Vorlesungsnotizen

Optimierung

Wintersemester 24/25

Emma Bach

Vorlesung gehalten von
Prof. Rolf Backofen
und
Prof. Moritz Diehl

March 16, 2025

Inhalt

Chapter 1

Einführung

Ein Optimierungsproblem besteht aus einer **zulässigen Menge** G und einer **Zielfunktion** $f : G \to H$, wobei wir im Allgemeinen von $H = \mathbb{R}$ ausgehen werden. Wir schreiben dann zum Beispiel

$$\min_{x \in \mathbb{R}} f(x),$$

um das Problem "finde den kleinsten Wert, den $f(x)$ bei reellem x annimt" zu notieren, und

$$\operatorname*{argmin}_{x \in \mathbb{R}} f(x) = 4$$

für das Problem "finde die kleinste reelle Zahl x, sodass $f(x) = 4$ ist.

Historisch ist das Feld der mathematischen Optimierung unter Nebenbedingungen stark verankert im Feld der **Operations Research**, welches sich mit der Optimierung von Produktionskosten unter gegebenen Bedingungen beschäftigt. Heutzutage hat die Optimierung zahlreiche Anwendungen, z.B. in der Pfadplanung, in Computer Vision, in der Bioinformatik, im maschinellen Lernen oder im Hardwaredesign.

1.1 Anwendungsbeispiel: Support Vector Machines

Ein klassisches Problem des maschinellen Lernes ist die Klassifizierung von Daten durch eine lineare Entscheidungsgrenze - alle Punkte (x_i, y_i) über einer Ebene werden einer Klasse zugeordnet, und alle Punkte unter der Ebene einer anderen Klasse. Das Problem besteht daraus, eine Optimale Trennungsebene zu finden. Diese wird beschrieben durch eine Gleichung der Form

$$\hat{y}(x) = w^T x + w_0.$$

Sei w^* der Vektor $(w_0, w_1, \ldots, w_{n-1})$. Es stellt sich heraus, dass das zu lösende Optimierungsproblem gegeben ist durch:

$$\begin{aligned} \operatorname*{argmin}_{w^* \in \mathbb{R}^n} \quad & \tfrac{1}{2}||w^*||^2 \\ \text{s.t.} \quad & \forall i \; y_i \hat{y}(x_i) \geq 1 \end{aligned}$$

Wir wollen die Länge $||w^*||$ des Vektors w^* minimieren, da dies zu einem größeren Abstand zwischen unserer Ebene und den Datenpunkten führt, wodurch unser Modell besser generalisiert. Die Nebenbedingung entspricht der Anforderung, dass alle Punkte korrekt klassifiziert werden sollen.

1.2 Typische Problemklassen

Optimierungsprobleme können gemäß diverser Kriterien klassifiziert werden:

- Probleme mit Nebenbedingungen vs Probleme ohne Nebenbedingungen

- Optimierung mit Variablen aus verschiedenen Mengen, insbesondere kontinuierliche Variablen vs diskrete Variablen
- Lineare vs nichtlineare Funktionen
- Eindimensionale vs mehrdimensionale Funktionen
- Konvexe Funktionen vs nicht konvexe Funktionen
- Konvexe Mengen vs nicht konvexe Mengen

Diese verschiedenen Problemklassen führen zu unterschiedlich schwierigen Problemen. Insbesondere sind konvexe Probleme einfacher zu lösen als nicht konvexe Probleme, kontinuierliche Probleme sind in der Regel einfacher zu lösen als diskrete Probleme, und lineare Probleme sind einfacher als nichtlineare Probleme. Relevante Fragen sind dann:

- Wie schnell kovergiert das Verfahren zu einer Lösung? Wie viele Iterationen sind nötig? Was ist die Komplexität (in O-Notation) einer einzelnen Iteration?
- Konvergiert das Verfahren immer gegen ein Globales Optimum? Falls nein, gibt es garantierte obere/untere Schranken für die maximale Abweichung vom globalen Optimum?

1.3 Konvexität

Eine Menge G ist **konvex**, wenn für beliebige Punkte $x, y \in G$ auch beliebige lineare Interpolationen zwischen den Punkten in der Menge enthalten sind:

$$x, y \in G \implies \{(1-\alpha)x + \alpha y \mid \alpha \in [0,1]\} \subset G$$

Intuitiv entspricht das der Forderung, dass zwischen für alle Paare von Punkten eine Verbindungsstrecke zwischen den Punkten in der Menge enthalten sein muss.

Die **konvexe Hülle** einer Menge G ist die kleinste konvexe Menge H sodass $G \subset H$.

Analog zur Definition einer konvexen Menge ist eine **konvexe Funktion** definiert als eine Funktion, für die gilt:

$$x, y \in G \implies f((1-\alpha)x + \alpha y) \leq (1-\alpha)f(x) + \alpha f(y)\ \forall \alpha \in [0,1]$$

Dies entspricht der Forderung, dass jede Verbindungsstrecke zwischen zwei Punkten auf dem Graphen unterhalb des Graphen liegen muss.

Eine nicht konvexe Funktion, in der jedes lokale Minimum auch ein globales Minimum ist, wird als **quasikonvex** bezeichnet. Da dies dem Hauptvorteil von konvexen Funktionen in der Optimierung entspricht, ist die Optimierung von quasikonvexe Funktionen ebenfalls einfacher als die Optimierung allgemeiner nichtlinearer Funktionen.

Chapter 2

Gradientenverfahren

2.1 Wiederholung: Mehrdimensionale Differentiation

Zur Erinnerung: der **Gradient** ∇f ist ein Vektor, der alle partiellen Ableitungen von f enthält, also die Ableitung von f in alle Koordinatenrichtungen:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)^T$$

Im eindimensionalen Fall entspricht das genau der gewöhnlichen Ableitung, da partielle Ableitungen letztendlich genau wie ordinäre Ableitungen berechnet werden. Sei $f_1(x_1, x_2, x_3)$ eine beliebige Funktion $\mathbb{R}^3 \to \mathbb{R}$ und sei $f_2(x_1) = f_2(x_1, x_2, x_3)$ für beliebige Konstante x_2, x_3. Dann ist

$$\frac{\partial}{\partial x_1} f_1(x_1, x_2, x_3) = \frac{d}{dx_1} f_2(x_1)$$

Zum Beispiel:

$$\begin{aligned} f_1(x_1, x_2, x_3) &= x_1^2 x_2 + x_1 x_3 + 2x_2 \\ \frac{\partial}{\partial x_1} f_1(x_1, x_2, x_3) &= 2x_1 x_2 + x_3 \end{aligned}$$

entspricht der Ordinären Ableitung

$$\begin{aligned} f_2(x_1) &= x_1^2 x_2 + x_1 x_3 + 2x_2 \\ \frac{d}{dx_1} f_2(x_1) &= 2x_1 x_2 + x_3 \end{aligned}$$

Eine mehrdimensionale Funktion $f : \mathbb{R}^n \to \mathbb{R}$ kann nicht nur in die Koordinatenrichtungen abgeleitet werden, sondern in beliebige Richtungen $\boldsymbol{d} \in \mathbb{R}^n$ Um Formeln verständlicher zu notieren, werde ich so gut es geht Variablen $\boldsymbol{x} \in \mathbb{R}^n$ im Fall $n > 1$ immer durch fette Buchstaben notieren.. Diese **Richtungsableitung** wird geschrieben als $\nabla_d f(x)$. Analog zur ordinären Definition der Ableitung ist die Richtungsableitung definiert als:

$$\nabla_d f(x) = \lim_{h \to 0} \frac{f(x + h\boldsymbol{d}) - f(x)}{h}$$

f wird als differenzierbar bezeichnet, falls die Richtungsableitung in allen Richtungen existiert. Für ein differenzierbares f kann die Richtungsableitung mit Hilfe des Gradienten sehr einfach als Skalarprodukt berechnet werden:

$$\nabla_d f(x) = \nabla f(x)^T \boldsymbol{d}$$

Im Rahmen dieser Vorlesung gehen wir davon aus, dass $f \in C^1$, also dass die zu minimierende Funktion mindestens einmal differenzierbar ist. Im Allgemeinen ist es jedoch wichtig, anzumerken, dass f nicht umbedingt differenzierbar ist, nur weil partielle Ableitungen in alle Richtungen existieren.

2.2 Iterative Optimierung

Eine Funktion $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ kann numerisch durch iteratives "Ausprobieren" verschiedener Werte von $\boldsymbol{x}$ optimiert werden. Wir starten mit einem beliebigen Startwert $\boldsymbol{x}^{(0)}$ [1] , und bewegen uns dann bei jedem Schritt k in eine Richtung $\boldsymbol{d}^{(k)} \in \mathbb{R}^n$, welche uns hoffentlich näher zum Optimum bringt. Wir nutzen zusätzlich einen Parameter $\tau^{(k)} \in \mathbb{R}$, welcher die Schrittweite beschreibt.

$$\begin{aligned} \boldsymbol{x}^{(0)} &\in \mathbb{R}^n \\ \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} + \tau^{(k)} \boldsymbol{d}^{(k)} \end{aligned}$$

In der Praxis wird oft einfach $\boldsymbol{x}^{(0)} = \mathbf{0}$ gewählt. Ist jedoch f nicht quasikonvex, so kann eine schlechte Wahl von $\boldsymbol{x}^{(0)}$ dazu führen, dass das Verfahren nur gegen ein lokales Minimum kovergiert, welches oft kein globales Minimum ist und somit in vielen Fällen ein schlechtes Ergebnis.

2.2.1 Welche Richtung ist optimal?

Das Optimale $\boldsymbol{x}^{(k+1)}$ ist das, das $f(\boldsymbol{x}^{(k+1)}$ minimiert. Wir können $f(\boldsymbol{x}^{(k+1)})$ für kleine Schrittweiten $\tau^{(k)}$ folgendermaßen approximieren (**Taylor-Approximation**):

$$f(\boldsymbol{x}^{(k+1)}) = f(\boldsymbol{x}^{(k)} + \tau^{(k)} \boldsymbol{d}^{(k)}) \approx f(\boldsymbol{x}^{k}) + \tau^{(k)} \nabla f(\boldsymbol{x}^{(k)}) \boldsymbol{d}^{(k)}$$

Da wir den Wert von f möglichst stark verringern wollen, sollte $\tau^{(k)} \nabla f(\boldsymbol{x}^{(k)}) \boldsymbol{d}^{(k)}$ ein negativer Term mit möglichst großem Betrag sein. Dabei können wir $\tau^{(k)}$ aber nicht zu hoch wählen, da sonst die Approximation ungenau wird.

Nach der Definition des Skalarprodukts gilt:

$$\nabla f(\boldsymbol{x}^{(k)}) \boldsymbol{d}^{(k)} = ||\nabla f(\boldsymbol{x}^{(k)})||\ ||\boldsymbol{d}^{(k)}|| \cos(\theta) \tag{2.1}$$

Wobei θ der Winkel zwischen $\nabla f(\boldsymbol{x}^{(k)})$ und $\boldsymbol{d}^{(k)}$ ist. Diese Gleichung ist minimal, wenn $\cos(\theta) = -1$, also $\theta = 180$. Dementsprechend muss $\boldsymbol{d}^{(k)}$ in die umgekehrte Richtung von $\nabla f(\boldsymbol{x}^{(k)})$ zeigen, also $\boldsymbol{d}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)})$. [2]

Mit dieser optimalen Wahl der Abstiegsrichtung $\boldsymbol{d}^{(k)}$ erhalten wir das **Gradientenverfahren** (auch bekannt als "Gradientenabstieg", auf Englisch "**Gradient Descent**"):

$$\begin{aligned} \boldsymbol{x}^{(0)} &\in \mathbb{R}^n \\ \boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} - \tau^{(k)} \nabla f(\boldsymbol{x}^{(k)}) \end{aligned}$$

2.3 Bestimmung der Schrittweite

Die Wahl der Schrittweite $\tau^{(k)}$ ist für das Gradientenverfahren von entscheidender Bedeutung. Ist $\tau^{(k)}$ zu klein, wird die Konvergenz des Algorithmus oft deutlich verlangsamt. Ist $\tau^{(k)}$ zu groß, wird die Approximation ungenau. Dies kann dazu führen, dass das Verfahren über das Ziel hinausschießt, und im schlimmsten Fall eventuell gar nicht konvergiert.

Die Bestimmung eines optimalen Werts für $\tau^{(k)} \in \mathbb{R}$ ist ein eigenes Optimierungsproblem. In einigen Fällen ist ein dauerhaft konstantes τ genug, um die Konvergenz zu garantieren. In diesen Fällen ist eine konstante Schrittweite oft effizienter.

[1] Ich schreibe hier $\boldsymbol{x}^{(k)}$ mit der k in Klammern, um klarzustellen, dass $\boldsymbol{x}^{(k)}$ nicht "$\boldsymbol{x}$ hoch k" ist, sondern "das k-te $\boldsymbol{x}$". Eigentlich würde ich das am liebsten als $\boldsymbol{x}_k$ notieren, aber da $\boldsymbol{x} \in \mathbb{R}^n$ ist, ist diese Notation viel zu leicht mit der Notation x_k für die verschiedenen Komponenten von $\boldsymbol{x}$ zu verwechseln ($\boldsymbol{x} = (x_1, \ldots, x_n)^T$). Vermutlich werde ich an einigen Stellen aus Versehen trotzdem $\boldsymbol{x}^k$ schreiben, ich entschuldige mich im Voraus. (´-w-‘)

[2] Technisch gesehen wird die Gleichung durch $\boldsymbol{d}^{(k)} = -\alpha \cdot \nabla f(\boldsymbol{x}^{(k)})$ mit möglichst großen $\alpha \in \mathbb{R}$ minimiert. Die "Rolle" dieses Parameters α wir jedoch bereits durch unseren Schrittweiten-Parameter $\tau^{(k)}$ abgedeckt, und es gilt auch für dieses theoretische α, dass die Approximation sehr ungenau wird, wenn wir ein hohes α wählen.

Um ein optimales $\tau^{(k)}$ für eine gegebene Iteration k zu finden, definieren wir eine neue Funktion $h : \mathbb{R} \to \mathbb{R}$:

$$h(\tau) = f(\boldsymbol{x}^{(k)} + \tau \boldsymbol{d}^{(k)})$$

$$\tau^{(k)} = \underset{\tau \in \mathbb{R}}{\operatorname{argmin}}\ h(\tau)$$

Um die Ableitung von h zu finden, beschreiben wir den Term $\boldsymbol{x}^{(k)} + \tau \boldsymbol{d}^{(k)}$ als eine eigene eindimensionale Funktion $g(\tau)$. Dann ist $h(\tau) = f(\boldsymbol{g(\tau)})$, gemäß der mehrdimensionalen Kettenregel gilt also:

$$\begin{aligned}
\frac{d}{d\tau} h(\tau) &= \frac{d}{d\tau} f(\boldsymbol{g(\tau)}) \\
&= \sum_{i=1}^{n} \left(\frac{d}{d\tau} \boldsymbol{g(\tau)} \right) \frac{\partial}{\partial g(\tau)_i} f(\boldsymbol{g(\tau)}) \\
&= \sum_{i=1}^{n} d_i^{(k)} \frac{\partial}{\partial g(\tau)_i} f(\boldsymbol{g(\tau)}) \\
&= \nabla f(\boldsymbol{g(\tau)}) \cdot \boldsymbol{d}^{(k)} \\
&= \nabla f(\boldsymbol{x}^{(k)} + \tau \boldsymbol{d}^{(k)}) \boldsymbol{d}^{(k)}
\end{aligned}$$

Eine exakte Bestimmung des Optimalwerts ist mit hohem Aufwand Verbunden und ist oft sowieso nur von geringem Vorteil, da in der Regel auch bei wiederholter optimaler Wahl von $\tau^{(k)}$ für den letztendlichen Gradientenabstieg viele Schritte nötig sind. Stattdessen werden in der Praxis approximative Lösungen gefunden, welche bestimmte **Qualitätsbedingungen** erfüllen.

Eine einfache Suche nach einem beliebigen Wert, welcher den Funktionswert reduziert, also

$$f(\boldsymbol{x}^{(k)} + \tau^{(k)} \boldsymbol{d}^{(k)}) - f(\tau^{(k)}) \geq 0,$$

ist dabei nicht genug, da bei hinreichend schneller Konvergenz von $f(\boldsymbol{x}^{(k)} + \tau^{(k)} \boldsymbol{d}^{(k)}) - f(\tau^{(k)})$ gegen 0 eventuell nie das Optimum erreicht wird.

2.3.1 Die Armijo-Bedingung

Die **Armijo-Bedingung** hat das Ziel, sicherzustellen, dass die Zielfunktion bei einem Schritt um die gegebenen Schrittweite $\tau^{(k)}$ hinreichend reduziert wird. Die Bedingung ist gegeben durch die folgende Ungleichung:

$$f(\boldsymbol{x}^{(k)} + \tau^{(k)} \boldsymbol{d}^{(k)}) \leq f(\boldsymbol{x}^{(k)}) + \delta \tau^{(k)} \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)}$$

Die minimal notwendige Reduktion wird dabei durch einen Parameter $\delta \in (0, 1)$ gesteuert. Ein üblicher Wert ist dabei $\delta \approx 10^{-4}$. Die Multiplikation mit dem Gradienten bedeutet, dass bei einem steileren Gradienten auch nach einer höheren Reduktion gesucht wird.

2.3.2 Die Wolfe-Bedingungen

Die **schwache Wolfe-Bedingung** fordert, dass der Gradient an der Zielposition entweder weniger steil als an der Startposition ist oder das Vorzeichen ändert:

$$-\nabla f(\boldsymbol{x}^{(k)} + \tau^{(k)} \boldsymbol{d}^{(k)})^T \boldsymbol{d}^{(k)} \leq -\eta \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)}$$

Das Minus auf beiden Seiten kommt davon, dass bei einer Richtung $\boldsymbol{d}^{(k)}$, welche zu einem Abstieg führt, notwendigerweise $\nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)}$ negativ ist.

Wir wählen hier $\eta \in (\delta, 1)$. In der Regel wird η deutlich größer als δ gewählt, Wikipedia zitiert einen Richtwert von $\eta \approx 0.9$.

Bei der **starken Wolfe-Bedingung** wird der Betrag der Terme genommen, also gefordert, dass auch bei Vorzeichenwechsel die Steigung nach dem Schritt weniger steil als davor sein soll:

$$|\nabla f(\boldsymbol{x}^{(k)} + \tau^{(k)} \boldsymbol{d}^{(k)})^T \boldsymbol{d}^{(k)}| \leq \eta |\nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)}|$$

Intuitiv verhindern die Wolfe-Bedingungen ineffizient kleine Schritte, bei denen man ohne Probleme in die gleiche Richtung weitergehen könnte.

Schrittweiten, welche sowohl die starke Wolfe-Bedingung als auch die Armijo-Bedingung [3] erfüllen, existieren unter den von uns gemachten Annahmen immer. Als Erinnerung: Wir haben folgende Annahmen gemacht:

- $f(x)$ ist kontinuierlich differenzierbar ($f(x) \in C^1$)
- $\boldsymbol{d}^{(k)}$ ist eine Abstiegsrichtung an der Stelle $\boldsymbol{x}^{(k)}$ (also $\nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)} < 0$)
- $0 \leq \delta \leq \eta \leq 1$

Für die schwache Wolfe-Bedingung muss zusätzlich gegeben sein, dass die Menge $\{f(\boldsymbol{x}^{(k)} + \tau \boldsymbol{d}^{(k)}) \mid \tau > 0\}$ von unten beschränkt ist.

Beweis, nach Nocedal und Wright [4] (In der Vorlesung nicht gegeben):

- $h(\tau) = f(\boldsymbol{x}^{(k)} + \tau \boldsymbol{d}^{(k)})$ ist nach Annahme für alle τ nach unten beschränkt.
- Da $\boldsymbol{d}^{(k)}$ eine Abstiegsrichtung und δ positiv ist, ist die Linie $l_{Arm}(\tau) := f(\boldsymbol{x}^{(k)}) + \delta \tau \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)}$ nach unten unbeschränkt.
- Es gilt außerdem $h(0) = l_{Arm}(0)$ und $l_{Arm}(0) = \delta h'(0) < h'(0)$, also muss für kleine τ in Abstiegsrichtung gelten, dass $h(\tau) < l_{Arm}(\tau)$.
- Somit muss $l_{Arm}(\tau)$ den Graphen von $h(\tau)$ für mindestens einen Wert $\tau > 0$ schneiden. Sei τ' der kleinste Wert, bei dem es das tut.
- Es folgt $\forall \tau < \tau' : h(\tau) < l_{Arm}(\tau)$, da sonst τ' nicht der minimale Schnittpunkt wäre. Also erfüllen alle $\tau \in (0, \tau')$ die Armijo-Bedingung.
- Nach dem Mittelwertsatz muss es nun ein $\tau'' \in (0, \tau')$ geben, sodass

$$f(\boldsymbol{x}^{(k)} + \tau' \boldsymbol{d}^{(k)}) - f(\boldsymbol{x}^{(k)}) = \tau' \nabla f(\boldsymbol{x}^{(k)} \tau'' \boldsymbol{d}^{(k)})^T \boldsymbol{d}^{(k)}$$

- Da τ' ein Schnittpunkt ist, gilt nun

$$\begin{aligned}
& f(\boldsymbol{x}^{(k)} + \tau' \boldsymbol{d}^{(k)}) = f(\boldsymbol{x}^{(k)}) + \tau' \delta \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)} \\
\implies & f(\boldsymbol{x}^{(k)} + \tau' \boldsymbol{d}^{(k)}) - f(\boldsymbol{x}^{(k)}) = \tau' \delta \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)} \\
\implies & \tau' \nabla f(\boldsymbol{x}^{(k)} \tau'' \boldsymbol{d}^{(k)})^T \boldsymbol{d}^{(k)} = \tau' \delta \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)} \\
\implies & \nabla f(\boldsymbol{x}^{(k)} \tau'' \boldsymbol{d}^{(k)})^T \boldsymbol{d}^{(k)} = \delta \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)} \\
\overset{\eta > \delta}{\implies} & \nabla f(\boldsymbol{x}^{(k)} \tau'' \boldsymbol{d}^{(k)})^T \boldsymbol{d}^{(k)} > \eta \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}^{(k)}
\end{aligned}$$

Also erfüllt τ'' sowohl die Armijo-Bedingung als auch die schwache Wolfe-Bedingung. Desweiteren ist der Term auf der linken Seite der finalen Ungleichung negativ, also gilt sogar die starke Wolfe-Bedingung. Da die Ungleichungen strikt sind existiert desweiteren ein Intervall um τ'', welches die beiden Bedingungen ebenfalls erfüllt.

2.3.3 Line Search

Ein einfaches Verfahren zur Bestimmung eines geeigneten τ ist die sogenannte **Backtracking Line Search**:

- Definiere $\tau^{(0)} = 1$

[3] In dieser Vorlesung wurden die starke Wolfe-Bedingung und die Armijo-Bedingung als seperate Bedingungen vorgestellt. In der Literatur wird die Armijo-Bedingung oft als Teil der starken Wolfe-Bedingung definiert, mit "Erfüllung der starken Wolfe-Bedingungen" ist dann die Erfüllung beider Bedingungen gemeint.

[4] J. Nocedal, S. J. Wright: Numerical Optimization, Springer, 2006, Seite 35ff

- Falls die Armijo-Bedingung nicht erfüllt ist, probiere als nächstes $\tau^{(k+1)} = \beta\tau^{(k)}$, wobei $\beta \in (0, 1)$.

Unter Nutzung des Gradienten lässt sich auch ein **Interpolationsverfahren** anwenden, welches schneller konvergiert:

- Definiere $\tau^{(0)} = 1$
- Betrachte die Quadratische Taylor-Approximation von $h(\tau)$:

$$h_q(\tau) = \left(\frac{h(\tau^{(0)}) - h(0) - \tau^{(0)}h'(0)}{{\tau^{(0)}}^2}\right)\tau^2 + h'(0)\tau + h(0)$$

  Diese ist minimiert durch

$$\tau^{(1)} = \frac{h'(0){\tau^{(0)}}^2}{2(h(\tau^{(0)}) - h(0) - h'(0)\tau^{(0)}}$$

- Solange $\tau^{(k)}$ die Armijo-Bedingung noch nicht erfüllt, bestimme $\tau^{(k+1)}$ durch "kubische Interpolation mit $h(0), h'(0), h(\tau^{(0)}, h(\tau^{(1)})$ [5]

Line Search Verfahren, welche die Wolfe-Bedingungen sicherstellen, sind komplizierter. Solche Verfahren bestehen aus zwei Komponenten:

- **Bracketing** erweitert das Suchintervall, bis darin geeignete Schrittweiten garantiert werden können.
- **Zooming** reduziert das Suchintervall, bis darin per Interpolationsverfahren eine geeignete Schrittweite gefunden wird.

2.3.4 Conjugate Gradient-Verfahren

Angenommen, unsere Zielfunktion ist eine quadratische Funktion in folgender Form:

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x} \tag{2.2}$$

Das sogenannte **Conjugate Gradient-Verfahren** funktioniert dann folgendermaßen:

- Wähle als erste Richtung den Gradienten an der Startposition:

$$\boldsymbol{d}^{(0)} = -\nabla f(\boldsymbol{x}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^0$$

- Bestimme die optimale Schrittweite:

$$\tau^{(k)} = \frac{|\nabla f(\boldsymbol{x}^{(k)})|}{{\boldsymbol{d}^{(k)}}^T A\boldsymbol{d}^{(k)}}$$

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \tau^{(k)}\boldsymbol{d}^{(k)}$$

- Wähle eine neue Richtung so, dass $\boldsymbol{d}^{(k+1)}$ und $\boldsymbol{d}^{(k)}$ gemäß der von A induzierten Metrik orthogonal sind, also ${\boldsymbol{d}^{(k+1)}}^T \cdot A \cdot \boldsymbol{d}^{(k)} = 0$. Dies wird garantiert durch:

$$\beta^{(k)} = \frac{|\nabla f(\boldsymbol{x}^{(k+1)})|^2}{|\nabla f(\boldsymbol{x}^{(k)})|^2}$$

$$\boldsymbol{d}^{(k+1)} = -\nabla f(\boldsymbol{x}^{k+1}) + \beta^{(k)}\boldsymbol{d}^{(k)}$$

Das Minimieren von f entspricht genau dem Lösen des Gleichungssystems $Ax = b$. Das CG-Verfahren garantiert dadurch für die Matrix $A = \mathbb{R}^{n\times n}$ eine bei exakter Zahlendarstellung exakte Lösung nach nur n Schritten. Wenn die **Konditionszahl** von A, definiert als der größte Eigenwert geteilt durch den kleinsten, klein ist, konvergiert es sogar wesentlich schneller.

Für nicht quadratische Funktionen kann das Conjugate-Gradient verfahren nahezu identisch angewandt werden, allerdings existiert im Allgemeinen für die Schrittweitenbedingung keine analytische Lösung , weshalb wieder Line Search nötig wird.

[5] Die Vorlesung war hier extrem vage - ich vermute, wir interpolieren iterativ immer $h(0), h'(0), h(\tau^{(k-1)}), h(\tau^{(k)})$? Wie genau man eine kubische Interpolation durchführt wurde aber nicht erklärt.

Chapter 3

Newton und Quasi-Newton-Verfahren

3.1 Das Newton-Verfahren

Das **Newton-Verfahren** ähnelt dem Gradientenverfahren, verwendet jedoch auch die Krümmung der Funktion, also die zweite Ableitung, gegeben durch die Hesse-Matrix H_f:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \tau^{(k)} H_f^{-1} \boldsymbol{x}^{(k)} \nabla f(\boldsymbol{x}^{(k)})$$

Dadurch ermöglicht das Newton-Verfahren größere Schritte in Richtungen mit schwacher Krümmung, bei denen die Approximation genauer und somit das "Risiko" geringer ist.

Zur Erinnerung: Die Hesse-Matrix H_f ist gegeben durch:

$$H_f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

Die Hesse-Matrix ist symmetrisch, da $\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial^2 f}{\partial x_j \partial x_i}$.

Während das Gradientenverfahren also mit einer linearen Taylor-Approximation arbeitet, nutzt das Newton-Verfahren eine quadratische Taylor-Approximation:

$$\begin{aligned} \boldsymbol{m}^{(k)}(\boldsymbol{d}) &:= f(\boldsymbol{x}^{(k)} + \boldsymbol{d}) \approx f(\boldsymbol{x}^{(k)}) + \boldsymbol{d}^T \nabla f(\boldsymbol{x}^{(k)}) + \frac{1}{2} \boldsymbol{d}^T H_f \boldsymbol{x}^{(k)} \boldsymbol{d} \\ \nabla \boldsymbol{m}^{(k)}(\boldsymbol{d}) &= \nabla f(\boldsymbol{x}^{(k)}) + H_f(\boldsymbol{x}^{(k)}) \boldsymbol{d} \end{aligned}$$

Für das optimale d ist $\nabla \boldsymbol{m}^{(k)}(\boldsymbol{d}) = 0$ und wir erhalten:

$$\begin{aligned} H_f(\boldsymbol{x}^{(k)}) \boldsymbol{d} &= -\nabla f(\boldsymbol{x}^{(k)}) \\ \implies \boldsymbol{d} &= -H_f^{-1} \boldsymbol{x}^{(k)} \nabla f(\boldsymbol{x}^{(k)}) \end{aligned}$$

Die Optimale Schrittweite $\tau^{(k)}$ muss auch beim Newton-Verfahren durch Line Search bestimmt werden.

3.2 Konvergenzordnungen

Sei s eine beliebige Folge und $\overline{s} := \lim_{k \to \infty} s_k$. Wir betrachten nun die Konvergenzordnung der Folge s, also die Geschwindigkeit, mit der die Folge konvergiert. In dieser Vorlesung unterscheiden wir folgende Konvergenzordnungen:

- s konvergiert **linear**, wenn:

$$\lim_{k \to \infty} \frac{|s_{k+1} - \overline{s}|}{|s_k - \overline{s}|} < 1$$

ein Beispiel für eine linear konvergierende Folge ist $s_k = 0.9^k$. Das Gradientenverfahren konvergiert ebenfalls linear.

- s konvergiert **superlinear**, wenn:

$$\lim_{k\to\infty} \frac{|s_{k+1} - \overline{s}|}{|s_k - \overline{s}|} = 0$$

ein Beispiel für eine superlinear konvergierende Folge ist $s_k = \frac{1}{k!}$.

- s konvergiert **quadratisch**, wenn:

$$\lim_{k\to\infty} \frac{|s_{k+1} - \overline{s}|}{|s_k - \overline{s}|^2} < 1$$

ein Beispiel für eine superlinear konvergierende Folge ist $s_k = 0.9^{(2^k)}$.

Das Newton-Verfahren konvergiert für glatte, konvexe Funktionen nahe der Lösung quadratisch. Nominal ist es also wesentlich effizienter als Gradientenabstieg. Dabei ist jedoch zu berücksichtigen, dass die Berechung und Invertierung der Hesse-Matrix wesentlich aufwendiger ist als die Berechnung des Gradienten. Demenssprechend ist ein einzelner Iterationsschritt des Newton-Verfahrens auch wesentlich aufwendiger als ein einzelner Schritt des Gradientenverfahrens.

3.2.1 Konvergenzradius des Newton-Verfahren

Damit $\boldsymbol{d}^{(k)} = -H_f^{-1}\boldsymbol{x}^{(k)}\nabla f(\boldsymbol{x}^{(k)})$ eine Abstiegsrichtung ist, muss H_f positiv definit sein, also die Krümmung positiv. Bei negativer Krümmung läuft das Newton-Verfahren in die falsche Richtung! Dies ist kein Problem für konvexe Funktionen, da diese überall positive Krümmung haben. Bei nicht konvexen Funktionen muss die Hesse-Matrix erst manipuliert werden, damit alle ihre Eigenwerte positiv werden.

3.3 Quasi-Newton Verfahren

Die Idee hinter Quasi-Newton-Verfahren ist es, die Hesse-Matrix mithilfe der Gradienten sukzessiv zu approxmieren. Man ersetzt also in der Iterationsformel des Newton-Verfahrens die Matrix H_f durch eine Approximation B:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \tau^{(k)}(B^k)^{-1}\nabla f(\boldsymbol{x}^{(k)})$$

Die einfachste Möglichkeit ist eine lineare Taylor-Approximation des Gradienten. Sei $s = \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}$. Dann haben wir:

$$\begin{aligned} \nabla f(\boldsymbol{x}^{(k)} + \boldsymbol{s}) &\approx \nabla f(\boldsymbol{x}^{(k)}) + H_f(\boldsymbol{x}^{(k)})\boldsymbol{s} \\ \implies H_f(\boldsymbol{x}^{(k)})(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}) &\approx \nabla f(\boldsymbol{x}^{(k+1)}) - \nabla f(\boldsymbol{x}^{(k)}) \end{aligned}$$

Wir suchen also in jedem Schritt eine Matrix $B^{(k)}$, sodass:

$$B^{(k)}(\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}) = \nabla f(\boldsymbol{x}^{(k+1)}) - \nabla f(\boldsymbol{x}^{(k)}) \tag{3.1}$$

Abgekürzt schreiben wir auch:

$$B^{(k)}\boldsymbol{s}^{(k)} = \boldsymbol{y}^{(k)}$$

Aus $B^{(k)}\boldsymbol{s}^{(k)} = \boldsymbol{y}^{(k)}$ folgt ${\boldsymbol{s}^{(k)}}^T B^{(k)}\boldsymbol{s}^{(k)} = {\boldsymbol{s}^{(k)}}^T\boldsymbol{y}^{(k)}$. Wir gehen von einer konvexen Funktion f aus, also ist $B^{(k)}$ positiv definit. Daraus folgt ${\boldsymbol{s}^{(k)}}^T B^{(k)}\boldsymbol{s}^{(k)} > 0$, damit gilt auch ${\boldsymbol{s}^{(k)}}^T\boldsymbol{y}^{(k)} > 0$. Diese Ungleichung können wir als Bedingung für die Schrittweitensuche nutzen.

Um die Anzahl an möglichen $B^{(k)}$ einzuschränken, können wir als weitere Bedingung fordern, dass $B^{(k)}$ in jedem Schritt möglichst ähnlich zur letzten Iteration sein soll:

$$B^{(k)} = \min_B ||B - B^{(k-1)}||$$

$||\cdot||$ kann eine beliebige Matrixnorm sein. An diesem Punkt führen verschiedene Normen zu verschiedenen Unterverfahren. Ein beliebtes Verfahren ist das **Davidon-Fletcher-Powell Verfahren**, welches eine sog. gewichtete Frobeniusnorm benutzt.

3.3.1 Das Davidon-Fletcher-Powell Verfahren

Beim Davidon-Fletcher-Powell (DFPV)-Verfahren werden die Approximation B^{k+1} und ihr Inverses $Q^{(k+1)}$ direkt aus den vorherigen Werten berechnet:

$$\begin{aligned}\rho^{(k)} &= \frac{1}{(\boldsymbol{y}^{(k)})^\top \boldsymbol{y}^{(k)}} \\ B^{k+1} &= (I - \rho^{(k)}\boldsymbol{y}^{(k)}(\boldsymbol{s}^{(k)})^\top)B^{(k)}(I - \rho^{(k)}\boldsymbol{s}^{(k)}(\boldsymbol{y}^{(k)})^\top) + \rho^{(k)}\boldsymbol{y}^{(k)}(\boldsymbol{y}^{(k)})^\top \\ Q^{(k+1)} &= Q^{(k)} - \frac{Q^{(k)}\boldsymbol{y}^{(k)}(\boldsymbol{y}^{(k)})^\top Q^{(k)}}{(\boldsymbol{y}^{(k)})^\top Q^{(k)}\boldsymbol{y}^{(k)}} + \frac{\boldsymbol{s}^{(k)}(\boldsymbol{s}^{(k)})^\top}{(\boldsymbol{y}^{(k)})^\top \boldsymbol{s}^{(k)}}\end{aligned}$$

3.3.2 Das Broyden-Fletcher-Goldfarb-Shanno Verfahren

Das Broyden-Fletcher-Goldfarb-Shanno (BFGS)-Verfahren ist eine etwas effizientere Variante des DFPV-Verfahrens, bei dem $Q^{(k+1)}$ direkt folgendermaßen berechnet wird:

$$Q^{k+1} = (I - \rho^{(k)}\boldsymbol{s}^{(k)}(\boldsymbol{y}^{(k)})^\top)Q^{(k)}(I - \rho^{(k)}\boldsymbol{y}^{(k)}(\boldsymbol{s}^{(k)})^\top) + \rho^{(k)}\boldsymbol{y}^{(k)}(\boldsymbol{y}^{(k)})^\top$$

Q^{k+1} ist immer positiv definit, wenn $Q^{(k)}$ positiv definit war.

Das volle BFGS-Verfahren verläuft dann also folgendermaßen:

1. $\boldsymbol{x}^{(0)} \in \mathbb{R}$, $Q^{(0)} \in \mathbb{R}^{n\times n}$ (z.B. $Q^{(0)} = I_n$)
2. $\boldsymbol{d}^{(k)} = Q^{(k)}\nabla f(\boldsymbol{x}^{(k)})$
3. Bestimme optimale Schrittweite $\tau^{(k)}$ durch Line Search mit Wolfe-Bedingungen. Es sollte immer zuerst $\tau^{(k)} = 1$ getestet werden, da diese Schrittweite zur schnellsten Konvergenz führt und in den meisten Iterationen in der Praxis akzeptiert wird.
4. Berechne neue Lösung $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \tau^{(k)}\boldsymbol{d}^{(k)}$
5. Berechne neue Approximation der inversen Hesse Matrix:

$$\begin{aligned}\boldsymbol{s}^{(k)} &= \boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)} \\ \boldsymbol{y}^{(k)} &= \nabla f(\boldsymbol{x}^{(k+1)}) - \nabla f(\boldsymbol{x}^{(k)}) \\ \rho^{(k)} &= \frac{1}{(\boldsymbol{y}^{(k)})^\top \boldsymbol{y}^{(k)}} \\ Q^{k+1} &= (I - \rho^{(k)}\boldsymbol{s}^{(k)}(\boldsymbol{y}^{(k)})^\top)Q^{(k)}(I - \rho^{(k)}\boldsymbol{y}^{(k)}(\boldsymbol{s}^{(k)})^\top) + \rho^{(k)}\boldsymbol{y}^{(k)}(\boldsymbol{y}^{(k)})^\top\end{aligned}$$

Die Vorlesung gibt folgende Iterationszahlen für ein Beispielproblem aus der Praxis:

- Gradientenabstieg: 5264 Iterationen
- BFGS: 34 Iterationen
- Newton: 21 Iterationen (Aber wie bereits gesagt ist dafür jede Iteration sehr viel teurer als bei BFGS)

3.4 Least-Squares-Aufgaben

Eine sehr häufige Klasse von Zielfunktionen, welche inbesondere in der Statistik und somit auch im maschinellen Lernen wichtig sind, ist gegeben durch **Least Squares**, auch bekannt als L_2-Loss:

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j(\boldsymbol{x})^2$$

Wobei r_j in der Regel ein Klassifikationsfehler für einen Datenpunkt x_j, y_j ist, also:

$$r_j(\boldsymbol{x}) = (\hat{y}_j(\boldsymbol{x}) - y_j) \in \mathbb{R}$$

Wenn wir die Residuen r_j als einen **Residuenvektor** $\boldsymbol{r}(\boldsymbol{x}) = (r_1(\boldsymbol{x}), \ldots, r_m(\boldsymbol{x}))^\top$ schreiben, erhalten wir eine gekürzte Schreibweise des Problems:

$$f(x) = \frac{1}{2}||\boldsymbol{r}(\boldsymbol{x})||^2$$

Der Residuenvektor ist eine Funktion $r : \mathbb{R}^n \to \mathbb{R}^m$, also eine Funktion mit mehrdimensionaler Bildmenge. Die Ableitung einer solchen Funktion ist gegeben durch die sog. **Jacobimatrix**:

$$J_{\boldsymbol{r}}(\boldsymbol{x}) = \begin{pmatrix} \nabla r_1(\boldsymbol{x}^\top) \\ \vdots \\ \nabla r_m(\boldsymbol{x}^\top) \end{pmatrix} \in \mathbb{R}^{m \times n}$$

Im Least-Squares-Problem lässt sich damit der Gradient von $f(x)$ sehr kompakt darstellen:

$$\nabla f(\boldsymbol{x}) = \frac{1}{2} \sum_{j=1}^{m} 2 r_j(\boldsymbol{x}) \nabla r_j(\boldsymbol{x}) = J_{\boldsymbol{r}}(\boldsymbol{x})^\top \boldsymbol{r}(\boldsymbol{x})$$

Die Hesse-Matrix ist dann gegeben durch:

$$\begin{aligned} H_f(\boldsymbol{x}) &= \sum_{j=1}^{m} \nabla r_j(\boldsymbol{x}) \nabla r_j(\boldsymbol{x})^\top + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(\boldsymbol{x}) \\ &= J(\boldsymbol{x})^\top J(\boldsymbol{x}) + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(\boldsymbol{x}) \end{aligned}$$

3.4.1 Lineare Residuen

Ist die Residuenfunktion $\boldsymbol{r}(\boldsymbol{x})$ linear, dann ist $\boldsymbol{r}(\boldsymbol{x}) = J\boldsymbol{x} - \boldsymbol{b}$ mit $J \in \mathbb{R}^{m \times n}$. In diesem Fall lässt sich das Ergebnis weiter vereinfachen: Wir haben dann $J_{\boldsymbol{r}}(\boldsymbol{x}) = J$, und der zweite Teil der Hesse-Funktion fällt weg. Dann gilt also ganz einfach

$$H_f(\boldsymbol{x}) = J^\top J.$$

Das Problem der Optimierung von f ist dann durch folgendes lineares Gleichungssystem beschrieben:

$$\begin{aligned} \nabla f(\boldsymbol{x}) &= 0 \\ \implies J^\top \boldsymbol{r}(\boldsymbol{x}) &= 0 \\ \implies J^\top (J\boldsymbol{x} - \boldsymbol{b}) &= 0 \\ \implies J^\top J\boldsymbol{x} &= J^\top \boldsymbol{b} \end{aligned}$$

3.4.2 Nichtlineare Residuen

Für nichtlineare Residuen bietet es sich an, die Hesse-Matrix einfach durch der ersten Term zu approximieren:

$$\begin{aligned} H_f(\boldsymbol{x}) &= J(\boldsymbol{x})^\top J(\boldsymbol{x}) + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(\boldsymbol{x}) \\ &\approx J(\boldsymbol{x})^\top J(\boldsymbol{x}) \end{aligned}$$

Dies entspricht der in dieser Vorlesung bereits oft verwendeten linearen Taylor-Approximation einer Funktion. Anders als bei BFGS wird hier die spezifische Problemstruktur von Summen quadrierter

Residuen ausgenutzt. Wir können die Newtonsche Gleichung von ganz am Anfang nutzen, um ein simples lineares Gleichungssystem zu erhalten:

$$H_f(\boldsymbol{x}^{(k)})\boldsymbol{d} = -\nabla f(\boldsymbol{x}^{(k)}) = -J_r(\boldsymbol{x}^{(k)})^\top \boldsymbol{r}(\boldsymbol{x})$$
$$\implies J_r(\boldsymbol{x}^{(k)})^\top J_r(\boldsymbol{x}^{(k)})\boldsymbol{d}^{(k)} = -J_r(\boldsymbol{x}^{(k)})^\top \boldsymbol{r}(\boldsymbol{x})$$

Die Lösung ist dann eine optimale Abstiegsrichtung $\boldsymbol{d}^{(k)}$. Dieses Verfahren ist bekannt als **Gauss-Newton-Verfahren**, da es das Newton-Verfahren durch das Lösen einer Sequenz von linearen Gleichungssystem approximiert (Welche dann wiederum durch das Gauss-Verfahren gelöst werden können). Das Verfahren funktioniert genau dann gut, wenn ale Eigenwerte von $J^\top J$ deutlich positiv sind. Bei Eigenwerten Nahe null sind andere Verfahren notwendig, welche in dieser Vorlesung nicht besprochen werden, z.B. das Verfahren von Levenberg-Marquardt.

Chapter 4

Optimierung mit Nebenbedingungen

Bisher war der Suchraum für ein Optimum einer Funktion $f : \mathbb{R}^n \to \mathbb{R}$ immer der gesamte Urbildraum $\mathbb{R}^n$. In vielen Fällen ist der Suchraum aber eine Teilmenge $G \subset \mathbb{R}^n$, welche eingeschränkt ist durch Nebenbedingungen wie "$x_1 \leq 5$". Die Lösung eines Problems ohne Nebenbedingung gibt offensichtlich immer eine untere Schranke für die tatsächliche Lösung, da natürlich jede Lösung des Problems mit Nebenbedingungen auch eine Lösung des Problems ohne Nebenbedingungen ist. Oft, aber nicht immer, liegt das Optimum auf dem Rand von G. Wir gehen in der Regel davon aus, dass G Kompakt ist, da sonst in diesem Fall kein tatsächliches Optimum existiert.

Ein Optimierungsproblem wird im Allgemeinen geschrieben als:

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} & f(x) \\ \text{subject to} & c_i(\boldsymbol{x}) = 0 \quad \forall i \in \mathcal{E} \\ & c_i(\boldsymbol{x}) \geq 0 \quad \forall i \in \mathcal{I} \end{array}$$

4.1 Optimierung mit Gleichheitsbedingungen

Die m Gleichheitsbedingungen sind zusammengefasst durch eine vektorwertige Funktion $\boldsymbol{c(x)} : \mathbb{R}^n \to \mathbb{R}^m$ gegeben. Es gilt demenssprechend genau dann $\boldsymbol{c(x)} = \boldsymbol{0}$, wenn alle Gleichheitsbedingungen erfüllt sind. Ein Schritt in Richtung $\boldsymbol{d}$ ist dementsprechend genau dann erlaubt, wenn $\nabla \boldsymbol{c}^\top \boldsymbol{d} = 0$ und $\nabla f^\top \boldsymbol{d} \leq 0$, also wenn $\boldsymbol{d}$ eine Abstiegsrichtung ist und wenn sich der Wert von $\boldsymbol{c(x)}$ nach dem Schritt nicht ändert (und demenssprechend die Gleichheitsbedingungen danach immer noch erfüllt sind).

Es stellt sich heraus, dass in einem Extrempunkt die Gradienten ∇f und ∇c immer parallel sein müssen, also

$$f(\boldsymbol{x}) = \boldsymbol{\lambda}(\boldsymbol{x}) \quad \boldsymbol{\lambda} \in \mathbb{R}^m$$

Ist diese Bedingung nicht gegeben, so existiert ein Schritt, welcher f ändert, aber nicht c, und somit ist die momentane Position $\boldsymbol{x}$ kein Extremum. [1]

Diese Bedingung können wir nun ausnutzen. Angenommen, f ist eine Funktion $\mathbb{R}^n \to \mathbb{R}$ und c ist eine Funktion $\mathbb{R}^n \to \mathbb{R}^m$. Dann ist die **Lagrange-Funktion** $\mathcal{L}$ des Problems die folgende Funktion $\mathbb{R}^{n+m} \to \mathbb{R}$:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \boldsymbol{\lambda} \boldsymbol{c}(\boldsymbol{x})$$

Die Schreibweise "$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$" fand ich persönlich sehr gewöhnungsbedürftig und anfangs sehr verwirrend. Es ist wichtig, sich zu merken, dass $\boldsymbol{x}$ selbst ein n-dimensionaler Vektor ist. Letzendlich "erfinden" wir quasi eine neue Menge an Eingabevariablen, gegeben durch den Vektor $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_m)$, welchen wir hinten an den "Eingabevektor" $\boldsymbol{x} = (x_1, \ldots, x_n)^\top$ hängen. So erhalten wir einen neuen Vektor $\boldsymbol{x'} = (x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_m)^\top$, welchen wir dann in die Lagrange-Funktion einsetzten. Eine intuitivere,

[1] Falls dies intuitiv noch unklar ist empfehle ich als zusätzliche Erklärung das Video *Understanding Lagrange Multipliers Visually*, von *Serpentine Integral*.

und arguably korrektere, Schreibweise dafür wäre $\mathcal{L}(x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_m)$. Da das jedoch den meisten Mathematikern zu viel Schreibarbeit ist wird in der Praxis nunmal $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ geschrieben. Die "Original-Variablen" x_i werden oft als die **primalen Variablen** bezeichnet, die neuen Variablen λ_i hingegen als die **Lagrange-Multiplikatoren**.

Die Bedingung für ein Minimum von f kann nun mithilfe der Lagrange-Funktion ausgedrückt werden als:

$$\nabla_x \mathcal{L}(\boldsymbol{x}, \lambda) = \nabla f(\boldsymbol{x}) - \boldsymbol{\lambda} \nabla \boldsymbol{c}(\boldsymbol{x}) = 0$$

Wir können also nun unser Optimierungsproblem mit Gleichheitsbedingungen in ein äquivalentes Optimierungsproblem ohne Nebenbedingungen umformen. Wenn für eine Komponente i diese Bedingung bereits durch $\lambda_i = 0$ erfüllt ist, dann ist c_i eine **Nebenbedingung ohne Relevanz**, d.h. das Minimum unter Einschränkung durch die Nebenbedingung c_i ist dasselbe wie das Minimum ohne die Nebenbedingung. Man sagt in diesem Fall auch, dass die Nebenbedingung **inaktiv** ist. Die Menge der aktiven Nebenbedingungen wir auch als die **aktive Menge** bezeichnet.

4.2 Optimierung mit Ungleichheitsbedingungen

Auch hier ist die Parallelität der Gradienten eine notwendige Bedingung. Allerdings zählt nun das Vorzeichen von $\boldsymbol{\lambda}$, da wir $\boldsymbol{c}$ am Rand zwar nicht verringern können, aber beliebig erhöhen. Wir erhalten also die Bedingung

$$f(\boldsymbol{x}) = \boldsymbol{\lambda} \boldsymbol{c}(\boldsymbol{x}) \quad \boldsymbol{\lambda} \in \mathbb{R}^m_+$$

bzw.

$$\nabla_x \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \nabla f(\boldsymbol{x}) - \boldsymbol{\lambda} \nabla \boldsymbol{c}(\boldsymbol{x}) = 0, \quad \forall i \ \lambda_i > 0$$

Zusätzlich gilt am Optimum immer

$$\boldsymbol{\lambda} \boldsymbol{c}(\boldsymbol{x}) = 0,$$

da sich für jedes i entweder das Optimum auf dem Rand befindet, also $c_i(\boldsymbol{x}) = 0$, oder die Nebenbedingung inaktiv ist, also $\lambda_i = 0$. Wir sagen, eine Nebenbedingung ist **streng komplementär**, wenn immer nur genau einer dieser beiden Fälle auftritt.

Zusammenfassend sind die **notwendigen Bedingungen** für ein lokales Optimum die folgenden:

$$\begin{aligned} \nabla_x \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) &= 0 \\ c_i(\boldsymbol{x}) &= 0 \quad \forall i \in \mathcal{E} \\ \lambda_i c_i(\boldsymbol{x}) &= 0 \quad \forall i \in \mathcal{I} \\ c_i(\boldsymbol{x}) &\geq 0 \quad \forall i \in \mathcal{I} \\ \lambda_i &\geq 0 \quad \forall i \in \mathcal{I} \end{aligned}$$

Diese Bedingungen sind bekannt als die **Karush-Kuhn-Tucker-Bedingungen (KKT)**. Später werden wir sehen, dass wir durch die Einführung weiterer Lagrange-Multiplikatoren Ungleichheitsbedingungen als weitere Gleichheitsbedingungen ausdrücken können. So werden wir letztendlich bei der Folgenden deutlich übersichtlicheren Liste landen:

$$\begin{aligned} \nabla_x \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) &= \boldsymbol{0} \\ c_i(\boldsymbol{x}) &= 0 \quad \forall i \in (1, \ldots, m), \end{aligned}$$

oder noch genauer:

$$\begin{aligned} \tfrac{\partial \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})}{\partial x_i} &= 0 \quad \forall i \in (1, \ldots, n) \\ c_i(\boldsymbol{x}) &= 0 \quad \forall i \in (1, \ldots, m) \end{aligned}$$

4.3 Dualität

Mithilfe der Lagrange-Funktion können wir nun aus dem Originalproblem (auch **primales Problem** genannt)

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \quad & f(x) \\ \text{subject to} \quad & c_i(\boldsymbol{x}) = 0 \quad \forall i \in \mathcal{E} \\ & c_i(\boldsymbol{x}) \geq 0 \quad \forall i \in \mathcal{I} \end{aligned}$$

ein sogenanntes **duales Problem** erhalten:

$$q(\boldsymbol{\lambda}) := \inf_{\boldsymbol{x} \in \mathbb{R}^n} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}), \forall i : \lambda_i \geq 0$$

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} q(\boldsymbol{\lambda})$$

Intuitiv kann man sich vorstellen, dass für ein festes $\boldsymbol{x}$ die Lagrange-Funktion einen höheren Wert hat, je mehr Bedingungen erfüllt sind. Somit führt die Maximierung der dualen Funktion auch zu einer Optimierung der Originalfunktion. Die duale Problemlösung ist hauptsächlich dann sinnvoll, wenn es eine analytische Lösung zur Maximierung von $q(\boldsymbol{\lambda})$ gibt, und/oder wenn im dualen Problem keine Nebenbedingungen mehr existieren.

Lemma: Für beliebige $\hat{\boldsymbol{\lambda}} \in \mathbb{R}^m$, $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ gilt $q(\hat{\boldsymbol{\lambda}}) \leq f(\hat{\boldsymbol{x}} \in \mathbb{R}^n)$.
Beweis:

$$q(\hat{\boldsymbol{\lambda}}) = \inf_{\boldsymbol{x}} f(\boldsymbol{x}) - \hat{\boldsymbol{\lambda}}^\top \boldsymbol{c}(\boldsymbol{x}) \leq f(\hat{\boldsymbol{x}}) - \hat{\boldsymbol{\lambda}}^\top \boldsymbol{c}(\boldsymbol{x}) \leq f(\hat{\boldsymbol{x}})$$

Also erhalten wir eine untere Schranke für $f(\boldsymbol{x})$, indem wir $q(\hat{\boldsymbol{\lambda}})$ maximieren.

Lemma: $q(\boldsymbol{\lambda})$ ist konkav.
Beweis:

$$\begin{aligned} \mathcal{L}(\boldsymbol{x}, (1-\alpha)\boldsymbol{\lambda}_0 + \alpha\boldsymbol{\lambda}_1) &= f(\boldsymbol{x}) - (1-\alpha)\boldsymbol{\lambda_0}\boldsymbol{c}(x) - \alpha\boldsymbol{\lambda}_1\boldsymbol{c}(x) \\ &= (1-\alpha)f(\boldsymbol{x}) + \alpha f(x) - (1-\alpha)\boldsymbol{\lambda_0}\boldsymbol{c}(x) - \alpha\boldsymbol{\lambda}_1\boldsymbol{c}(x) \\ &= (1-\alpha)\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_0) + \alpha\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_1) \end{aligned}$$

Für beliebige Funktionen f, g gilt im Allgemeinen $\inf_x(f(x) + g(x)) \leq \inf_x f(x) + \inf_x g(x)$, also:

$$\inf_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, (1-\alpha)\boldsymbol{\lambda}_0 + \alpha\boldsymbol{\lambda}_1) \leq (1-\alpha) \inf_x \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_0) + \alpha \inf_x \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_1). \quad \square$$

In vielen Fällen ist das duale Problem einfacher zu lösen als das primale Problem. Das duale Problem liefert eine untere Schranke für die Lösung des Originalproblems:

$$\inf_x f(\boldsymbol{x}) - \boldsymbol{\lambda}^\top \boldsymbol{c}(\boldsymbol{x}) \leq f(\boldsymbol{x}) - \boldsymbol{\lambda}^\top \boldsymbol{c}(\boldsymbol{x}) \leq f(\boldsymbol{x})$$

Wir sprechen von **starker Dualität**, wenn das Optimum des dualen Problems auch das primale Problem minimiert. Dies ist der Fall unter folgenden Bedingungen:

- $f(x)$ ist konvex
- Die gültige Menge G ist konvex
- $\mathcal{L}(x, \hat{\boldsymbol{\lambda}})$ ist **streng** konvex, wobei $\hat{\boldsymbol{\lambda}}$ die Lösung des dualen Problems ist.

Für ein "gut benommenes" Problem ist die Lösung des dualen Problems äquivalent zum Wert der konvexen Entspannung des primalen Problems, also zu dem Problem, was wir erhalten, wenn wir das feasible

Set G durch seine konvexe Hülle und die Funktion $f(\boldsymbol{x})$ durch ihre konvexe Schließung ersetzen[2]. "Gut benommen" bedeutet in diesem Kontext, dass das Minimum die KKT-Bedingungen erfüllt - Gegenbeispiele sind hier oft Probleme mit seltsamen Feasible Sets, bei denen z.B. Ungleichheitsbedingungen existieren, welche nur einen einzigen Punkt zulassen. Es gibt eine lange Reihe an sogenannten Regularitätsbedingungen, die zu dieser Bedingung äquivalent sind. Einige Beispiele sind:

- **Linearity Constraint Qualification (LCQ)**: Alle c_i sind affin.
- **Linear Independence Constraint Qualification (LICQ)**: Die Gradienten der Ungleichheitsbedingungen und die Gradienten der Gleichheitsbedingungen sind am Minimum linear unabhängig.
- **Slater's Condition**: In einem Problem mit konvexem f, konvexen Ungleichheitsbedingungen und affinen Gleichheitsbedingungen existiert ein Punkt $\boldsymbol{x}$, welcher alle Gleichheitsbedingungen erfüllt und alle Ungleichheitsbedingungen Strikt erfüllt (also $c_i(\boldsymbol{x}) > 0$ anstatt nur $c_i(\boldsymbol{x}) \geq 0$).
- etc. etc.

4.3.1 Beispiel: Optimierung mit Hilfe des dualen Problems

Gegeben sei folgendes konvexes Problem:

$$\begin{array}{ll} \underset{x_1, x_2 \in \mathbb{R}}{\operatorname{argmin}} & \frac{1}{2}(x_1^2 + x_2^2) \\ \text{s.t.} & x_1 - 1 \geq 0 \end{array}$$

Die Lagrange-Funktion ist dann:

$$\mathcal{L}(x_1, x_2, \lambda) = \frac{1}{2}(x_1^2 + x_2^2) - \lambda(x_1 - 1)$$

Nötige Bedingung ist, dass die partiellen Ableitung von $\mathcal{L}$ Null sind, also:

$$\begin{aligned} \frac{\partial \mathcal{L}(x_1, x_2, \lambda)}{\partial x_1} &= x_1 - \lambda = 0 \\ \frac{\partial \mathcal{L}(x_1, x_2, \lambda)}{\partial x_2} &= x_2 = 0 \end{aligned}$$

Wir setzen diese Werte in die Lagrange-Funktion ein und erhalten:

$$\mathcal{L}(x_1, x_2, \lambda) = \frac{1}{2}(x_1^2 + x_2^2) - \lambda(x_1 - 1) = \frac{1}{2}(\lambda^2 + 0^2) - \lambda(\lambda - 1) = -\frac{1}{2}\lambda^2 + \lambda$$

Das Maximum dieser Funktion ist $\lambda = 1$. Unsere Lösung ist also $x_1 = \lambda = 1, x_2 = 0$.

4.3.2 Wichtige Probleme mit starker Dualität

Lineare Programme sind Probleme der Form:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\min} & \boldsymbol{w}^\top \boldsymbol{x} \\ \text{s.t.} & A\boldsymbol{x} - \boldsymbol{b} \geq \boldsymbol{0} \end{array}$$

Das duale Problem eines linearen Programms ist:

$$\max_{\boldsymbol{\lambda}} \boldsymbol{b}^\top \boldsymbol{\lambda}, A^\top \boldsymbol{\lambda} = \boldsymbol{w}, \boldsymbol{\lambda} \geq 0$$

[2] "Duality Gap", Wikipedia

Quadratische Programme sind Probleme der Form:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \tfrac{1}{2}\boldsymbol{x}^\top Q\boldsymbol{x} + \boldsymbol{w}^T\boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} - \boldsymbol{b} \geq \mathbf{0} \end{aligned}$$

Das duale Problem eines quadratischen Programms ist:

$$\max_{\boldsymbol{\lambda}} -\frac{1}{2}(A^\top\boldsymbol{\lambda} - \boldsymbol{w})^\top Q^{-1}(A^\top\boldsymbol{\lambda} - \boldsymbol{w}) + \boldsymbol{b}^\top\boldsymbol{\lambda}, \boldsymbol{\lambda} \geq 0$$

Chapter 5

Lineare Programmierung

Ist eine Zielfunktion linear, so ist auch ihr Gradient konstant. Dementsprechend gilt für die Lösung eines linearen Problems immer Einer der folgenden drei Fälle:

- Es existiert keine Lösung, da die Nebenbedingungen widersprüchlich sind.
- Die Lösung liegt auf einer "Ecke" des feasible Sets. (Oder auf einer Kante, oder die Zielformel ist konstant und alle Punkte sind eine Lösung.)
- Das feasible Set ist in Richtung des negativen Gradienten nicht geschlossen, die Funktion wird beliebig klein.

Dieser Fakt wird genutzt durch das sogenannte **Simplex-Verfahren**, welches zu Beginn eine beliebige Ecke "ausprobiert", und dann immer zu der benachbarten Ecke geht, welche den Funktionswert verringert. Dabei wird der Fakt ausgenutzt, dass wir genau dann auf einer Ecke liegen, wenn m Variablen 0 sind, wobei m der Anzahl an Variablen in der Zielfunktion entspricht.

Ein Lineares Programm in **Standardform** sieht im Allgemeinen folgendermaßen aus:

$$\begin{gathered}\min_{\boldsymbol{x} \in \mathbb{R}^{m+n}} \ \boldsymbol{w}^\top (x_0, \ldots, x_m)^\top \\ s.t. A\boldsymbol{x} - \boldsymbol{b} = 0 \\ x_i \geq 0\end{gathered}$$

Wobei A die Matrix ist, welche die Gleichheitsbedingungen enthält. Sind also alle Gleichheitsbedingungen gegeben durch $\boldsymbol{c}_i^\top \boldsymbol{x} = 0$, so ist [1]

$$A = \begin{pmatrix} \boldsymbol{c}_1 \\ \vdots \\ \boldsymbol{c}_n \end{pmatrix}, \text{ also } A\boldsymbol{x} = \begin{pmatrix} c_1(\boldsymbol{x}) \\ \vdots \\ c_n(\boldsymbol{x}) \end{pmatrix}$$

Falls wir mit einem Programm mit n Ungleichheitsbedingungen starten, so sind $x_{m+1}, \ldots, x_{m+n}$ sogenannte "Schlupfvariablen", welche wir hinzufügen, um eine Ungleichheitsbedingung $c_i(x_1, \ldots, x_m) \geq b_i$ als eine Gleichheitsbedingung $c_i(x_1, \ldots, x_m) - b_i - x_j = 0$ mit $x_i, x_{m+i} \geq 0$ auszudrücken. In der Vorlesung wurde dies komplizierter geschrieben - die Schlupfvariablen wurden ξ_i genannt und das Gleichungssystem wurde kompliziert umgeformt, um auszudrücken, dass wir keine negativen x_i haben dürfen. Ich persönlich finde diese vereinfachte Schreibweise mit $\boldsymbol{x} \in \mathbb{R}^{m+n}$ jedoch deutlich sinnvoller und einleuchtender.

An dieser Stelle möchte ich noch einmal kurz zur geometrischen Intuition hinter diesem Problem kommen. Jede der ursprünglichen Ungleichheitsbedingungen spannt eine Hyperebene auf, welche den Raum $\mathbb{R}^m$ in zwei Hälften teilt - für eine Ungleichheitsbedingung $c_i(x_1, \ldots, x_m) \geq b_i$ ist diese Hyperebene durch die Menge $H_i = \{(x_0, \ldots, x_m) \mid c_i(x_1, \ldots, x_m) = b_i\}$ gegeben. Die Punkte $\geq b_i$ sind dann die Hälfte des

[1] Ich betreibe hier im Namen der Übersichtlichkeit etwas unsaubere Notation, indem ich c_i als eine Funktion interpretiere und $\boldsymbol{c}_i$ als einen Vektor, mit dem in der Funktion multipliziert wird, also $c_i(\boldsymbol{x}) = \boldsymbol{c}_i \boldsymbol{x}$

Raumes, welche in der Bedingung erlaubt ist, und die Punkte $< b_i$ sind die Hälfte des Raumes, welche nicht erlaubt ist. Durch die Bedingungen $x_i \geq 0$ an die ursprünglichen Variablen werden m zusätzliche Hyperebenen an den Koordinatenebenen aufgespannt. Nimmt man alle Bedingungen zusammen, so ist das feasible Set G eine konvexe Menge, deren Seiten durch die Hyperebenen begrenzt sind. Das ist entweder ein normaler Polyeder, oder bei unbeschränkten Problemen eine Art "offener Polyeder", welchen man sich als Polyeder vorstellen kann, bei dem die Ecken an Punkten in der Unendlichkeit liegen dürfen.

Die Punkte im Lösungsraum sind trotz Einführung der Schlupfvariablen intuitiv gegeben durch die ursprünglichen $\boldsymbol{x} \in \mathbb{R}^m$. Also befinden wir uns im m-Dimensionalen Raum, dies bedeutet ein Punkt $\boldsymbol{x}$ ist auf einer Ecke, wenn sich an ihm m Hyperebenen schneiden. Dies ist der Fall, wenn $\boldsymbol{x}$ auf mindestens m der Hyperebenen gleichzeitig liegt.

Was bedeutet das nun für unser Problem in Standardform, in dem die Ungleichheitsbedingungen durch Gleichheitsbedingungen ersetzt sind? Wenn $c_i(x_1, \ldots, x_m) \geq b_i$ die ursprüngliche Ungleichheitsbedingung ist, dann ist die äquivalente Gleichheitsbedingung $c_i(x_1, \ldots, x_m) - x_{m+1} = b_i$. Wenn also in der ursprünglichen Ungleichheitsbedingung c_i bereits Gleichheit gegeben ist, so muss in der Gleichheitsbedingung $x_{m+i} = 0$ gelten.

Wir befinden uns also genau dann auf einer Hyperebene, wenn entweder eine Schlupfvariable $x_{m+i} = 0$ ist - dann befinden wir uns auf der Ebene $H_i = \{(x_0, \ldots, x_m) \mid c_i(x_1, \ldots, x_m) = b_i\}$, die der Bedingung c_i entspricht - oder wenn eine der Ursprünglichen Variablen 0 ist - dann befinden wir uns auf der Ebene $x_i = 0$, welche der Bedingung $x_i \geq 0$ entspricht. Also sind wir immer dann auf einer Ecke, wenn mindestens m der Variablen 0 sind.

Dies ist genau der Fakt, welchen sich das Simplex-Verfahren zunutze macht. Wir starten mit einem beliebigen Punkt $\boldsymbol{x} \in \mathbb{R}^{m+n}$. An jedem Schritt nehmen wir dann eine Variable x_i, welche momentan 0 ist - es bietet sich an, die zu nehmen, welche in der Zielfunktion den negativen Koeffizienten mit dem höchsten Betrag hat, da diese zur Stärksten Verbesserung führt - und erhöhen diese. Dafür müssen wir dann eine andere Variable, welche mit x_i in Relation steht, auf 0 setzen. An jedem Schritt stellen wir unsere Gleichheitsbedingungen so um, dass jede Gleichheitsbedingung auf der linken Seite genau eine Variable enthält, welche momentan nicht 0 ist, und setzen ebenso in der Zielfunktion immer die Variablen ein, welche gerade 0 sind.

Indem wir die Zielfunktion umstellen, erhalten wir eine simple Bedingung, an der wir erkennen können, ob wir das Optimum erreicht haben: Wir haben das Optimum erreicht, wenn alle Koeffizienten in der Zielfunktion positiv sind, da in diesem Fall jede Veränderung der Variablen zu einer Verschlechterung des Ergebnisses führt.

Ich persönlich fand die in der Vorlesung gegebene Erklärung des Simplex-Verfahrens viel zu kompliziert und insgesamt sehr verwirrend. Dementsprechend möchte ich an dieser Stelle explizit das Video "The Art of Linear Programming", von Tom S, erwähnen, durch welches das Simplex-Verfahren für mich endlich übersichtlich und intuitiv verständlich wurde.

Chapter 6

Nichtlineare Programmierung