ML Konzepte: Teil 1 (Unbearbeitete Version)

Hayrettin Acar

December 14, 2023

1 Einführung

Maschinelles Lernen ist ein Teilgebiet der künstlichen Intelligenz (KI), das sich damit beschäftigt wie Computer aus Erfahrung lernen können, ohne explizit programmiert zu werden. Computern wird es damit ermöglicht aus Daten zu lernen und Muster zu erkennen, mit denen Vorhersagen getroffen werden.
Hierbei lassen sich die nachfolgenden Teilgebiete definieren:

- Überwachtes Lernen (Supervised Learning)
  - Das Hauptziel besteht darin, eine Abbildung zwischen vorhandenen Trainingsdaten zu finden und auf neue Daten anzuwenden.
  - Regressions-Problem: Hierbei handelt es sich um die Abbildung auf reelle Zahlen.
  - Klassifikations-Problem: Dabei erfolgt die Abbildung auf vordefinierte Werte (z.B. 0,1).
- Unüberwachtes Lernen (Unsupervised Learning)
  - Dieser Ansatz befasst sich mit Aufgaben wie der Mustererkennung, bei denen die Daten keine vorherige Beschriftung oder Kategorisierung aufweisen.
- Vertiefendes Lernen (Deep Learning)
  - Dieser Bereich konzentriert sich auf neuronale Netzwerke mit vielen Schichten (tiefe Netzwerke), um komplexe Probleme zu bewältigen. Hierbei werden verschiedene Algorithmen und Architekturen verwendet.
- Verstärkendes Lernen (Reinforcement Learning)
  - Das Hauptziel besteht darin, *optimales* Verhalten zu erlernen, indem der Algorithmus durch Feedbackstrukturen seine Entscheidungen und Aktionen verbessert.

2 Grundlagen der statistischen Lerntheorie

Die statistische Lerntheorie ist ein Rahmen, der uns hilft, die Leistung von Algorithmen für maschinelles Lernen zu verstehen und zu analysieren. Sie stellt mathematische Werkzeuge und theoretische Erkenntnisse zur Verfügung, um zu beurteilen, wie gut ein Lernalgorithmus aus einem begrenzten Datensatz verallgemeinern kann. Im Wesentlichen hilft sie dabei, die Beziehung zwischen den Daten und dem Lernalgorithmus zu verstehen. Im Folgenden sollen hierzu einige grundlegende Definitionen und Konzepte dargelegt werden.

Definition (Aktion)**.** *Eine Aktion ist der Oberbegriff für das, was unser System produziert.*

Viele Problemstellungen des maschinellen Lernens können folgendermaßen formalisiert werden:

- Beobachte die Eingabe x.
- Führe Aktion a aus.

- Beobachte das Ergebnis y.
- Evaluiere die Aktion in Relation zum Ergebnis $l(a, y)$.

Die ersten Schritte bei der Formalisierung eines maschinellen Lernproblems bestehen in der Regel darin, den Aktionsraum zu definieren und ein Evaluationskriterium zu finden. In diesem Zusammenhang werden die folgenden Räume definiert:

- Eingaberaum (input space) $\mathcal{X}$.
- Aktionsraum (action space) $\mathcal{A}$.
- Ergebnisraum (outcome space) $\mathcal{Y}$.

Definition (Entscheidungsfunktion)**.** *Eine Entscheidungsfunktion (oder Hypothese) nimmt die Eingabe $x \in \mathcal{X}$ und produziert die Aktion $a \in \mathcal{A}$:*

$$f : \mathcal{X} \to \mathcal{A}$$
$$x \mapsto f(x).$$

Definition (Verlustfunktion)**.** *Eine Verlustfunktion evaluiert eine Aktion im Kontext des Ergebnisses y*

$$l : \mathcal{A} \times \mathcal{Y} \to \mathbb{R}$$
$$(a, y) \mapsto l(a, y).$$

In der statistischen Lerntheorie wird zunächst angenommen, dass alle vorliegenden (Trainings- und Test-) Daten von einer gemeinsamen Wahrscheinlichkeitsverteilung $P_{\mathcal{X}\times\mathcal{Y}}$ stammen, wobei alle (x, y) Paare untereinander **unabhängig** und **gleich verteilt** sein sollen. Der Definitionsbereich aller vorliegenden Daten ist also gegeben mit $\mathcal{D}_m = \{(x_1, y_1), ..., (x_m, y_m)\}$.

Wir definieren nun das **Risiko** bzw. den **erwarteten Fehler** einer Entscheidungsfunktion f als den erwarteten Fehler der Verlustfunktion für ein zufällig gewähltes Paar (x, y), welches aus der Verteilung $P_{\mathcal{X}\times\mathcal{Y}}$ stammt

$$R(f) = \mathbb{E}l(f(x), y). \tag{1}$$

Definition (Bayes Entscheidungsfunktion)**.** *Eine Bayes Entscheidungsfunktion $f^* : \mathcal{X} \to \mathcal{A}$ erreicht das **minimale** Risiko unter allen möglichen Funktionen,*

$$f^* = \arg\min_f R(f). \tag{2}$$

Die Bayes-Funktion wird auch oft **Zielfunktion** oder **Risiko-Minimierer (RM)** genannt, da sie die beste, erreichbare Funktion darstellt.

Da die Verteilung $P_{\mathcal{X}\times\mathcal{Y}}$ üblicherweise nicht bekannt ist, kann der Erwartungswert und damit das Risiko in der Regel nicht bestimmt werden. Es besteht allerdings Hoffnung für eine brauchbare Annäherung, denn beispielsweise das (schwache) **Gesetz der großen Zahlen** verrät, dass das arithmetische Mittel von m unabhängigen Zufallvariablen mit steigender Anzahl gegen den Erwartungswert konvergiert,

$$\lim_{m\to\infty} \frac{1}{m} \sum_{i=1}^{m} z^{(i)} = \mathbb{E}z.$$

Das **empirische Risiko** für f kann jetzt abgeleitet werden als

$$\hat{R}_m(f) = \frac{1}{m} \sum_{i=1}^{m} l(f(x^{(i)}), y^{(i)}). \tag{3}$$

Für einen **empirischen Risiko-Minimierer (ERM)** gilt

$$\hat{f} = \arg\min_f \hat{R}_m(f). \tag{4}$$

Die Hoffnung ist nun, dass sich **Risiko** und **empirisches Risiko** für große m annähern. Um dies zu erleichtern wird f typischerweise nicht völlig generisch ermittelt, sondern mithilfe einiger Nebenbedingungen. Gleichzeitig minimieren solche Bedingungen auch die Gefahr der Überanpassung durch eine hoch spezialisierte Funktion, die nur auf die gegebenen Trainingsdaten passt. Implizit wird bei der Wahl eines Lernalgorithmus die sog. **Hypothesen-Klasse** $\mathcal{H}$ mit $f \in \mathcal{H}$ definiert. Beispielsweise wird die Auswahl bei der Festlegung auf Lineare Regression auf eben diese Arten der Vorhersagefunktionen limitiert. Wir können nun innerhalb der definierten Klasse ebenfalls den empirischen RM definieren:

$$\hat{f}_m = \arg\min_{f \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} l(f(x^{(i)}), y^{(i)}). \tag{5}$$

Analog wird der (Bayes) Risiko-Minimierer für die Klasse definiert zu

$$f_{\mathcal{H}} = \arg\min_{f \in \mathcal{H}} \mathbb{E} l(f(x), y). \tag{6}$$

2.1 Fehlerzerlegung

Wir wollen nun allgemein beschreiben, welche Fehlerarten bei der Erzeugung des Lernalgorithmus typischerweise auftreten. Dabei interessiert uns am meisten, wie groß der Fehler eines Algorithmus bei neuen, unvorhergesehenen Testdaten ist.

Gegeben sei ein Trainingset $\mathcal{D}_m = \{(x_1, y_1), ..., (x_m, y_m)\}$. Angenommen sei ebenfalls, dass diese Trainingsdaten unabhängig und gleich verteilt sind. Mit der *idealen* Hypothese f^* sei dann der wahre Wert von y gegeben durch

$$y^{(i)} = f^*(x^{(i)}) + \xi^{(i)},$$

wobei $\xi^{(i)}$ einen zufälligen Fehleranteil darstellt. Das ultimative Ziel ist die Annäherung an die Zielfunktion f^*. Wir wählen durch die Festlegung der Nebenbedingungen unsere Hypothesenklasse $\mathcal{H}$ aus, innerhalb derer unser Lernalgorithmus liegen kann. In dieser Klasse existiert ist nun eine bestmögliche Hypothese, welche durch den Risiko-Minimierer $f_{\mathcal{H}}$ gegeben ist, (1).

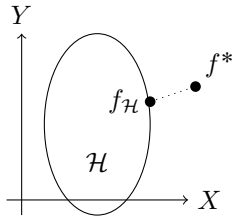

Figure 1: Hypothesenklasse.

Die Differenz zwischen dem Risiko der Zielfunktion und des Risiko-Minimierers innerhalb der Klasse wird **Approximations-Fehler** genannt,

$$\underbrace{R(f_{\mathcal{H}}) - R(f^*)}_{Approx. Fehler}.$$

Der Approximations-Fehler kann **nicht negativ** werden, da der Fehler von $R(f_{\mathcal{H}})$ definitionsgemäß immer größer ist als $R(f^*)$. Weiterhin ist innerhalb der Hypothesenklasse eine erste Annäherung an $(f_{\mathcal{H}})$ gegeben durch den empirischen Risiko-Minimierer $\hat{f}_m$, dessen Position nun von den vorhandenen Trainingsdaten abhängt, (2).

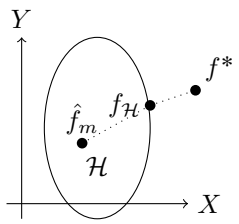

Figure 2: Hypothesenklasse.

Die Differenz zwischen dem Fehler des RMs der Klasse und des empirischen RMs wird **Abschätzungs-Fehler** genannt,

$$\underbrace{R(\hat{f}_m) - R(f_{\mathcal{H}})}_{Abschaetz.Fehler}.$$

In der Praxis wird allerdings auch der ERM $\hat{f}_m$ nicht immer gefunden, sondern lediglich eine Annäherung $\tilde{f}_m$, (3).

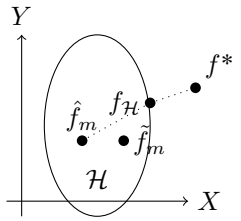

Figure 3: Hypothesenklasse.

Wir bezeichnen die Differenz zwischen dem Risiko des ERMs und der Annäherung als **Optimierungsfehler**,

$$\underbrace{R(\tilde{f}_m) - R(\hat{f}_m)}_{Optim.Fehler}.$$

Zu beachten ist hierbei, dass der Optimierungs-Fehler auch negativ werden kann, d.h. die Annäherung ist hinsichtlich der optimalen Hypothese f^* näher, als der eigentliche ERM dieser Hyopthesenklasse.

Der **Generalisierungsfehler** bezeichnet nun die Abweichung des Risikos zwischen gefundener Hypothese $\tilde{f}_m$ und der bestmöglichen Hypothese f^*,

$$\underbrace{R(\tilde{f}_m) - R(f^*)}_{Generalisierungsfehler} = \underbrace{R(\tilde{f}_m) - R(\hat{f}_m)}_{Optim.Fehler} + \underbrace{R(\hat{f}_m) - R(f_{\mathcal{H}})}_{Abschaetz.Fehler} + \underbrace{R(f_{\mathcal{H}}) - R(f^*)}_{Approx.Fehler}.$$

2.1.1 Fehlerabschätzung für endliche $\mathcal{H}$

Es sei die Annahme getroffen $\mathcal{H} = \{f_1, ..., f_k\}$ sei endlich und bestehe aus k Hypothesen. Mithilfe der ERM könnte nun diejenige Hypothese $\hat{f}_m$ bestimmt werden, die den kleinsten Trainingsfehler aufweist. Das Ziel ist nun eine Abschätzung für den Generalisierungsfehler $R(\hat{f}_m)$ geben zu können.

Als Beispiel sei eine Hypothese $f_i \in \mathcal{H}$ für eine Bernoulli Verteilung mit der Zufallsvariablen $Z = 1\{f_i(x) \neq y\}$ und $(x, y) \sim \mathcal{D}$ angenommen. Dabei bezeichnet $1\{x\}$ die Indikator-Funktion

$$1\{x\} = \begin{cases} 0, & \text{x=falsch} \\ 1, & \text{x=wahr} \end{cases}$$

Z_j ist hierbei ein Indikator dafür, ob f_i das j-te Trainingsbeispiel richtig klassifiziert hat, $Z_j = 1\{f_i(x^{(j)}) \neq y^{(j)}\}$. Der empirische Fehler einer Hypothese f_i ist dann gegeben mit

$$\hat{R}_m(f_i) = \frac{1}{m} \sum_{j=1}^{m} Z_j.$$

Mit der **Chernoff-Ungleichung** kann nun eine obere Schranke für die Wahrscheinlichkeit dafür gefunden werden, dass die Differenz zwischen empirischem und generalisiertem Fehler größer als eine Konstante γ ist,

$$P(|R(f_i) - \hat{R}_m(f_i)| > \gamma) \leqslant 2\exp(-2\gamma^2 m). \tag{7}$$

Damit ist ersichtlich, dass für große m die Wahrscheinlichkeit für eine große Differenz immer kleiner wird, d.h. $R(f_i)$ und $\hat{R}_m(f_i)$ nähern sich einander an. Wir wollen nun eine Abschätzung für alle $f_i \in \mathcal{H}$ gleichzeitig finden. Mithilfe der **Boolschen Ungleichung** kann nun formuliert werden

$$P(\exists f \in \mathcal{H}.|R(f_i) - \hat{R}_m(f_i)| > \gamma) \leqslant \sum_{i=1}^{k} 2\exp(-2\gamma^2 m) = 2k\exp(-2\gamma^2 m). \tag{8}$$

Die komplementäre Wahrscheinlichkeit ist dann gegeben als

$$\begin{aligned} & P(\neg\exists f \in \mathcal{H}.|R(f_i) - \hat{R}_m(f_i)| > \gamma) \leqslant 1 - 2k\exp(-2\gamma^2 m) \\ \Leftrightarrow\ & P(\forall f \in \mathcal{H}.|R(f_i) - \hat{R}_m(f_i)| \leqslant \gamma) \geqslant 1 - \underbrace{2k\exp(-2\gamma^2 m)}_{\delta}. \end{aligned} \tag{9}$$

Mit dieser Gleichung ist demnach die Wahrscheinlichkeit, dass die Differenz für alle Hypothesen kleiner ist als γ gegeben mit mindestens $1 - \delta$. Man kann sich nun beispielsweise fragen, wie groß die Anzahl der Trainingsdaten m sein muss, damit eine Wahrscheinlichkeit $1 - \gamma$ garantiert werden kann, mit der die Differenz für alle $f \in \mathcal{H}$ kleiner als γ ist

$$m \geqslant \frac{1}{2\gamma^2}\log(\frac{2k}{\delta}). \tag{10}$$

Man beachte dabei, dass die Anzahl der Hypothesen k nur logarithmisch wächst. Die Anzahl der Trainingsdaten m, die ein Algorithmus benötigt um ein bestimmtes Performancelevel zu erreichen, wird im Allgemeinen als die **sample complexity** bezeichnet.

Wir können weiterhin den Zusammenhang zur bestmöglichen Hypothese $f^* = \arg\min_{f\in\mathcal{H}} R(f)$ herstellen. Mit der Bedingung $|R(f) - \hat{R}_m(f)| \leqslant \gamma$ gilt

$$R(\hat{f}_m) \leqslant \hat{R}_m(\hat{f}_m) + \gamma \leqslant \hat{R}_m(f^*) + \gamma \leqslant R(f^*) + 2\gamma. \tag{11}$$

Trifft also die o.g. Bedingung zu, so ist der Generalisierungsfehler von $\hat{f}$ maximal 2γ schlechter als der Generalisierungsfehler der bestmöglichen, empirischen Hypothese,

$$R(\hat{f}_m) \leqslant \left(\min_{f\in\mathcal{H}} R(f)\right) + 2\gamma. \tag{12}$$

2.2 Bias und Varianz

Die statistischen Konzepte von Bias und Varianz sind eng verwandt mit der Fehlerbetrachtung einer Hypothese. Um die grundlegenden Konzepte darzustellen, sei zunächst angenommen es liege eine Wahrscheinlichkeitsverteilung P vor. Weiterhin liege eine zufällig gewählte Datenmenge $\mathcal{D}_m = (x_1, x_2, ..., x_m)$ vor, deren Zufallsvariablen unabhängig und gleichverteilt sind.

Definition (Parameter)**.** *Ein Parameter einer Wahrscheinlichkeitsverteilung P ist jede Funktion*

$$\mu = \mu(P).$$

Ein Parameter (z.B. der Erwartungswert) ist hierbei **nicht zufällig**.

Definition (Statistik)**.** *Eine Statistik ist jede Funktion der Datenmenge*

$$s = s(\mathcal{D}_m).$$

Eine Statistik ist meist die Annäherung an den wahren Wert einer Verteilung (z.B. der Erwartungswert $\hat{\mu} \approx \mu$). Da die Datenmenge zufällig ist, sind auch Statistiken **zufällig**, z.B. der Durchschnitt einer Datenmenge

$$\overline{x} = \frac{1}{m}\sum_{i=1}^{m} x_i.$$

Da Statistiken zufällig sind, haben sie damit selbst auch eine Wahrscheinlichkeitsverteilung (sog. **Stichprobenverteilung**). Interessant sind dabei die (**nicht zufälligen**) Parameter einer Stichprobenverteilung, wie z.B. die Standardabweichung.

Definition (Bias)**.** *Wir definieren den Bias des Erwartungswertes als die Abweichung*

$$Bias(\hat{\mu}) = \mathbb{E}\hat{\mu} - \mu.$$

Ein hoher Bias indiziert meist eine sog. **Unteranpassung**. Das aufgestellte Modell weist dann schon mit den Trainingsdaten einen hohen Vorhersagefehler auf.

Definition (Varianz)**.** *Wir definieren die Varianz des Erwartungswertes als die Abweichung*

$$Var(\hat{\mu}) = \mathbb{E}\hat{\mu}^2 - (\mathbb{E}\hat{\mu})^2.$$

Eine hohe Varianz indiziert meist eine sog. **Überanpassung**. Das Modell weist dann zwar einen geringen Fehler für eine definierte Datenmenge auf, allerdings nicht für unvorhergesehene Daten.

Weder Bias, noch Varianz sind abhängig von einer gewählten Datenmenge und damit **nicht zufällig**. Wie können diese Größen abgeschätzt werden? Sei B die Anzahl von unabhängig voneinander gezogenen Datensätzen, $\mathcal{D}_m^{(1)}, \mathcal{D}_m^{(2)}, ..., \mathcal{D}_m^{(B)}$. Es gilt dann

$$\mathbb{E}\hat{\mu} \approx \frac{1}{B}\sum_{i=1}^{B} \hat{\mu}(\mathcal{D}_m^{(i)})$$

$$\mathbb{E}\hat{\mu}^2 \approx \frac{1}{B}\sum_{i=1}^{B} [\hat{\mu}(\mathcal{D}_m^{(i)})]^2$$

Damit kann die Varianz angenähert werden zu

$$Var(\hat{\mu}) \approx \frac{1}{B}\sum_{i=1}^{B} [\hat{\mu}(\mathcal{D}_m^{(i)})]^2 - [\frac{1}{B}\sum_{i=1}^{B} \hat{\mu}(\mathcal{D}_m^{(i)}]^2 \tag{13}$$

Mit einer großen Anzahl von Datensätzen kann die Varianz also verringert werden. Liegt diese nicht vor, so kann die sog. **bootstrap**-Methode Abhilfe verschaffen. Hierbei wird der ursprüngliche Datensatz in kleinere Untermengen aufgeteilt und so behandelt, als wären diese unabhängig und gleichverteilt.

2.3 Normalisierung und Regularisierung

Die Regularisierung ist ein Ansatz, um eine Überanpassung des Algorithmus an die Trainingsdaten zu verhindern. Durch die Regularisierung wird ein Strafterm eingeführt, der das Modell davon abhält, den Merkmalen übermäßig große Koeffizienten zuzuweisen. Die Hoffnung ist hierbei, dass im Austausch durch eine geringere Anpassung an die Trainingsdaten eine bessere Generalisierung auf ungesehene Testdaten erzielt werden kann. Da die relative Skalierung der Koeffizienten untereinander, ausschlaggebend für den Effekt der Regularisierung ist, sollte der Eingaberaum möglichst vor der Durchführung einer Optimierung mit Regulariserungsterm **normalisiert** werden. Üblicherweise erfolgt damit eine Skalierung der Werte auf einen Bereich von $x' \in [0, 1]$, beispielsweise durch

$$x' = \frac{x - \min(\mathcal{D})}{\max(\mathcal{D}) - \min(\mathcal{D})}$$

wobei ausreißende Daten vorher möglichst beseitigt werden sollten. Sind die Daten darüber hinaus normalverteilt, so wird häufig mit dem Erwartungswert $\mu(\mathcal{D})$ und der Standardabweichung $\sigma(\mathcal{D})$ gearbeitet,

$$x' = \frac{x - \mu}{\sigma}.$$

Das Konzept der **Regularisierung** ist eng mit einem definierten **Komplexitätsgrad** der Hypothese verknüpft. Wir definieren eine Komplexitätsbemessung als $\Omega : \mathcal{H} \to [0, \infty]$. Damit kann eine Hypothesenklasse auf den Komplexitätsradius r beschränkt werden.

$$\mathcal{H}_r = \{f \in \mathcal{H} | \Omega(f) \leqslant r\}. \tag{14}$$

Die regularisierte Form eines empirischen Risiko-Minimierers kann dann in der sog. **Tikhonov** Form geschrieben werden als

$$\min_{f \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} l(f(x^{(i)}), y^{(i)}) + \lambda \Omega(f). \tag{15}$$

Dabei ist λ ein Hyperparameter, welcher den Grad der Regularisierung beeinflusst.

Das Konzept der Regularisierung soll hier beispielhaft anhand der **Methode der kleinsten Quadrate** ($l(f(x), y) = (f(x) - y)^2$) vertieft werden. Betrachtet sei hierfür das d-dimensionale, lineare Modell

$$\mathcal{H} = \{f : \mathbb{R}^d \to \mathbb{R} | f(x) = \theta^T x, \text{ für } \theta \in \mathbb{R}^d\}.$$

Der ERM kann gefunden werden zu

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2.$$

Üblicherweise werden folgende Komplexitätsdefinitionen verwendet:

- l_0 **Komplexität**: Die Anzahl der nicht-null Koeffizienten
- l_1 **Lasso**: Die Betragssumme der Koeffizienten $\sum_{i=1}^{d} |\theta_i|$
- l_2 **Ridge**: Die Quadratsumme der Koeffizienten $\sum_{i=1}^{d} \theta_i^2$

Die Ridge Regression wäre dann gegeben mit

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2 + \lambda ||\theta||_2^2. \tag{16}$$

Wie kommt nun der Regularisierungseffekt zustande? Betrachtet man hierzu beispielsweise die **Cauchy-Schwarz Ungleichung**, so ergibt sich eine Limitierung der Änderungsrate der Hypthese für die Änderung $|\hat{f}(x + \Delta)|$,

$$|\hat{f}(x + \Delta) - \hat{f}(x)| = |\hat{\theta}^T \Delta| \leqslant ||\hat{\theta}||_2 ||\Delta||_2. \tag{17}$$

Die Regularisierung kann auch im Falle von **Multikollinearität**, d.h. hohe Interkorrelationen zwischen zwei oder mehr unabhängigen Variablen in einem multiplen Regressionsmodell, Abhilfe zur verbesserten Koeffizientenwahl schaffen. Das Problem ist hierbei, dass die Koeffizientenabschätzung bei starker Korrelation ungenauer und weniger verlässlich wird. Spezielle Regularisierungsansätze (z.B. **Elastic Net**) helfen dann dabei die korrelierenden Koeffizienten *gleichmäßiger* zu vergeben um u.a. einen gegenseitigen Auslöschungseffekt zu erzielen.

2.4 Feature Extraktion

In einem maschinellen Lernproblem liegen die gewählten Merkmale herkömmlicher Weise nicht in der Form $\mathbb{R}^d$ (**Feature-Vektor**) vor, z.B. Textdokumente, Sätze, Bilder etc. Wir bezeichnen die Abbildung

$$\phi(x) \to \mathbb{R}^d$$

als **Feature Extraktion**. Der Feature-Vektor kann auf unterschiedliche Weise repräsentiert werden. Eine übliche Art ist die sog. **one-hot encoded** Vektor Form, in der exakt ein Eintrag des Vektors ungleich Null ist,

$$\vec{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}.$$

Bei sehr dicht befüllten Vektoren eignet sich auch die Verwendung einer **Feature-Matrix**, mit dem Vorteil dass etwaige Matrizenoperationen vergleichsweise effizient durchgeführt werden können (z.B. auf GPU Basis). Bei sehr dünn besetzten Vektoren wird häufig auch eine **Key-Value Pair** basierte Form verwendet, in der nur die entsprechend relevanten Merkmale berücksichtigt werden.

Zumeist ist allerdings die Beurteilung der Relevanz eines Features für eine gute Vorhersage nicht eindeutig. Eine Abhilfe können hier sog. **Feature-Templates** schaffen. Sie stellen eine Gruppe von Merkmalen dar, welche alle in ähnlicher Weise generiert werden. Soll beispielsweise bei einem Wort erkannt werden, ob es sich um eine gültige E-Mail handelt, so könnte anstatt des Merkmals *endet-mit-com* einfach eine Gruppe von Merkmalen definert werden, die alle möglichen Kombinationen aus den letzten drei Buchstaben berücksichtigt. Der Vorteil hierbei ist dabei, dass der Lernalgorithmus durch entsprechende Gewichtszuteilungen selbst entscheidet, wie relevant jedes Merkmal ist.

3 Lineare Regression

Bei der linearen Regression wird die Hypothese durch eine lineare Funktion

$$h : X \to Y$$
$$h_\theta(x_1, x_2, ...) = \theta_0 + \theta_1 x_1 + ... = \sum_{j=0}^{d} \theta_j x_j. \tag{18}$$

dargestellt. Die Raumdefinitionen sind damit

- Eingaberaum (input space) $\mathcal{X} = \mathbb{R}^d$.
- Aktionsraum (action space) $\mathcal{A} = \mathbb{R}$.
- Ergebnisraum (outcome space) $\mathcal{Y} = \mathbb{R}$.

Die Parameter der Hypothese θ_j werden auch als **Gewichte** bezeichnet. Die Zielwerte sind hierbei im Raum der reellen Zahlen ($y^{(i)} \in \mathbb{R}$) zu finden. Mit der Konvention $x_0 = 1$ (sog. **Intercept-Term**) lässt sich auch die vektorielle Darstellung

$$h_\theta(x_1, x_2, ...) = \theta^T \vec{x}$$

verwenden. Dies ist üblich um den additiven Term θ_0 mitzuführen, welcher dem Modell zusätzliche Flexibilität verschafft. Damit ist der Hypothesenraum definiert mit

$$\mathcal{H} = \{f : \mathbb{R} \to \mathbb{R} | f(x) = \theta^T \vec{x}, \theta \in \mathbb{R}^d\}. \tag{19}$$

Für die Lineare Regression werden die folgenden **Konventionen** definiert:

- **Merkmale (Features)**: Die zum Raum $X = \mathbb{R}^d$ gehörenden Merkmale werden in der Eingabematrix definiert
$$X = \begin{bmatrix} x_0^1 & x_1^1 & x_2^1 \\ x_0^2 & x_1^2 & x_2^2 \\ x_0^3 & x_1^3 & x_2^3 \\ ... & ... & .... \end{bmatrix}.$$
Es wird zu jedem Datensatz
$$i = 1, ..., m$$
ein entsprechendes Feature
$$x_j^{(i)},\ mit\ j = 0, .., d$$
definiert. Hierbei is d die Anzahl der zu berücksichtigenden Features ($x_0^{(i)} = 1$ wird nicht als eigenständiger Parameter betrachtet).

- **Zielwerte**: Die Ergebnisdaten des Raums $Y = \mathbb{R}$ werden mit

$$\vec{y} = \begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix}$$

dargestellt.

- **Trainingsdaten**: Ein Trainings-Satz wird beschrieben durch

$$(x^{(i)}, y^{(i)}) \in (X \times Y).$$

3.1 Verlustfunktion

Mithilfe einer Verlustfunktion der Form $(h_\theta(x^{(i)}) - y^{(i)})$ kann ein Maß für die Abweichung zwischen der Hypothese und den jeweiligen Zielwerten eines Trainingssatzes gefunden werden.

Eine passende Verlustfunktion für die Lineare Regression, kann durch eine wahrscheinlichkeitsbasierte Betrachtung gefunden werden. Es sei die Annahme getroffen, dass die Abweichung zwischen der Vorhersage und dem Zielwert mithilfe eines normalverteilten Fehlerterms $y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$ dargestellt werden kann. Weiterhin sei angenommen, dass die Fehlerterme $\epsilon^{(i)}$ unabhängig und normalverteilt sind, $\epsilon^{(i)} \sim \mathcal{N}(\mu = 0, \sigma^2)$. Die Wahrscheinlichkeitsdichte lässt sich dann schreiben als

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}). \tag{20}$$

Mit den Zufallsvariablen $x^{(i)}$ und den Parametern θ gilt für die Wahrscheinlichkeitsdichte

$$p(y^{(i)}|x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}). \tag{21}$$

Bisher wurde die Wahrscheinlichkeit als Funktion der Eingabedaten X betrachtet. Betrachtet man die Verteilung allerdings als Funktion der Parameter $\theta \in \Theta$ und einem festen Datensatz (sog. **likelihood-Funktion**), so ergibt sich

$$L(\theta) = L(\theta; X; \vec{y}) = P(\vec{y}|X; \theta). \tag{22}$$

Mit der Annahme, dass die Fehlerterme unabhängig sind, ergibt sich

$$L(\theta) = \prod_{i=1}^{m} P(y^{(i)}|x^{(i)}; \theta) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}). \tag{23}$$

Um die Funktion $L(\theta)$ hinsichtlich ihrer Wahrscheinlichkeit zu maximieren, sog. **maximum likelihood**),

$$\hat{\theta}_{MLE} = \arg\max_{\theta \in \Theta} L(\theta),$$

genügt es jede monoton steigende Funktion von $L(\theta)$ zu maximieren. Man findet schnell den Zusammenhang

$$l(\theta) = \log(L(\theta)) = m \cdot \log(\frac{1}{\sqrt{2\pi}\sigma}) - \frac{1}{\sigma^2}\frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2. \tag{24}$$

Die Maximierung der Funktion entspricht also der Minimierung des quadratischen Fehlerterms

$$\frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T x^{(i)})^2.$$

Ein natürlicher Ansatz zur Optimierung der Parameter θ, ist demnach die Verwendung der kleinsten Fehlerquadrate als Verlustfunktion. Diese Art der Verlustfunktion wird auch **distanz-basierte**

Verlustfunktion genannt, da sie funktional nur vom Restwert zwischen Vorhersage und dem Zielwert abhängt ,

$$l(f(x), y) = \psi((f(x) - y), \psi : \mathbb{R} \to \mathbb{R}.$$

Es gilt dabei immer, dass der Verlust null ist, falls der Restwert ebenfalls null ist, $\psi(0) = 0$. Das empirische Risiko, welches wir nun minimieren wollen, kann damit definiert werden zu

$$\hat{R}(\theta) = \frac{1}{m} \sum_{i=1}^{m} l(f(x^{(i)}, y^{(i)}) = \frac{1}{m} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2. \tag{25}$$

Gleichzeitig definieren wir die **Kostenfunktion** zu

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} l(f(x^{(i)}, y^{(i)}) = \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2. \tag{26}$$

3.2 Optimierung durch Gradientenverfahren (Gradient Descent)

Das Gradientenverfahren hat sich als effiziente Methode zur Optimierung für konvexe Kostenfunktionen herausgestellt. Es ist ein iterativer Optimierungsalgorithmus, der verwendet wird, um das Minimum einer differenzierbaren Funktion zu finden. Die Grundidee des Gradientenabstiegs besteht darin, Schritte in Richtung der steilsten Abnahme der Kostenfunktion zu unternehmen. Die steilste Abnahme wird durch den Gradienten bestimmt, der ein Vektor der partiellen Ableitungen der Kostenfunktion in Bezug auf jeden Modellparameter ist. In jedem Iterationsschritt wird dabei der Gradient

$$\nabla J(\theta) = \frac{1}{2} \sum_{i=1}^{m} \nabla_\theta l(f(x^{(i)}, y^{(i)})$$

berechnet. Mit anderen Worten: Der Gradient zeigt in die Richtung des schnellsten Abstiegs der Kostenfunktion.

Mit einer initialen Wahl von θ werden alle Parameter simultan aktualisiert durch

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta). \tag{27}$$

Die Lernrate α ist ein Hyperparameter, der die Schrittgröße zur Minimierung der Verlustfunktion in jeder Iteration bestimmt. Sie beeinflusst die Geschwindigkeit und Konvergenz des Optimierungsprozesses. Eine höhere Lernrate kann zu einer schnelleren Konvergenz führen, kann aber auch zu einem Überschreiten der optimalen Lösung führen. Eine niedrigere Lernrate kann zu einer langsameren Konvergenz, aber zu einer höheren Präzision führen. Für ein effektives Training ist es wichtig, eine geeignete Lernrate zu wählen. Da in diesem Verfahren unmittelbar alle Trainingsdaten für einen einzelnen Aktualisierungsschritt herangezogen werden, wird dieses Verfahren auch als **stapelweises** Gradientenverfahren (Batch Gradient Descent) bezeichnet. Man findet schnell, dass die Regel für einen einzelnen Trainingssatz gegeben ist durch

$$\theta_j := \theta_j - \alpha(h_\theta(x) - y)x_j.$$

Für den gesamten Trainingsdatensatz ist die dazugehörige Regel gegeben durch

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}. \tag{28}$$

Dabei werden alle Parameter $j = 0, ..., d$ gleichzeitig aktualisiert. Bei großen Datensätzen kann das **stochastische** Gradientverfahren (SGD) schneller zur Konvergenz führen. Hierbei ist ein Iterationsschritt nicht durch den gesamten Trainingsdatensatz bestimmt, sondern jeder einzelne Datensatz führt zu einer Anpassung in der Parameteroptimierung. Die Optimierung ist dann gegeben durch

$$\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}, \forall j. \tag{29}$$

3.3 Optimierung durch Normalen-Gleichung

Wird die Kostenfunktion gemäß der kleinsten Fehlerquadrate gewählt, so lässt sich die Optimierung in geschlossener Form und ohne Iteration finden. Die Trainingsdaten können in Matrixschreibweise geschrieben werden als

$$X = \begin{bmatrix} - & (x^1)^T & - \\ - & (x^2)^T & - \\ - & (x^3)^T & - \\ & \vdots & \\ - & (x^m)^T & - \end{bmatrix}.$$

Mit den Zielwerten y kann die Hypothese $h_\theta(x^{(i)}) = (x^{(i)})^T\theta$ geschrieben werden als

$$X\theta - y = \begin{bmatrix} (x^1)^T\theta \\ \vdots \\ (x^m)^T\theta \end{bmatrix} - \begin{bmatrix} y^1 \\ \vdots \\ y^m \end{bmatrix}. \tag{30}$$

Damit kann die Kostenfunktion geschrieben werden als

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 = \frac{1}{2}(X\theta - y)^T(X\theta - y). \tag{31}$$

Um J bezogen auf die Parameter θ zu minimieren, muss die Ableitung des Ausdrucks bestimmt werden. Man findet schnell, dass gilt

$$\nabla_\theta J(\theta) = \nabla_\theta(\frac{1}{2}(X\theta - y)^T(X\theta - y)) = X^TX\theta - X^Ty. \tag{32}$$

Damit kann die Minimierung bestimmt werden zu

$$\theta = (X^TX)^{-1}X^Ty. \tag{33}$$

3.4 Beispielbetrachtung

Betrachtet sei der Profit von Eiscremeverkäufen in Abhängigkeit der Außentemperatur. In diesem Beispiel wird ein einzelnes Eingabemerkmal (Temperatur) betrachtet. Mit Berücksichtigung von $x_0^i = 1$ wird also eine Eingabematrix mit den Dimensionen $(m \times 2)$ verwendet. Insgesamt wurden $m = 76$ Trainingsdaten verwendet. Für das Gradientenverfahren wurde initial $\theta = (0, 0)$ gewählt und eine Iterationszahl von 15 definiert. Mithilfe der optimierten Parameter von θ lassen sich die Regressionsgeraden darstellen wie in (4) gezeigt. Im Vergleich sind hier die Regressionsgeraden der Gradientenverfahren (BGD, SGD) und der Normalengleichung (NEQN) zu sehen.

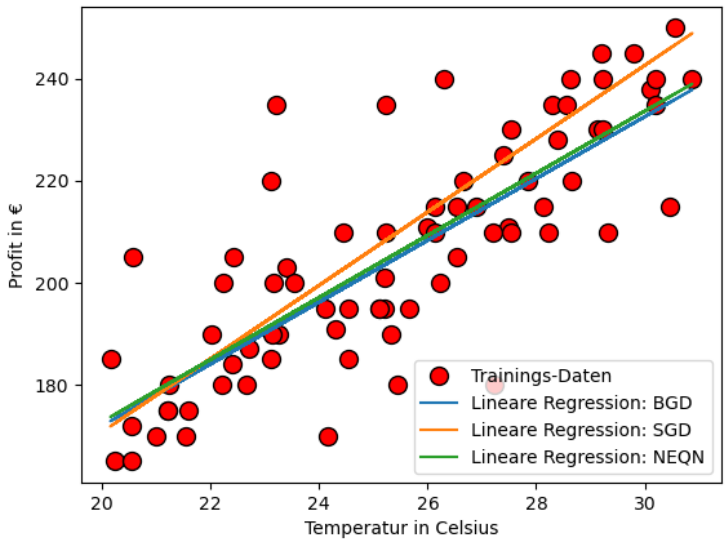

Figure 4: Der Profit in Abhängigkeit der Außentemperatur mit Regressionsgeraden im Vergleich.