $R_D(h)$ approximieren vermittels

$$\frac{1}{m - m'} \sum_{i=m'+1}^{m} \ell(h(x_i), y_i).$$

Das konvergiert im Fall $m' - m \to \infty$ nach einer Form des Gesetzes der großen Zahlen gegen $R_D(h)$, weil unsere Daten u.i.v. gezogen sind.

Definition 1.2.10. Mit dem jetzigen Wissen lässt sich *underfitting* und *overfitting* geschickt definieren.

- Wir sprechen von *underfitting*, wenn $\inf_{h \in \mathcal{H}} L(h)$ groß ist. In diesem Fall ist die *Hypothesis Class* vermutlich nicht komplex genug für das gegebene Problem.
- Wir sprechen von *overfitting*, wenn ein *Predictor* $h \in \mathcal{H}$ einen kleinen *Empirical Error* $L(h)$, aber einen großen *Generalized Error* $R_D(h)$ aufweist. In diesem Fall ist unser *Predictor* zu kompliziert und passt sich nur den Trainingsdaten an, statt den Zusammenhang zu verallgemeinern.

In **Definiton 1.2.10** wurde eine Möglichkeit vorgestellt, $R_D(h)$ zu approximieren und *overfitting* zu erkennen. Hat man hingegen mehrere *Hypothesis Classes* $\mathcal{H}_1, \dots, \mathcal{H}_n$, so wäre es denkbar die jede dieser $\mathcal{H}_i$ auf $\mathcal{S}_{\text{train}}$ zu trainieren und die Werte von $R_D(h)$ mit $\mathcal{S}_{\text{test}}$ zu approximieren (Vgl. **Definition 1.2.10**). Nun ließen sich die verschiedenen Approximationen vergleichen und das geeignetste Modell wählen. Diese naive Herangehensweise könnte jedoch wiederum zu *overfitting* des *test sets* führen. Dieses Verhalten versuchen wir im Folgenden zu umgehen.

3.1 Validation Set

Seien $\mathcal{H}_1, \ldots, \mathcal{H}_n$ *Hypothesis Classes* und $\mathcal{S}$ ein *dataset*. Folge nun dem folgenden Verfahren.

1. Unterteile
$$\mathcal{S} = \mathcal{S}_{\text{train}} \dot\cup \mathcal{S}_{\text{val}} \dot\cup \mathcal{S}_{\text{test}}.$$
2. Trainiere jede dieser n *Hypothesis Classes* auf $\mathcal{S}_{\text{train}}$. Man erhält für jede Klasse einen *Predictor* $h_i \in \mathcal{H}_i$.
3. Vergleiche nun deren Performance auf S_{val} durch die Approximation von
$$\frac{1}{|\mathcal{S}_{\text{val}}|} \sum_{(x,y)\in\mathcal{S}_{\text{val}}} \ell(h_i(x), y). \tag{1}$$
Wähle anschließend das Modell $\mathcal{H}_i$, bei der **(1)** minimal ist.
4. Trainiere diese *Hypothesis Class* $\mathcal{H}_i$ auf $S_{\text{train}} \dot\cup S_{\text{val}}$ und erhalte ein neuen, nun auf einem größeren Datensatz trainierten, *Predictor* $h \in \mathcal{H}_i$.
5. Prüfe die *Goodness* dieses *Predictors* h, also den Term
$$\frac{1}{|S_{\text{test}}|} \sum_{(x,y)\in S_{\text{test}}} \ell(h_i(x), y). \tag{2}$$
Hiermit wird $R_D(h)$ approximiert.

Bemerkung 1.3.1.1. Die Unterteilung
$$\mathcal{S} = \mathcal{S}_{\text{train}} \dot\cup \mathcal{S}_{\text{val}} \dot\cup \mathcal{S}_{\text{test}}$$
ließe sich auch durch einen menschlichen Lernprozess vergleichbar mit einer Klausurvorbereitung veranschaulichen. Die Menge $\mathcal{S}_{\text{train}}$ nimmt hier die Position von Übungsaufgaben ein, mit denen eingangs gelernt wird. Die Menge $\mathcal{S}_{\text{val}}$ beschreibt Altklausuren, mit denen man überprüft, ob man das Wissen der Übungsaufgaben vertieft hat und bereit für die Klausur ist. Die Menge $\mathcal{S}_{\text{test}}$ ist dann die finale Klausur, bei der das Wissen bewertet wird.

Bemerkung 1.3.1.2. Ein Nachteil dieses Verfahren ist, dass für jede *Hypothesis Class* $\mathcal{H}_i$ nur ein Repräsentant $h_i \in \mathcal{H}_i$ gewählt wird. Dieser Repräsentant kann irreführen und ein Modell besser oder schlechter bewerten, als es tatsächlich ist. Es wäre sinnvoller, mehrere *Predictor* zu wählen, um einschätzen zu können, wie geeignet eine *Hypothesis Class* ist.

3.2 Cross-validation

Sei zunächst eine feste *Hypothesis Class* $\mathcal{H}$ gegeben. Das Ziel ist es, ein Maß dafür anzugeben, wie gut geeignet $\mathcal{H}$ ist, um anschließend mehrere *Hypothesis Classes* vergleichen zu können. Sei hierfür erneut
$$\mathcal{S} = \mathcal{S}_{\text{train}} \dot\cup \mathcal{S}_{\text{test}}$$
eine Aufteilung von $\mathcal{S}$. Weiter unterteilen wir
$$\mathcal{S}_{\text{train}} = \mathcal{S}_1 \dot\cup \ldots \dot\cup \mathcal{S}_k,$$
wobei $(|\mathcal{S}_1|, \ldots, |\mathcal{S}_k|) = (m_1, \ldots, m_k) \in \mathbb{N}^k$ und alle m_k in etwa gleich groß sind. Nun trainieren wir $\mathcal{H}$ auf der Menge
$$\bigcup_{\substack{j=1\\ j\neq i}}^{k} S_j = S_{\text{train}} \setminus S_i.$$
Da wir für unterschiedliche i unterschiedliche Mengen erhalten, können wir dieses Training k mal ausführen und erhalten *Predictor* $h_1, \ldots, h_k$. Wir setzen wie bei **3.1 Validation Set** fort, indem wir insgeheim für jedes h_i das *Validation Set* $\mathcal{S}_{\text{val}} = \mathcal{S}_i$ festlegen. Konkret heißt das, wir evaluieren analog zur Gleichung **(1)**
$$L_i(h_i) := \frac{1}{m_i} \sum_{(x,y)\in\mathcal{S}_i} \ell(h_i(x), y).$$
Anschließend bilden wir den Durchschnitt über alle **Predictor** vermittels
$$\mathrm{CV}(\mathcal{H}) := \frac{1}{k} \sum_{i=1}^{k} L_i(h_i).$$

Wir erhalten also für unsere *Hypothesis Class* $\mathcal{H}$ einen Wert $\mathrm{CV}(\mathcal{H})$.
Im Falle von mehreren *Hypothesis Classes* $\mathcal{H}_1, \ldots, \mathcal{H}_n$ wählen wir die Klasse $\mathcal{H}_i$ mit den minimalen Wert von $\mathrm{CV}(\mathcal{H}_i)$. Um heuristisch den besten *Predictor* zu finden, trainieren wir die *Hypothesis Class* $\mathcal{H}_i$ anschließend auf der, nun größeren, Menge $\mathcal{S}_{\text{train}}$. Schließlich lässt sich analog zu Schritt 5 in Abschnitt **3.1 Validation Set** ein Wert bestimmen, der $R_D(h)$ approximiert.

Bemerkung 1.3.1.3. Die Wahl von k ist nicht trivial und wir werden sehen, dass es keinen optimalen Wert hierfür gibt. Denkbar wären ad absurdum Wahlen wie

- $k = m'$. In diesem Fall lassen wir nur einen Datenpunkt der Trainingsdaten aus. $L_i(h_i)$ berechnet sich also ausschließlich mit Hilfe dieses Datenpunktes und ist nicht besonders aussagekräftig. Zusätzlich ist der Rechenaufwand enorm, weil wir m' verschiedene *Predictor* generieren müssen, die sich voraussichtlich kaum unterscheiden. Diese Wahl hat sich in Experimenten als eher schlecht geeignet herausgestellt.
- $k = 2$. Wir haben eine geringere Laufzeit, da nur zwei *Predictor* berechnet werden müssen. Diese werden jedoch auf einer kleinen Datenmenge etwa der Größe $\frac{m'}{2}$ trainiert und sind somit tendenziell schlecht geeignet.

In der Praxis hängt die Wahl von k von der Größe des *Datasets* $\mathcal{S}$ ab. Bei großen Datenmengen kann man sich ein kleineres k leisten, um die Laufzeit zu verkürzen. Bei kleinen Datensätzen wählt man k hingegen oft größer, da die Rechenzeit weniger ins Gewicht fällt, während eine möglichst große Trainingsmenge entscheidend ist. Es hat sich herausgestellt, dass in der Praxis k zwischen 5 und 10 gewählt wird, um eine geeignete Balance der Vor- und Nachteile zu finden.

3.3 No-free-lunch Theorem

Das *„No-free-lunch“* Theorem besagt, dass sich in vielen Fällen Algorithmen nicht allgemein optimieren lassen, wenn alle möglichen Probleme betrachten werden. Eine häufige Formulierung dieses Theorems lautet *„All optimization algorithms perform equally well when their performance is averaged across all possible problems.“*

Bemerkung 1.3.1.3. Das *„No-free-lunch“* Theorem ist kein Theorem im mathematischen Sinne. Es gibt weder einen Beweis, noch eine präzise Formulierung, was es aussagen soll. Viel mehr ist es ein Merksatz oder ein Sprichtwort, dass einen erinnern soll, Algorithmen nicht universell optimieren lassen, wenn alle möglichen Probleme beachtet werden.
Im Zusammenhang mit Machine Learning lässt sich festhalten, dass es ohne vorheriges Wissen über unser Problem wie die Wahrscheinlichkeitsverteilung D und der unbekannten Abbildung f keine Möglichkeit gibt, vorherzusagen, ob ein Paar einer *Hypothesis Class* $\mathcal{H}$ und einem *Training Algorithm* $(\mathcal{H}, \mathcal{A})$ gut geeignet ist.