

# Optimierung

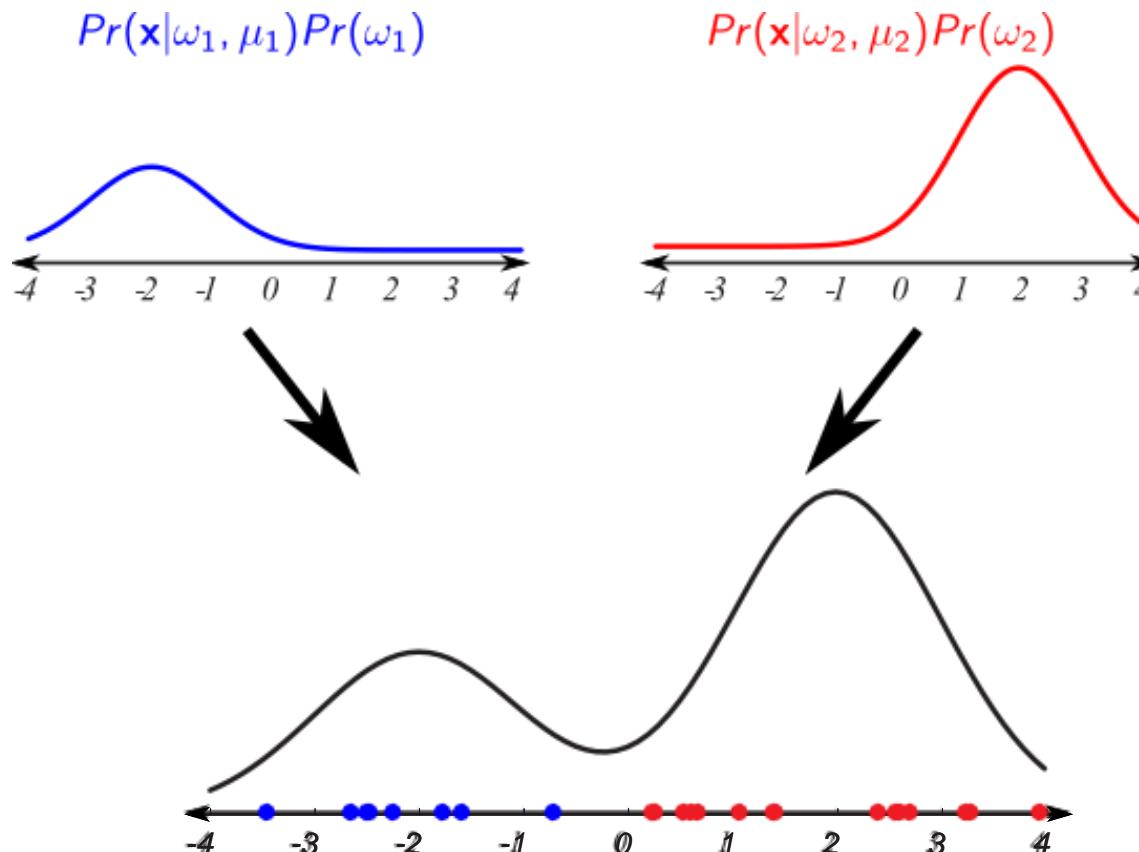
---

Vorlesung 1  
Einführung und Überblick

- Informatik befasst sich viel mit **Algorithmen**
- Noch vor dem Algorithmus steht jedoch das zu lösende **Problem**, welches man idealerweise formal sauber in Form eines **Modells** ausdrückt.
- Überspringt man die Modellierung, führt dies zu “Hacks” mit zahlreichen negativen Konsequenzen:
  - Zusammenhänge zwischen ähnlichen Problemen gehen verloren
  - Kombinationen und Erweiterungen werden sehr schnell sehr kompliziert
  - Theoretische Analyse und Garantien oft nicht möglich
- Konsequenter Zugang:
  - Formulierung des zu lösenden Problems  
**(Modell, Zielfunktion)**
  - Anwendung von passenden Algorithmen entsprechend der Problemklasse  
**(Optimierungsmethode)**

# Beispiel Hauspreise

- Informatiker im Bankengewerbe, Auftrag: Erwarteter Gewinn Hausverkäufe
- Lösung 1: Einfach Erwartungswert bilden.
- Lösung 2: Sich die Verteilung ansehen -> Nachbarschaften



- Optimierungsproblem: Optimiere Parameter der beiden Verteilungen

- Basteln und Ausprobieren ist die kreative Seite der Wissenschaft
- Für komplexe Aufgaben reicht Kreativität alleine nicht aus
- Mache die Aufgabe erst so einfach wie möglich
  - das Wesentliche erkennen
  - Zusammenhänge zu bekannten Lösungen sehen
  - bekannte Analysen und Eigenschaften nutzen
- Übrig bleibt (hoffentlich) eine viel einfachere Aufgabe, an der dann gebastelt werden darf
- Theorie und Praxis sind keine Gegensätze!



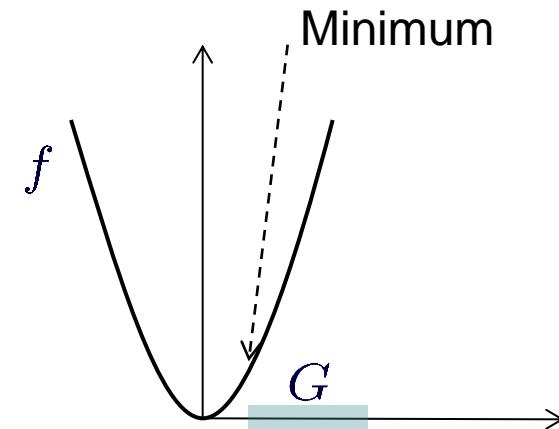
# Wie sieht ein Optimierungsproblem aus?

- Ein Optimierungsproblem besteht aus einer **zulässigen Menge**  $G$  und einer **Zielfunktion**  $f : G \rightarrow \mathbb{R}$

- Beispiel:

$$f(x) = x^2$$

$$G = \{x \in \mathbb{R} | 2 \leq x \leq 5\}$$



- **Minimum:**  $\min_x f(x) = 4$
- **Minimierer:**  $\operatorname{argmin}_x f(x) = 2$
- Wir haben also ein Modell in Form einer Zielfunktion und möchten aus einer großen Anzahl möglicher Lösungen (zulässige Menge) die beste finden (Minimierer).

- Historisch stark verankert in “**Operations Research**”: Optimierung von Produktionskosten unter diversen Nebenbedingungen
- Fast alle modernen Verfahren aus dem **maschinellen Lernen** basieren auf Optimierung
- **Pfadplanung** basiert auf effizienten Optimierungsverfahren (z.B. kürzeste Pfade in Navigationssystemen)
- **Hardwaredesign** und **paralleles Rechnen** basiert auf der Optimierung von Layouts und Netzwerken
- **Computer Vision** basiert größtenteils auf Optimierung
- Große Teile der **Bioinformatik** nutzen Optimierungsverfahren

- Gegeben: Samples  $x_n$  zweier Klassen:  $t_n \in \{-1, 1\}$
- Ziel: Finde lineare Entscheidungsgrenze  $y = \mathbf{w}^\top \mathbf{x} + b$  um ein neues Beispiel  $\mathbf{x}$  zu klassifizieren

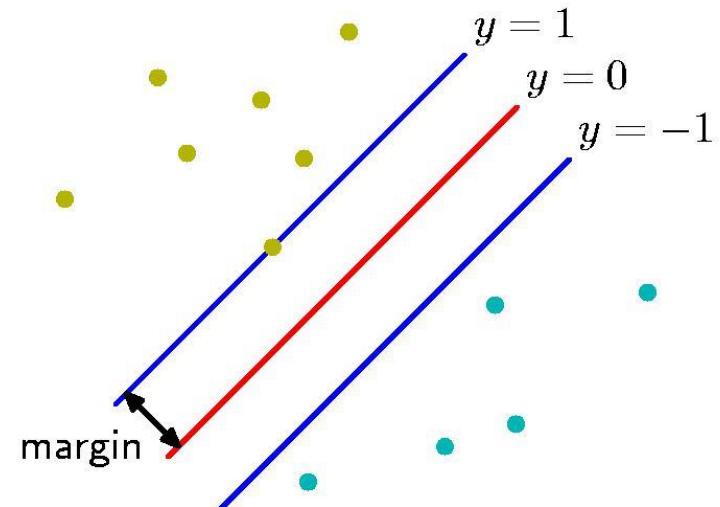
1. Formuliere das Problem als Kostenfunktion

$$\operatorname{argmin}_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad (\text{kleine Gewichte sind günstiger})$$

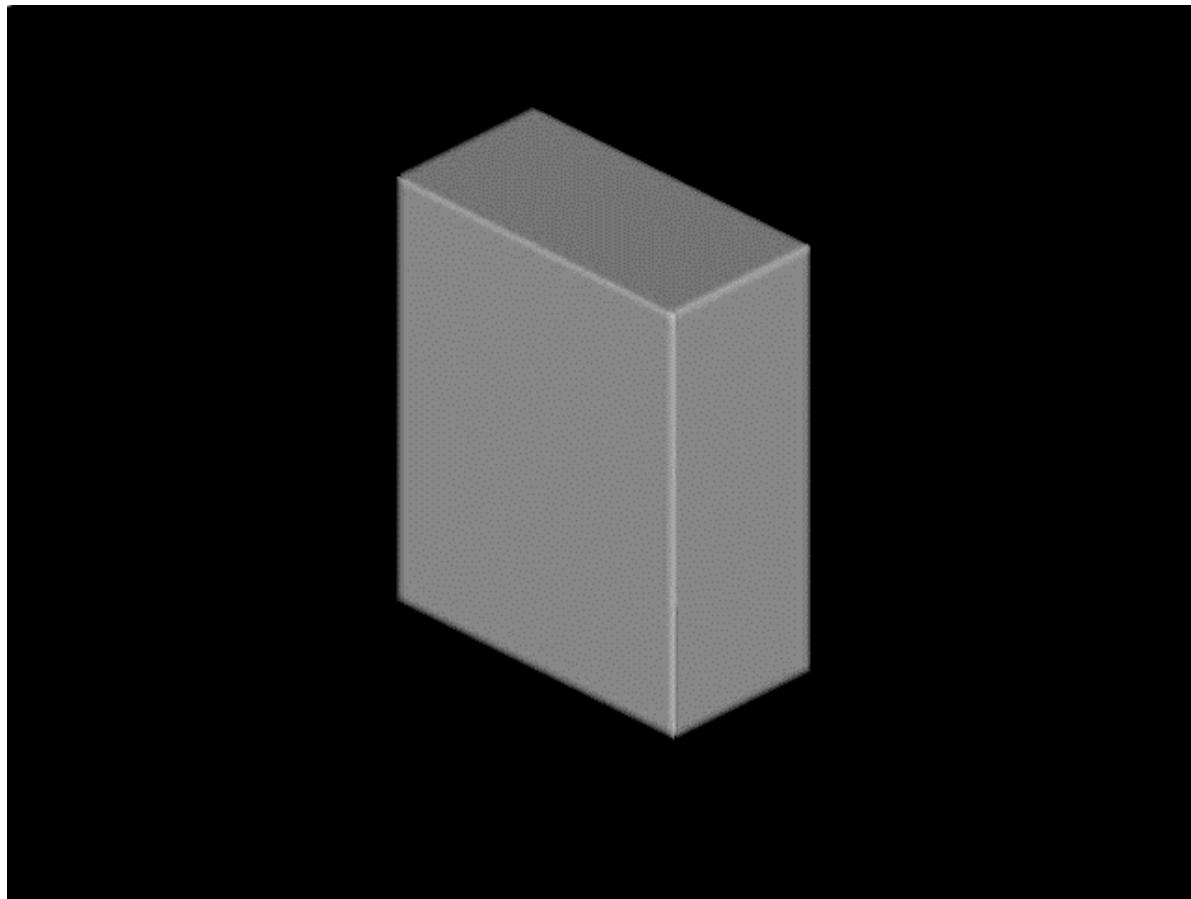
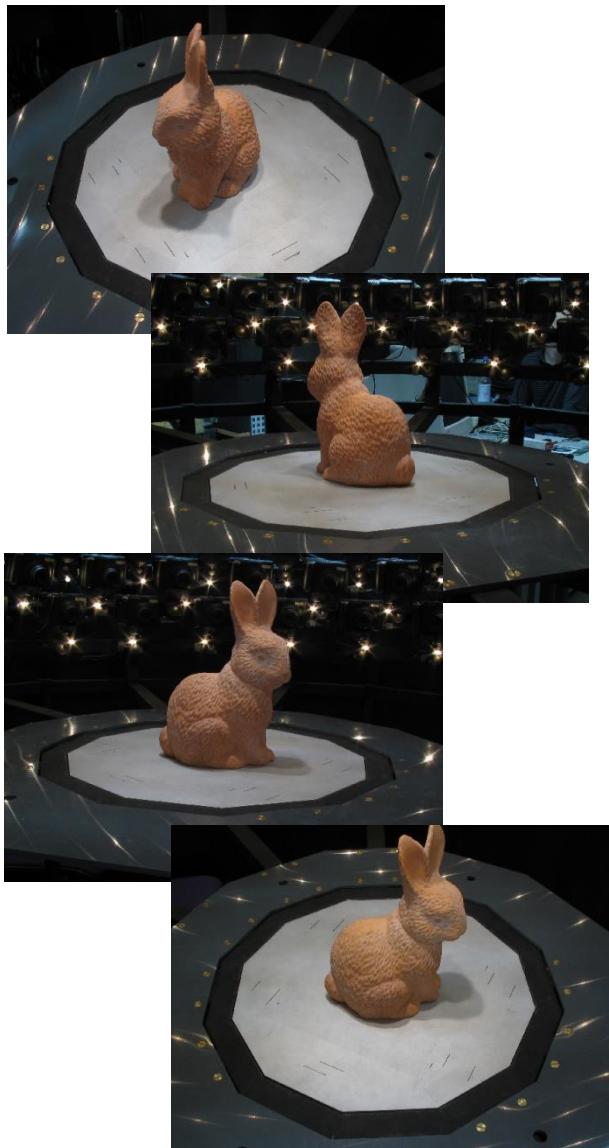
mit Nebenbedingungen

$$t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1 \quad (\text{alle Samples müssen richtig klassifiziert werden})$$

2. Dieses Optimierungsproblem ist ein **quadratisches Programm**  
→ kann mit Standardverfahren garantiert optimal gelöst werden



# Beispiel aus Computer Vision: 3D Rekonstruktion



Finde die Oberfläche, die am besten alle Eingabebilder erklärt

$$\operatorname{argmin}_{u(\mathbf{X}) \in [0,1]} \int R u + \nu \rho |\nabla u| d\mathbf{X}$$

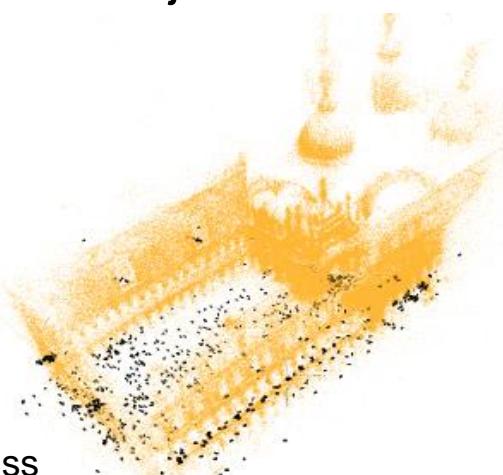
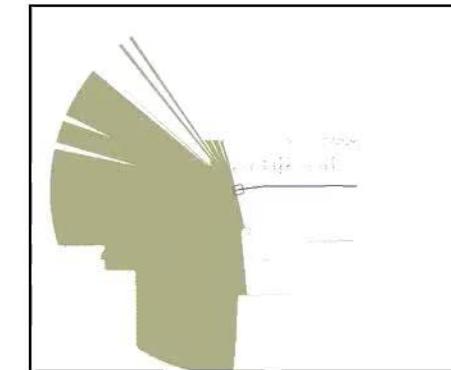
- Ziel: Finde eine Kontur  $C$ , die das Bild In Vordergrund  $\Omega_1$  und Hintergrund  $\Omega_2$  aufteilt
- Beide Regionen werden durch die mittlere Helligkeit  $\mu_1, \mu_2$  modelliert
- Kürzere Konturen sind besser
- Kostenfunktion:

$$E(C) = \int_{\Omega_1} (I - \mu_1)^2 dx + \int_{\Omega_2} (I - \mu_2)^2 dx + \nu |C|$$

Kann mittels **Gradientenabstieg** minimiert werden

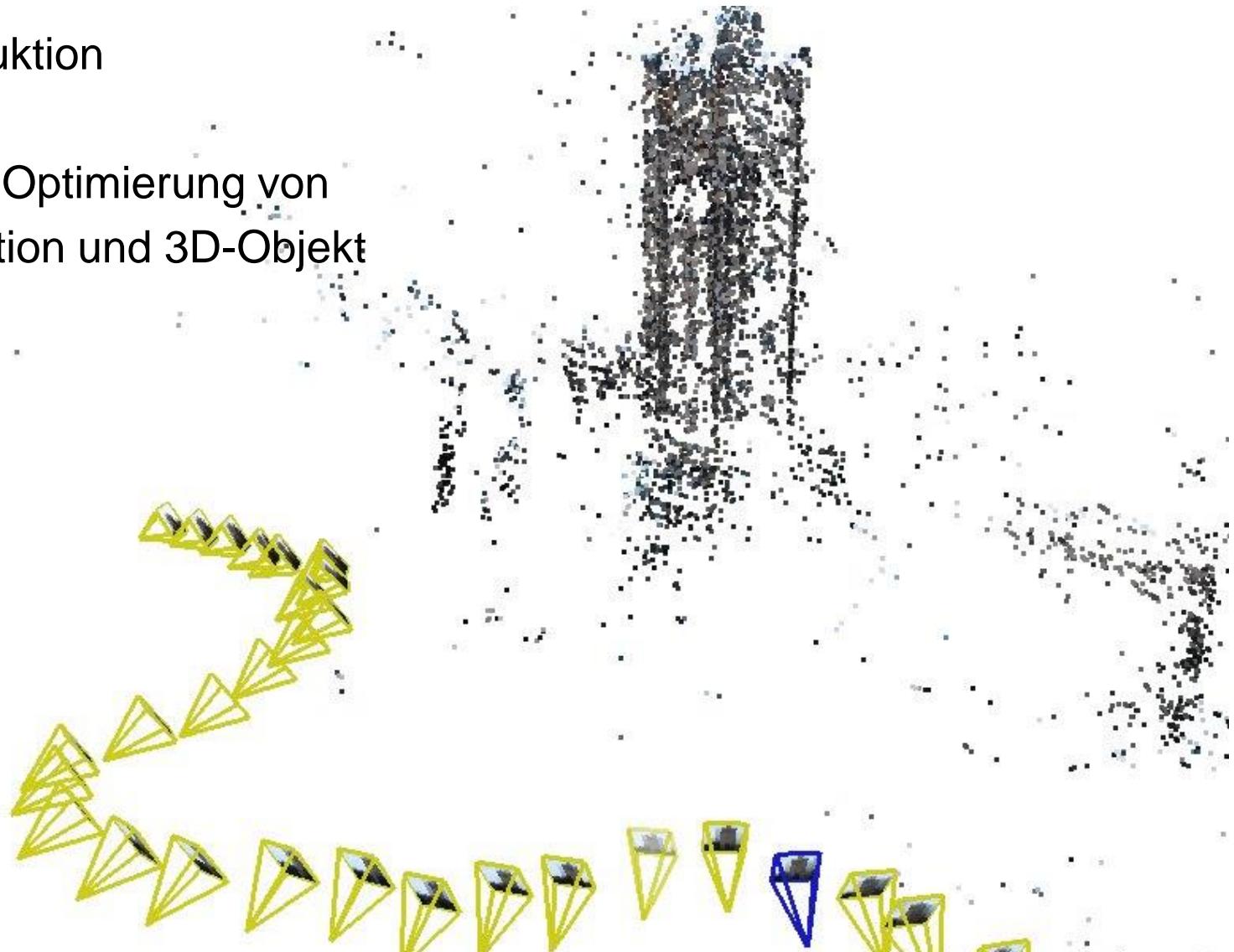


- Ziel: Erstellen einer Umgebungskarte und gleichzeitige Lokalisierung in der Karte (SLAM = simultaneous localization and mapping)
- Sensormessungen sind jeweils relativ zur Roboterposition  
→ Berücksichtigung aller Messungen in einem Optimierungsproblem
- Lösung mit  
**Gauss-Newton-Verfahren**
- Äquivalentes Problem (im Sinne der Optimierung) in Computer Vision: Bundle Adjustment



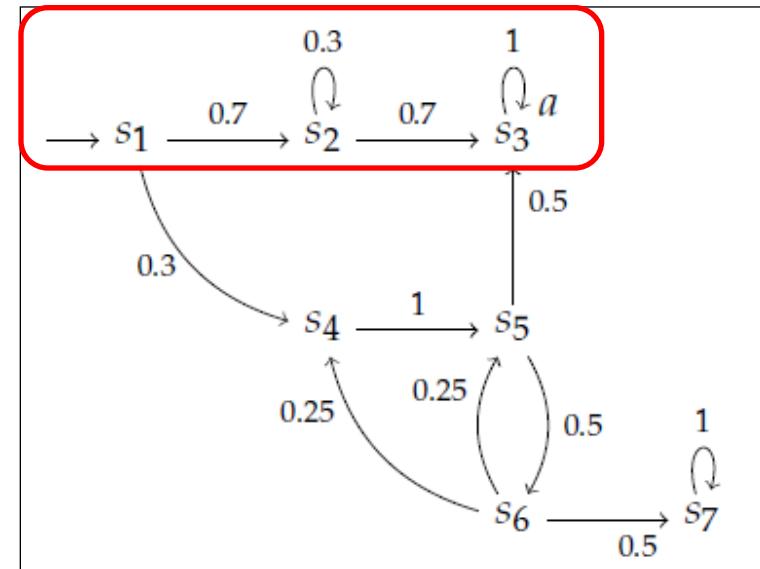
Autor: Cyrill Stachniss

- 3D-Rekonstruktion
- Gleichzeitige Optimierung von Kamera-Position und 3D-Objekt



C. Olsson, A. Eriksson, Ri. Hartley, CVPR 2010

- Für sicherheitsrelevante Systeme muss gezeigt werden, dass sie bestimmte Spezifikationen (mit einer bestimmten Wahrscheinlichkeit) erfüllen.
- Exploration sehr großer Zustandsräume
- Oft ist nur ein bestimmtes Teilsystem für die Abweichung von der Spezifikation verantwortlich
- Das kleinste relevante Teilsystem lässt sich mithilfe eines **linearen Programms** bestimmen.



Minimales kritisches Teilsystem

$$\text{minimize} \quad -\frac{1}{2} p_{s_{\text{init}}} + \sum_{s \in S} x_s$$

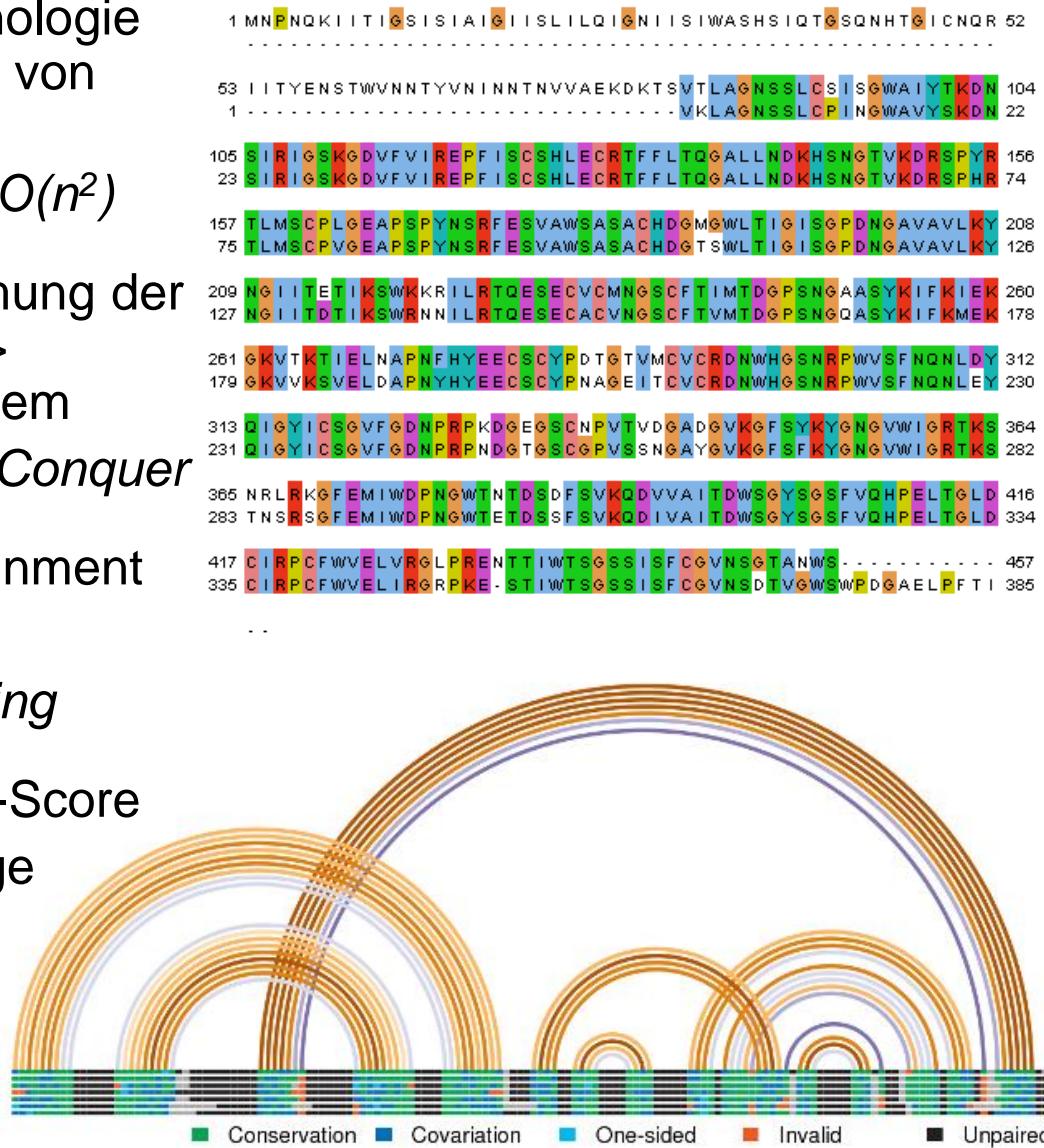
such that

$$\begin{aligned} \forall s \in T_a : \quad & p_s = x_s \\ \forall s \in S \setminus T_a : \quad & p_s \leq x_s \\ \forall s \in S \setminus T_a : \quad & p_s \leq \sum_{s' \in \text{succ}(s)} P(s, s') \cdot p_{s'} \\ & p_{s_{\text{init}}} > \lambda . \end{aligned}$$

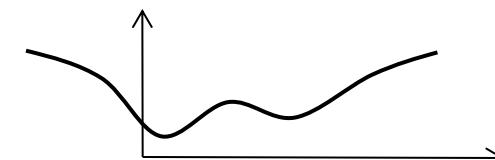
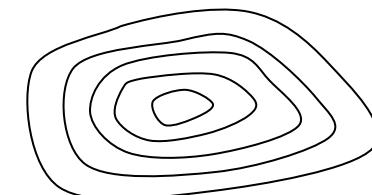
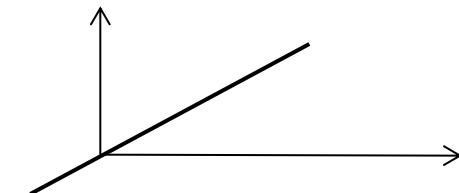
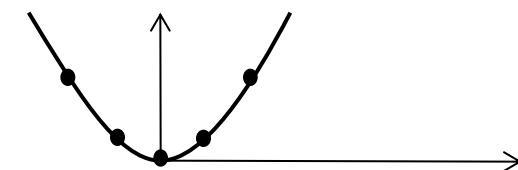
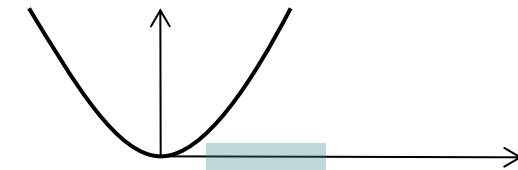
Mixed integer linear program

Quelle: Wimmer et al. 2012

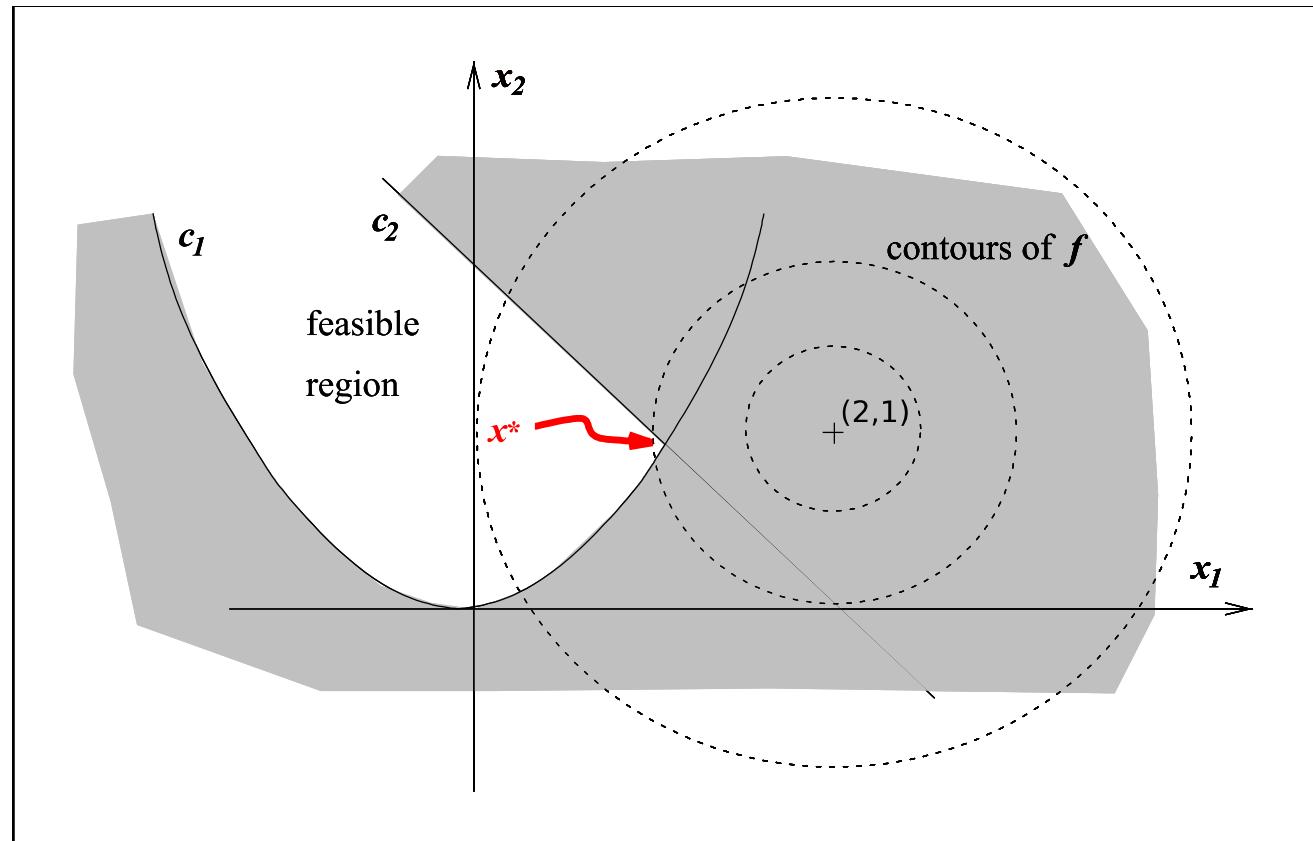
- Sequenzalignment Basistechnologie zur Bestimmung der Evolution von Genen  
*Dynamische Programmieren  $O(n^2)$*
- Anwendung auch zur Bestimmung der Ähnlichkeit ganzer Genome -> Hauptspeicher wird zum Problem  
*Linear-Space mit Divide-and-Conquer*
- Erweiterung: Strukturelles Alignment ist NP-schwer  
*Integer Linear Programming*
- Lokales Alignment: Alignment-Score wird durch die Alignment-Länge geteilt  
*Fractional Programming*



- Probleme mit Nebenbedingungen vs. Probleme ohne Nebenbedingungen
- Optimierung mit kontinuierlichen Variablen vs. diskreten Variablen
- Lineare vs. nichtlineare Funktionen
- Eindimensionale vs. mehrdimensionale Funktionen
- Konvexe vs. nicht-konvexe Funktionen und Mengen (mehr dazu gleich)

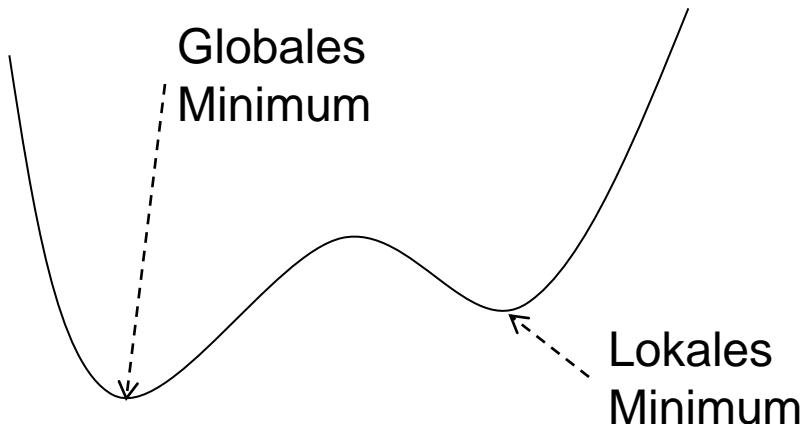


- Problem:  $\min(x_1 - 2)^2 + (x_2 - 1)^2$  subject to
 
$$\begin{aligned}x_1 + x_2 &\leq 2 \\x_1^2 - x_2 &\leq 0\end{aligned}$$
- Lösungsbereich:



- Bem: allg. Kreisgleichung lautet  $(x_1 - M_1)^2 + (x_2 - M_2)^2 = r^2$

- Die verschiedenen Problemklassen (und ihre Kombinationen) sind unterschiedlich “schwierig”.
- Bei diskreten Optimierungsproblemen:
  - Lässt sich das Optimum **garantiert in polynomieller Zeit** finden?
  - Falls ja, wie groß ist die **Komplexität** des Optimierungsverfahrens?
  - Gibt es Approximationen mit Garantien bzgl. der maximalen Abweichung (obere/untere Schranken)?
- Bei kontinuierlichen Optimierungsproblemen:
  - Gibt es Verfahren, die in allen Fällen zu einer Lösung **konvergieren**?
  - Konvergiert das Verfahren garantiert zum **globalen Optimum**?
  - Wie schnell konvergiert das Verfahren (**Konvergenzordnung**)?



- Für ein **lokales Minimum**  $f(x^L)$  muss gelten

$$f(x^L) \leq f(x) \quad \forall x \in U(x^L)$$

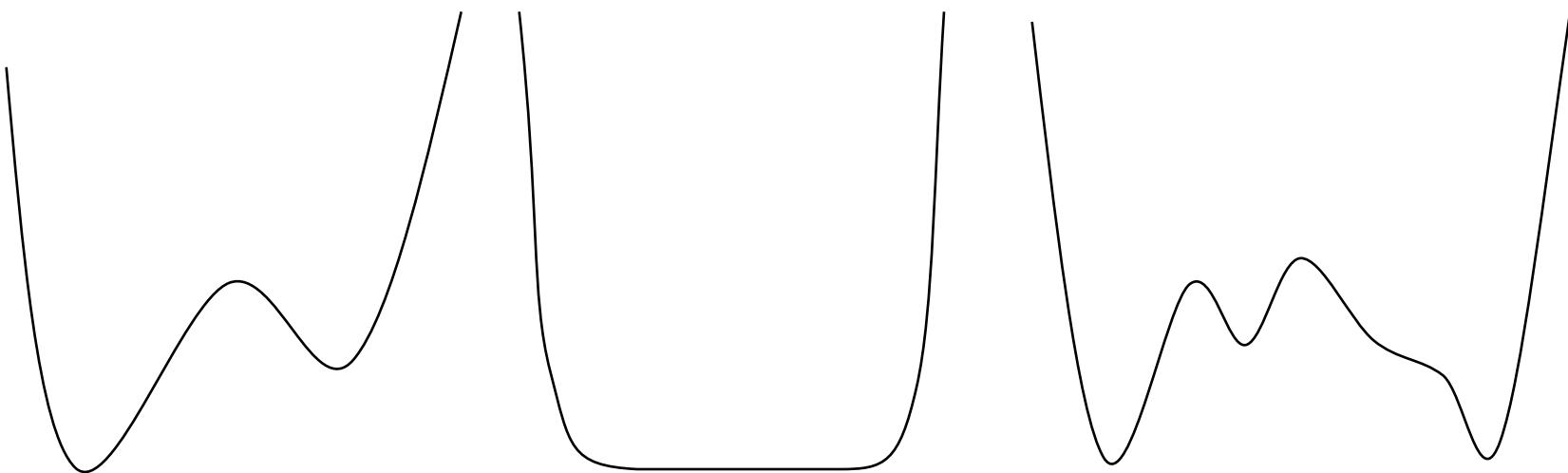
Dabei ist  $U(x^L)$  eine Normkugel mit einem hinreichend kleinen Radius  $\epsilon > 0$ :

$$U(x^L) = \{x \in \mathbb{R}^n \mid \|x - x^L\| < \epsilon\}$$

- Für ein **globales Minimum**  $f(x^*)$  muss gelten

$$f(x^*) \leq f(x) \quad \forall x \in \mathbb{R}^n$$

# Im allgemeinen kann es mehrere globale Minima geben

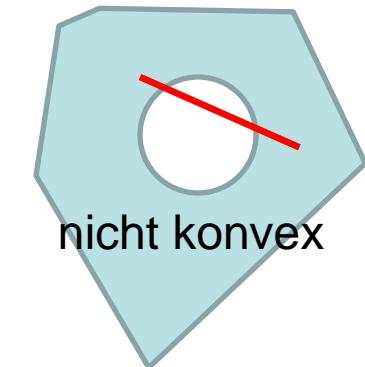
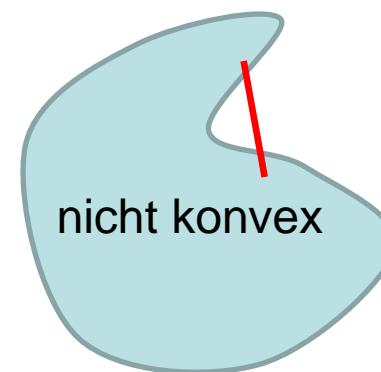
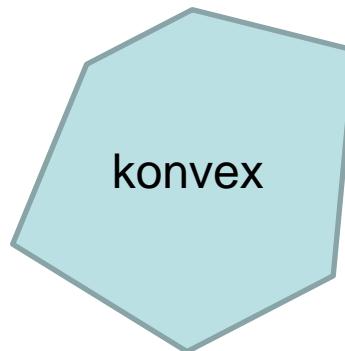
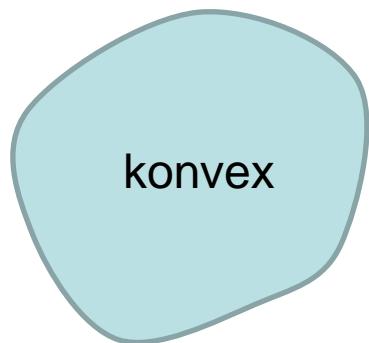


Eindeutiges  
globales  
Optimum

Zusammenhängende  
Region globaler  
Optima

Mehrere globale  
Optima

Kann man feststellen, ob eine Optimierungsaufgabe "gutmütig" ist, also ein eindeutiges globales Minimum und keine weiteren lokalen Minima hat?



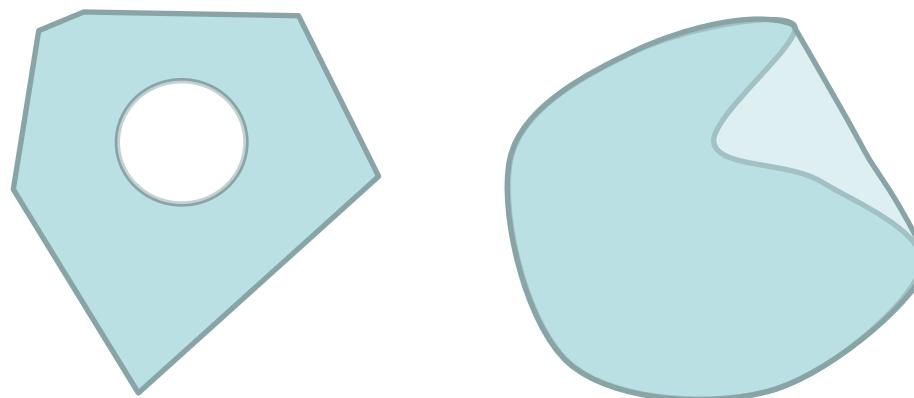
- Eine Menge  $G \subset \mathbb{R}^n$  ist konvex, wenn für beliebige Punkte  $x, y \in G$  auch die Verbindungsgerade

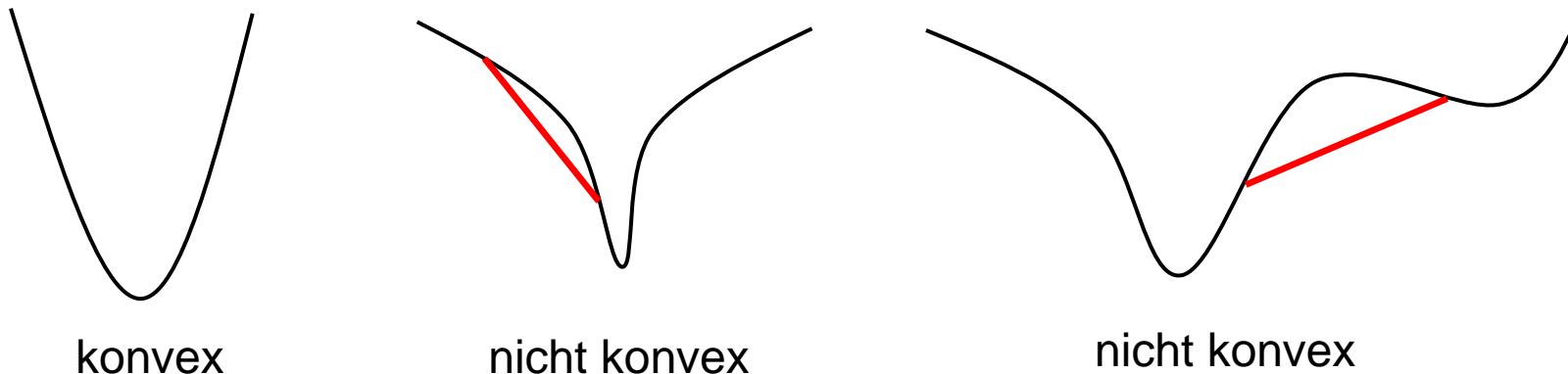
$$[x, y] := \{z := (1 - \lambda)x + \lambda y \mid \lambda \in [0, 1]\}$$

in  $G$  enthalten ist:

$$x, y \in G \Rightarrow [x, y] \subset G$$

- Die **konvexe Hülle** einer Menge  $G$  ist die kleinste konvexe Menge, die  $G$  vollständig enthält.





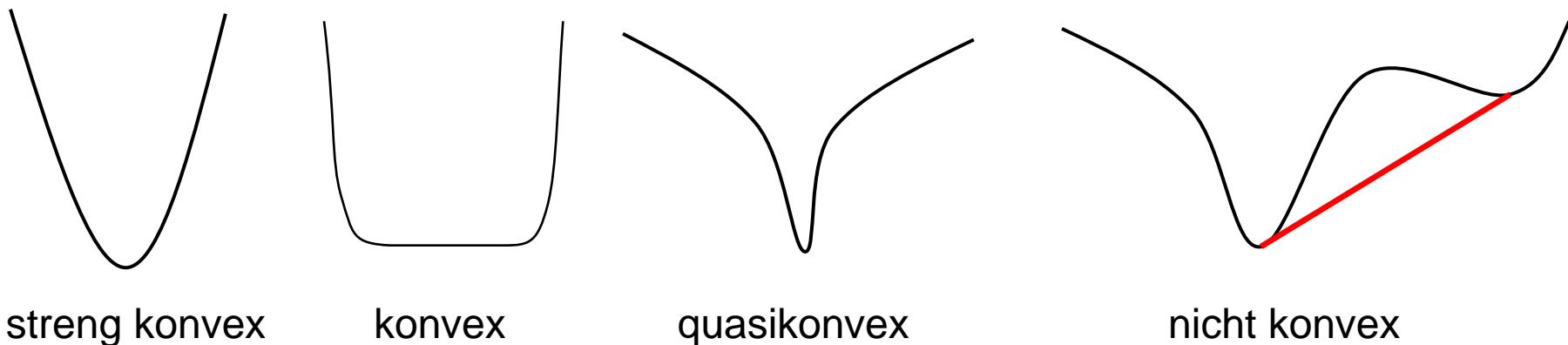
- Eine über einer konvexen Menge  $G$  definierte Funktion  $f : G \rightarrow \mathbb{R}$  heißt **konvex**, falls

$$x, y \in G; x \neq y \Rightarrow f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y) \quad \forall \lambda \in [0, 1]$$

- Sie heißt **streng konvex**, falls

$$x, y \in G; x \neq y \Rightarrow f((1-\lambda)x + \lambda y) < (1-\lambda)f(x) + \lambda f(y) \quad \forall \lambda \in [0, 1]$$

- Eine streng konvexe Funktion besitzt ein eindeutiges globales Minimum und keine lokalen Minima.
- Eine konvexe Funktion kann mehrere globale Minima besitzen, jedoch keine zusätzlichen lokalen Minima.
- Eine nicht-konvexe Funktion, die keine weiteren lokalen Minima besitzt, wird als **quasikonvex** bezeichnet.



- Die Verbindungsline zwischen globalem und lokalem Minimum muss unterhalb des Graphen liegen!

- Vorlesung 1: Einführung
  - Vorlesung 2: Gradientenverfahren
  - Vorlesung 3: Newton- und Quasi-Newton-Verfahren
  - Vorlesung 4: Optimierung mit Nebenbedingungen
  - Vorlesung 5: Lineare Programme
  - Vorlesung 6: Quadratische Programme
  - Vorlesung 7: Nichtlineare Programme
  - Vorlesung 8: Kombinatorische Optimierung
- 
- Übung 1: Gradientenverfahren
  - Übung 2: Quasi-Newton-Verfahren
  - Übung 3: Nebenbedingungen
  - Übung 4: Lineare Programme
  - Übung 5: Projektionsmethoden

- Übungen
  - Mehrheitlich praktische Aufgaben in Python
  - Ziel: Gefühl für einige wichtige Verfahren
  - Ca. jede dritte Woche (genaue Information auf der Internetseite der Vorlesung)
- Prüfung
  - Schriftliche Prüfung
  - Unbedingt rechtzeitig zur Prüfung anmelden
  - Keine Zulassungsbeschränkungen
- Folien und weiteres Material im ILIAS unter:  
<https://ilias.uni-freiburg.de>  
Kurs "Optimierung"  
(Passwort: reinda)  
direkter Link in Kursseite auf  
<http://www.bioinf.uni-freiburg.de/Lehre>

- Sie sind erwachsene Menschen und Studenten einer **Universität**
- Fachwissen ist nicht das primäre Lernziel (vergleiche Fachhochschulen)
- Lernziele an der Universität:
  - Fähigkeit zu abstrahieren
  - Lernen mit Schwierigkeiten positiv umzugehen
  - Verantwortung übernehmen (zunächst für sich selbst)
  - Durchzuhalten auch wenn es mal keinen Spaß macht
- Nutzen Sie die Freiheit richtig und nehmen Sie diese Ratschläge mit:
  - Nehmen Sie die Übungen ernst
  - Arbeiten Sie den Stoff nach der Vorlesung noch einmal kurz durch
  - Füllen Sie Verständnislücken möglichst zügig (eigenständig, mit Kommilitonen, durch Fragen in der Vorlesung oder Übung)
  - Betreiben Sie bewusst Zeitmanagement (just in time is often just too late)

- J. Nocedal, S. J. Wright: Numerical Optimization, Springer, 2006.
- H. Jongen, K. Meer, E. Triesch: Optimization Theory, Kluwer, 2004.
- C. Großmann, J. Terno: Numerik der Optimierung, Teubner, 1997.
- H. Hamacher, K. Klamroth: Lineare Optimierung und Netzwerkoptimierung, Vieweg, 2006.
- M. S. Bazaraa, H. D. Sherali, C. M. Shetty: Nonlinear Programming, Wiley, 2006.
- R. Wimmer, B. Becker, N. Jansen, E. Abraham, J.-P. Katoen: Minimal critical subsystems for discrete-time Markov models, Proc. TACAS, Springer LNCS, pp. 299–314, 2012.

# Optimierung

---

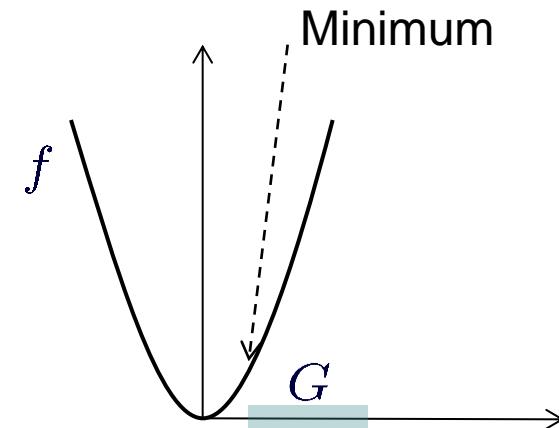
## Vorlesung 2 Gradientenverfahren

- Ein Optimierungsproblem besteht aus einer **zulässigen Menge**  $G$  und einer **Zielfunktion**  $f : G \rightarrow \mathbb{R}$

- Beispiel:

$$f(x) = x^2$$

$$G = \{x \in \mathbb{R} | 2 \leq x \leq 5\}$$



- **Minimum:**  $\min_x f(x) = 4$
- **Minimierer:**  $\operatorname{argmin}_x f(x) = 2$
- Die nächsten Vorlesungen:  $G = \mathbb{R}^n$ 
  - Kontinuierliche Variablen (beliebiger Dimensionalität)
  - Keine Nebenbedingungen

- Nachfolgend gehen wir davon aus, dass die Funktion  $f$  mindestens einmal differenzierbar ist:  $f \in \mathcal{C}^1$

- Gradient von  $f$  :

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

- **Notwendige Bedingung** für ein Minimum von  $f$ :

$$\nabla f(x) = \mathbf{0}$$

- Ist  $f$  konvex, so ist dies auch eine **hinreichende Bedingung**.
- Allgemein stellt die Bedingung  $\nabla f(x) = \mathbf{0}$  ein **nichtlineares Gleichungssystem** dar, das numerisch gelöst werden muss.

- Iteratives Verfahren

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \tau^k \mathbf{d}^k$$

mit einem Startpunkt  $\mathbf{x}^0$ , einer Änderungsrichtung  $\mathbf{d}^k$  und einer Schrittweite  $\tau^k$  (Bem: Vektoren werden fett gedruckt)

- Beim Gradientenverfahren entspricht die Änderungsrichtung dem negativen Gradienten der Zielfunktion  $f$  an der aktuellen Stelle  $\mathbf{x}^k$

$$\mathbf{d}^k := -\nabla f(\mathbf{x}^k)$$

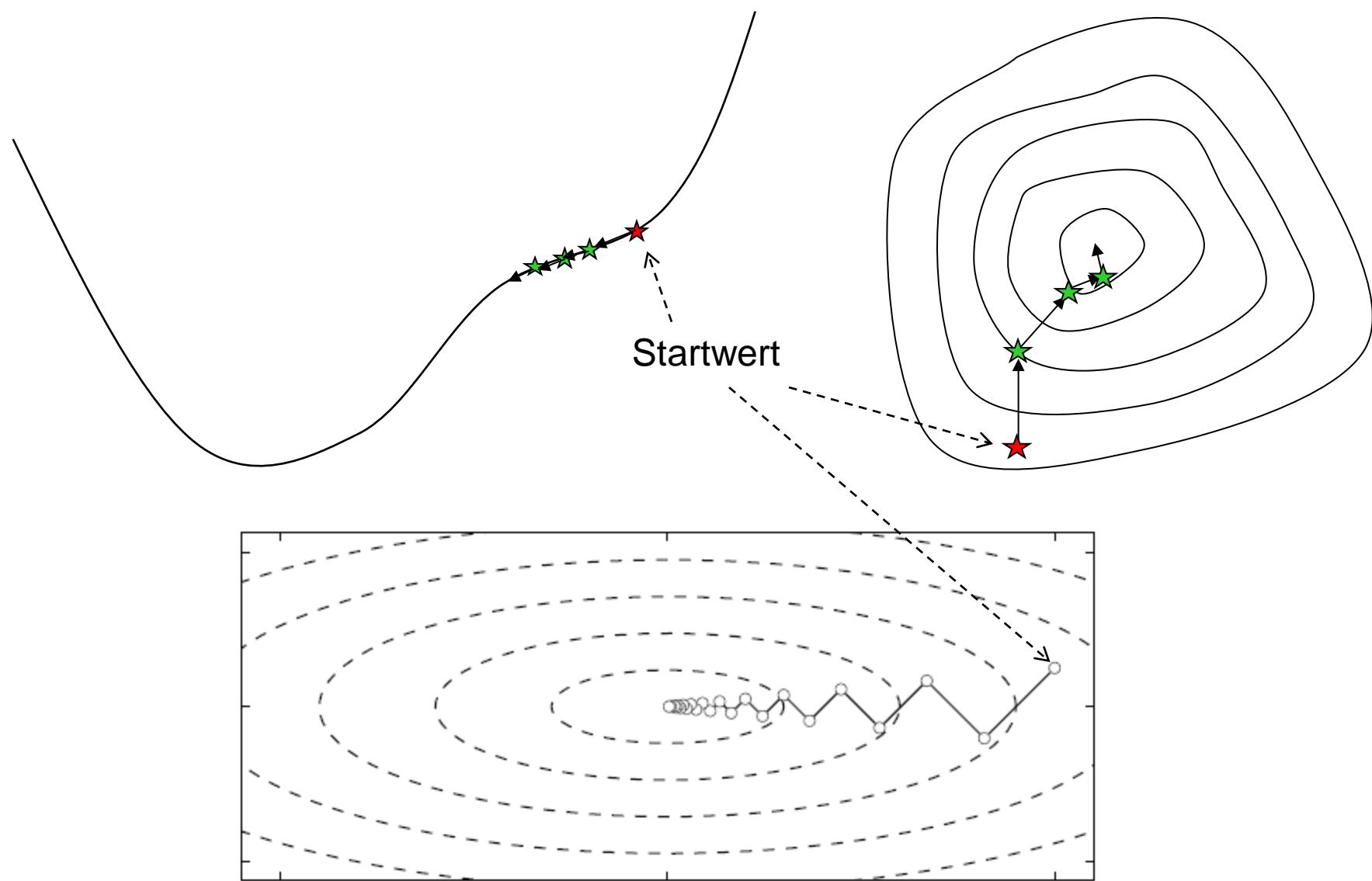
Optimalität ergibt sich aus der Taylor-Entwicklung.

- Die Schrittweite  $\tau^k$  wird optimal bestimmt, so dass

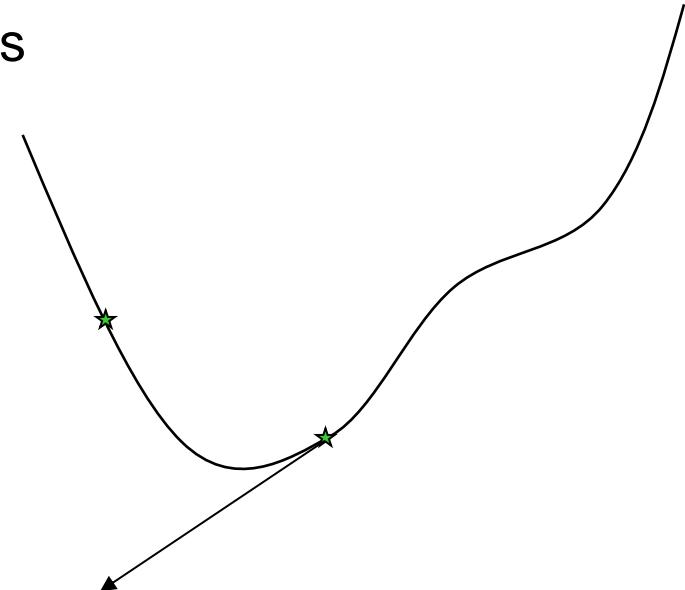
$$f(\mathbf{x}^k + \tau^k \mathbf{d}^k) \leq f(\mathbf{x}^k + \alpha \mathbf{d}^k) \quad \forall \alpha \geq 0$$

Mehr zur Schrittweitenbestimmung später...

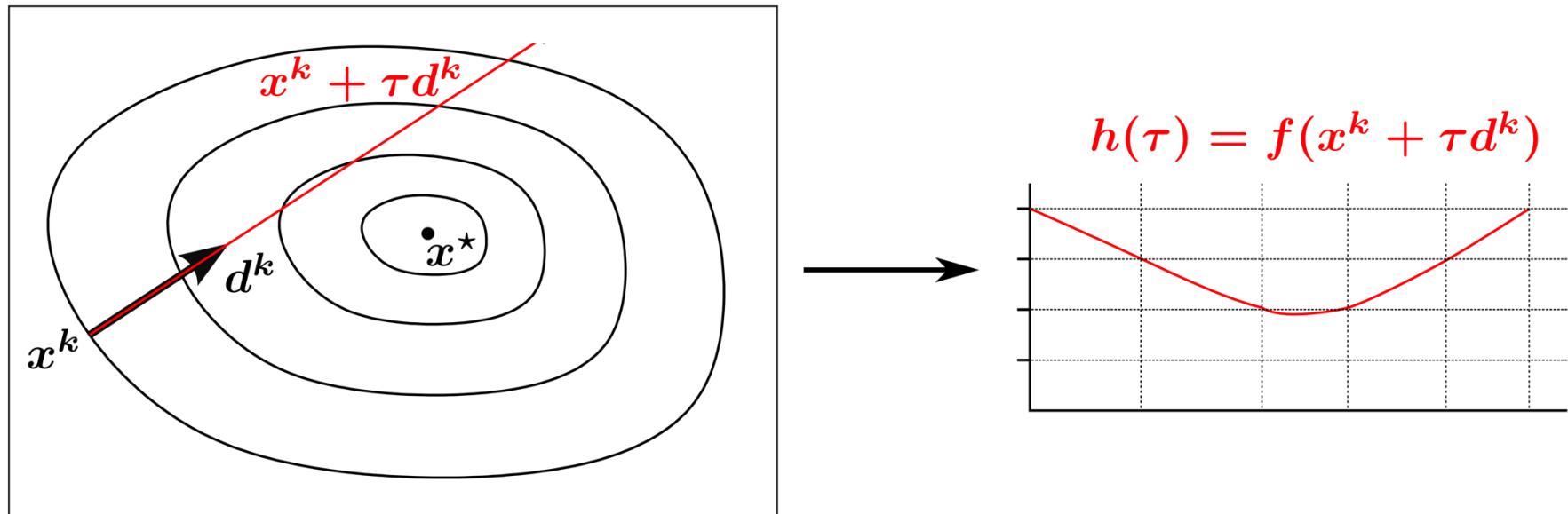




- Die Wahl der Schrittweite ist entscheidend für **Konvergenz** und Effizienz des Verfahrens
- Bei zu großen Schrittweiten schießt das Verfahren über das Ziel hinaus (u.U. keine **Konvergenz**)
- Bei kleinen Schrittweiten benötigt das Verfahren viele Iterationen zur Lösung  
→ hoher Rechenaufwand
- Richtung bereits durch  $d^k$  vorgegeben  
→ eindimensionales Optimierungsproblem über  $\tau^k$   
→ **Line search**
- In einigen Fällen ist eine feste Schrittweite  $\tau$ , die Konvergenz garantiert, effizienter als die Bestimmung einer optimalen Schrittweite.



- Zur Schrittweitenbestimmung: Situation im Schritt k.



- Im folgenden: Betrachte Richtungsableitung im Punkt  $x^k$  in Richtung  $d^k$ :

$$\lim_{\epsilon \rightarrow 0} \frac{f(x^k + \epsilon d^k) - f(x^k)}{\epsilon} = \nabla f(x^k)^T d^k$$

- Im folgenden: Zusammenhang Richtungsableitung und  $h(\tau)$

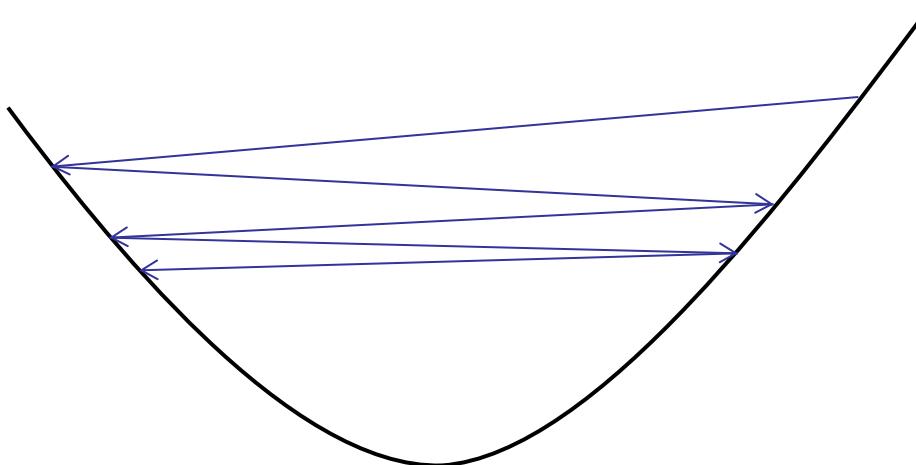
- Optimierungsaufgabe mit einer eindimensionalen Funktion  $h : \mathbb{R}_+ \rightarrow \mathbb{R}$   
 $\tau^* = \operatorname{argmin}_\tau h(\tau)$
- Falls  $h$  differenzierbar ist, ergibt sich die notwendige Bedingung  
 $h'(\tau) = 0$
- Normalerweise nichtlinear und dann nur mit großem Aufwand exakt lösbar
- Exakte Lösung oft nicht von Vorteil, da aktuelle Abstiegsrichtung auch bei optimaler Schrittweite ohnehin nicht direkt zum Ziel führt
  - Approximative Schrittweitensuche unter Einhaltung gewisser Qualitätsbedingungen

Diese Bedingungen garantieren Konvergenz und Effizienz der verschiedenen Verfahren

- Einfache Reduktion des Funktionswerts

$$f(x^k + \tau d^k) - f(x^k) < 0$$

ist nicht ausreichend.



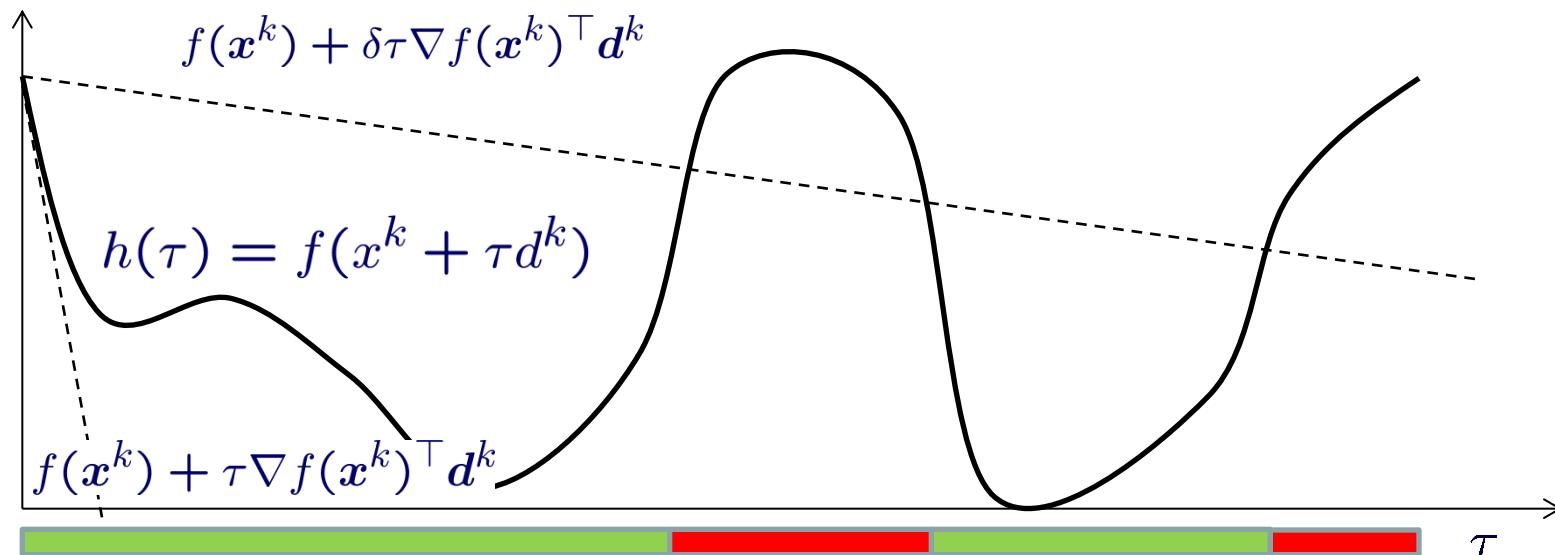
- Obwohl jede Iteration den Funktionswert reduziert, wird die Reduktion möglicherweise immer kleiner und erreicht das Optimum nie

## 1. Hinreichender Abstieg (Armijo-Bedingung):

$$f(x^k + \tau d^k) \leq f(x^k) + \delta \tau \nabla f(x^k)^\top d^k \quad \delta \in (0, 1)$$

Stellt sicher, dass die Zielfunktion  $f$  durch die Schrittweite  $\tau$  in Abhängigkeit der Steilheit des Gradienten "hinreichend" reduziert wird.  
Üblicherweise  $\delta = 10^{-4}$

Je steiler der Gradient und je größer der Schritt umso mehr muss sich der Funktionswert reduzieren

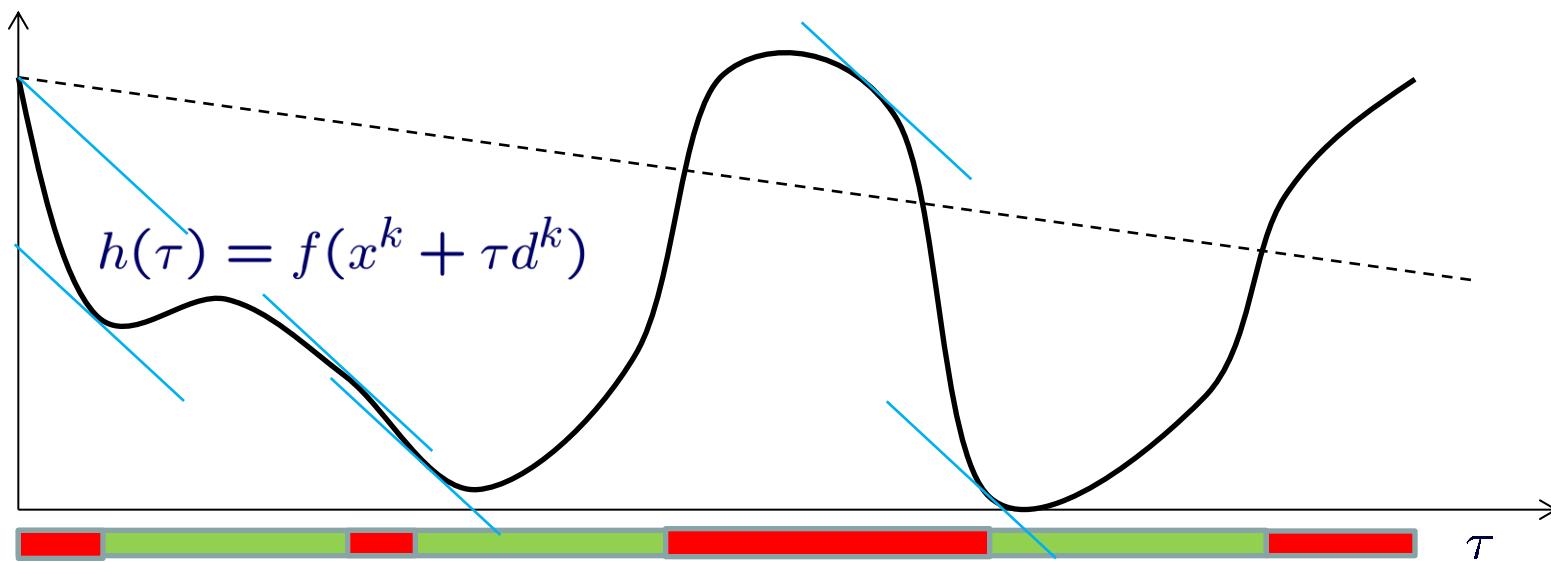


## 2. Krümmungsbedingung (schwache Wolfe Bedingung)

$$\nabla f(x^k + \tau d^k)^\top d^k \geq \eta \nabla f(x^k)^\top d^k \quad \eta \in (\delta, 1)$$

Gradient an der Zielposition ist weniger steil als an der Startposition oder sogar positiv.

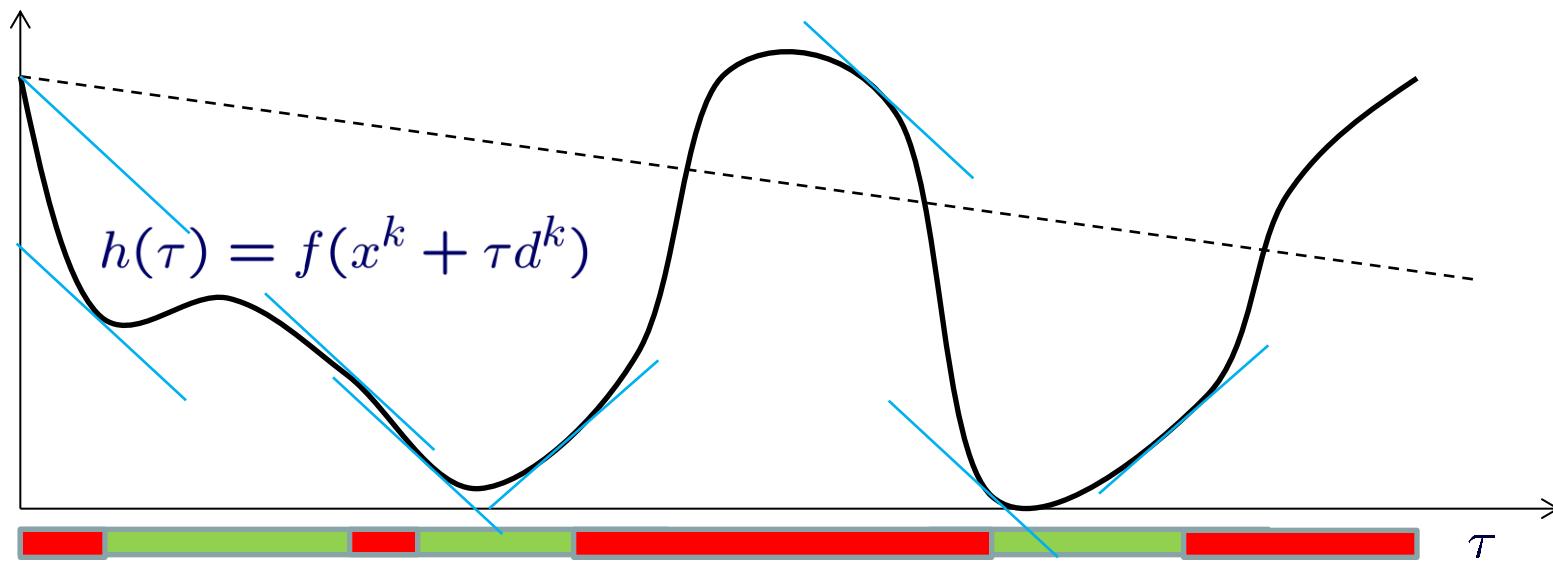
Verhindert ineffizient kleine Schritte in Bereiche hinein, in denen die aktuelle Abstiegsrichtung immer noch sehr gut ist.



## 2. Krümmungsbedingung (starke Wolfe Bedingung)

$$|\nabla f(x^k + \tau d^k)^\top d^k| \leq \eta |\nabla f(x^k)^\top d^k| \quad \eta \in (\delta, 1)$$

Mit dieser strikteren Bedingung ist die Zielposition meist in der Nähe eines lokalen Optimums von  $h(\tau)$



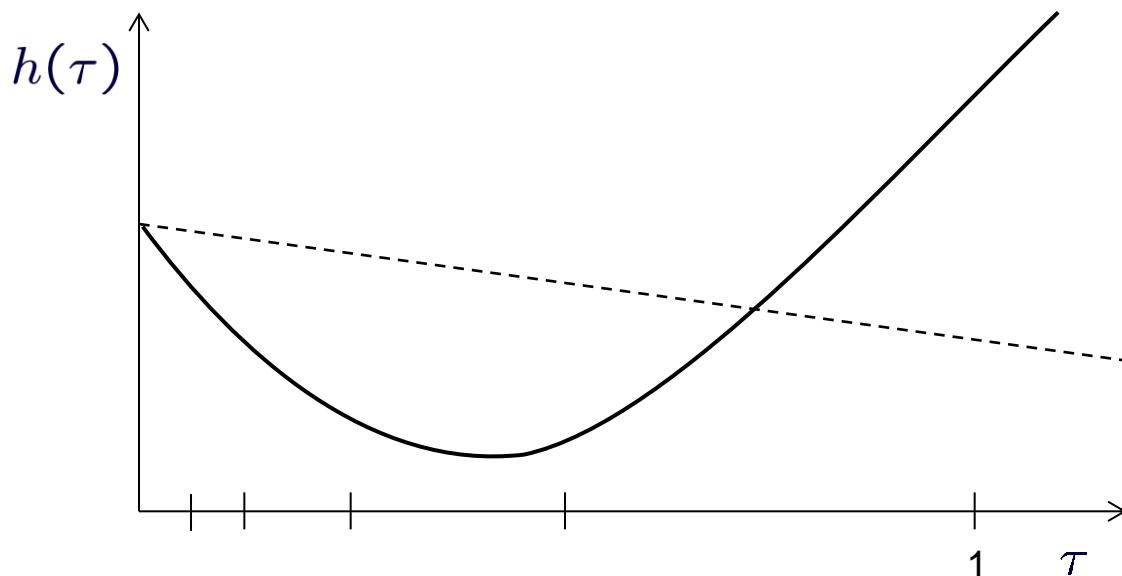
Annahmen:

- $f(\mathbf{x})$  ist kontinuierlich differenzierbar
- $\mathbf{d}^k$  ist eine Abstiegsrichtung an der Stelle  $\mathbf{x}^k$
- $f(\mathbf{x})$  ist entlang des Strahls  $[\mathbf{x}^k + \tau \mathbf{d}^k | \tau > 0]$  von unten beschränkt
- $0 < \delta < \eta < 1$

Sind diese Annahmen erfüllt, existieren Intervalle von Schrittweiten, welche die starken Wolfe Bedingung erfüllen.

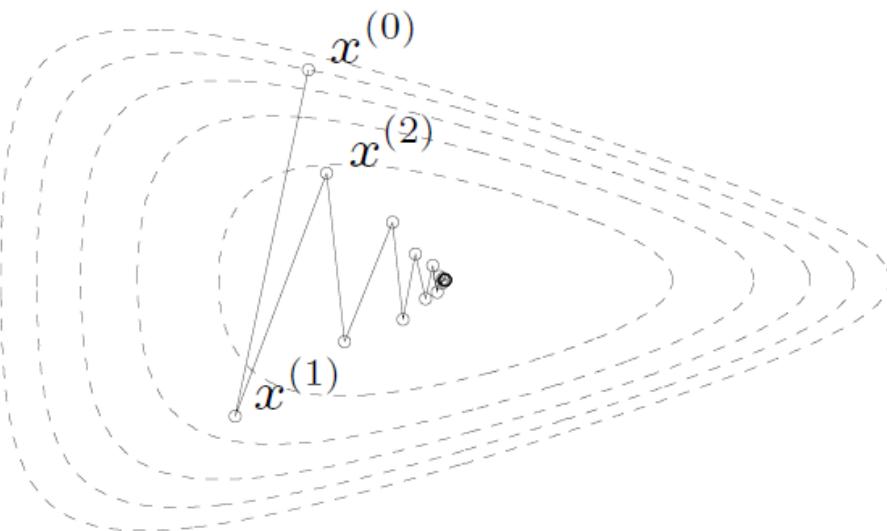
(Beweis in Nocedal-Wright pp. 35)

- Ziel: Approximative Minimierung von  $h(\tau) = f(x^k + \tau d^k)$ ,  $\tau \in (0, 1]$

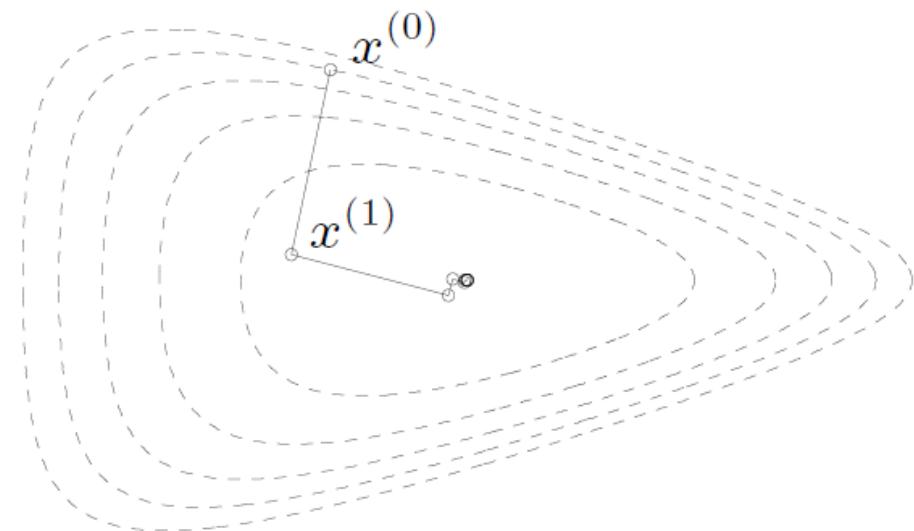


- Starte mit  $\tau^0 = 1$  und reduziere  $\tau^{k+1} = \beta\tau^k$ ,  $\beta \in (0, 1)$  solange bis die Armijo-Bedingung erfüllt ist.
- Lässt den Gradienten ungenutzt

# Relevanz der Schrittweite



Backtracking

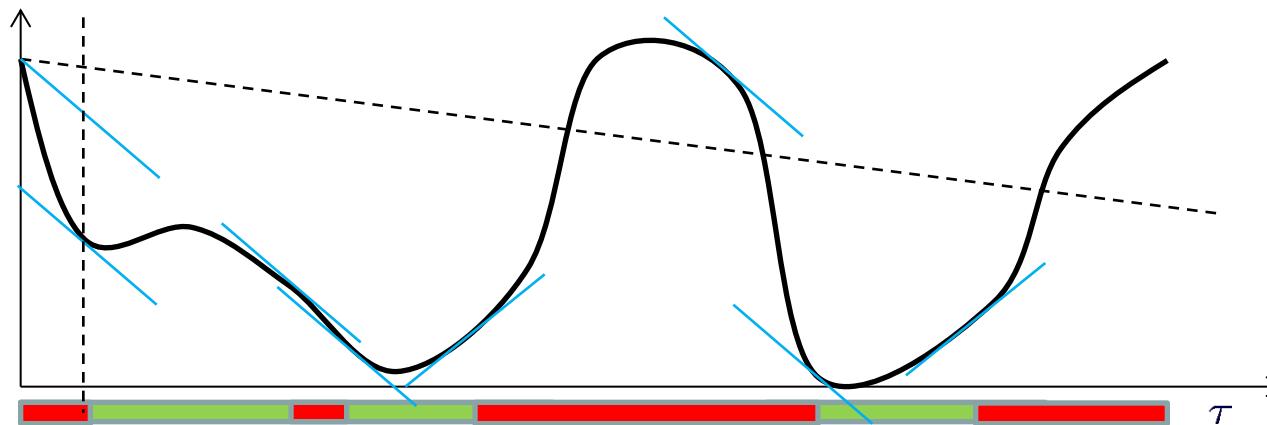


Optimale Schrittweite

- Starte mit  $\tau_0$  (z.B.  $\tau_0 = 1$ )
- Erfüllt  $\tau_0$  die Armijo-Bedingung, sind wir fertig
- Andernfalls gibt es im Intervall  $(0, \tau_0)$  eine Schrittweite, welche die Armijo-Bedingung erfüllt
- Quadratische Approximation von  $h(\tau)$  mithilfe von  $h(0), h'(0), h(\tau_0)$ ,  
$$h_q(\tau) = \left( \frac{h(\tau_0) - h(0) - \tau_0 h'(0)}{\tau_0^2} \right) \tau^2 + h'(0)\tau + h(0)$$
- Minimum bei  $\tau_1 = \frac{h'(0)\tau_0^2}{2(h(\tau_0) - h(0) - h'(0)\tau_0)}$
- Solange  $\tau_i$  die Armijo-Bedingung noch nicht erfüllt, kubische Interpolation mit  $h(0), h'(0), h(\tau_0), h(\tau_1)$ ,

# Line Search Verfahren für die Wolfe Bedingungen

- Bisher nur die Armijo-Bedingung sichergestellt
- Wolfe-Bedingungen aufwendiger zu erfüllen
- Verfahren bestehen aus zwei Komponenten:
  - **Bracketing** erweitert das Suchintervall bis darin geeignete Schrittweiten garantiert werden können
  - **Zooming** reduziert das Suchintervall bis eine geeignete Schrittweite gefunden wird (basierend auf Interpolationsverfahren)



- Details in Nocedal-Wright pp. 60; Public Domain Softwarepakete

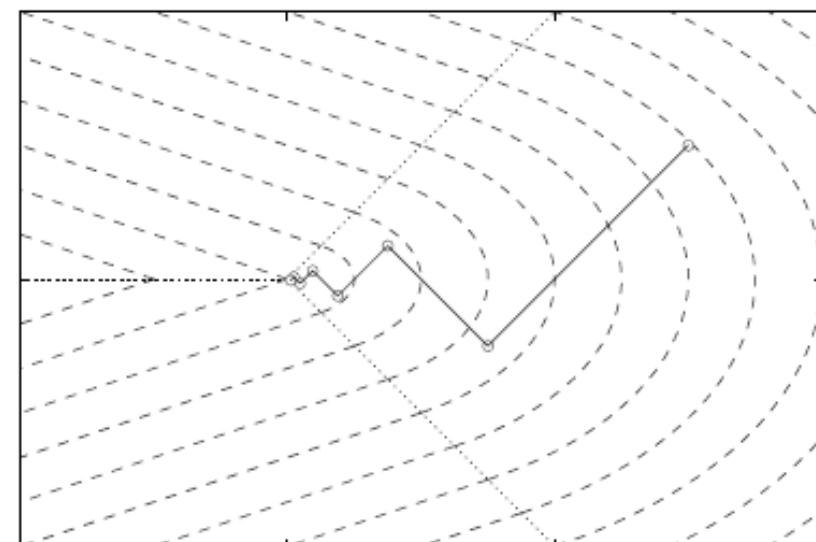
- Das Gradientenabstiegsverfahren mit einer Schrittweite, welche die Armijo-Bedingung erfüllt, konvergiert zu einem lokalen Minimum.
- Grund: Die Armijo-Bedingung

$$f(\mathbf{x}^k + \tau \mathbf{d}^k) - f(\mathbf{x}^k) \leq \delta \tau \nabla f(\mathbf{x}^k)^\top \mathbf{d}^k, \quad \delta \in (0, 1)$$

stellt sicher, dass der Funktionswert in jeder Iteration hinreichend sinkt, solange die notwendige Bedingung nicht erfüllt ist, also  $|\nabla f(\mathbf{x}^k)| > 0$

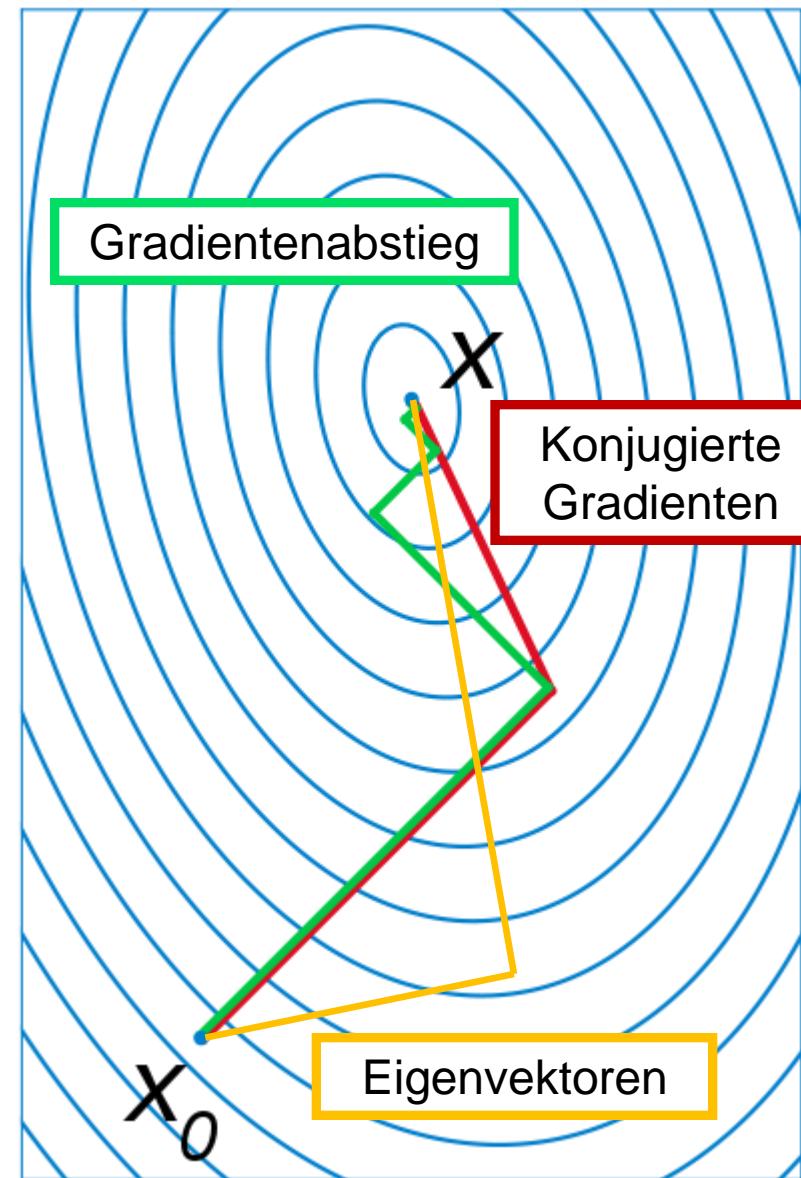
- Erinnerung: Wir gehen davon aus, dass  $f$  einmal stetig differenzierbar ist.

Notwendig für die Bestimmung des Gradienten und für Konvergenz zu einem lokalen Minimum.



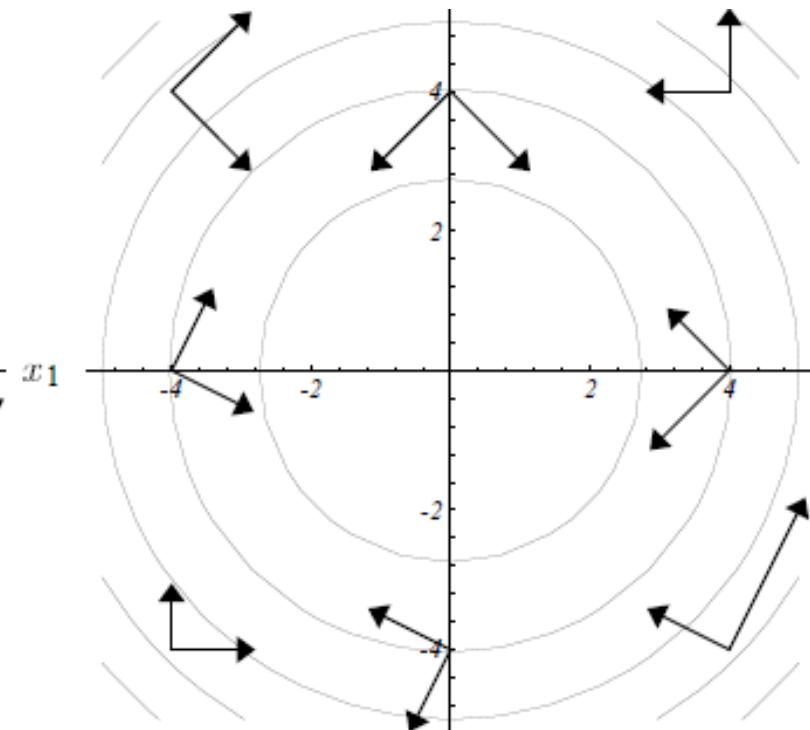
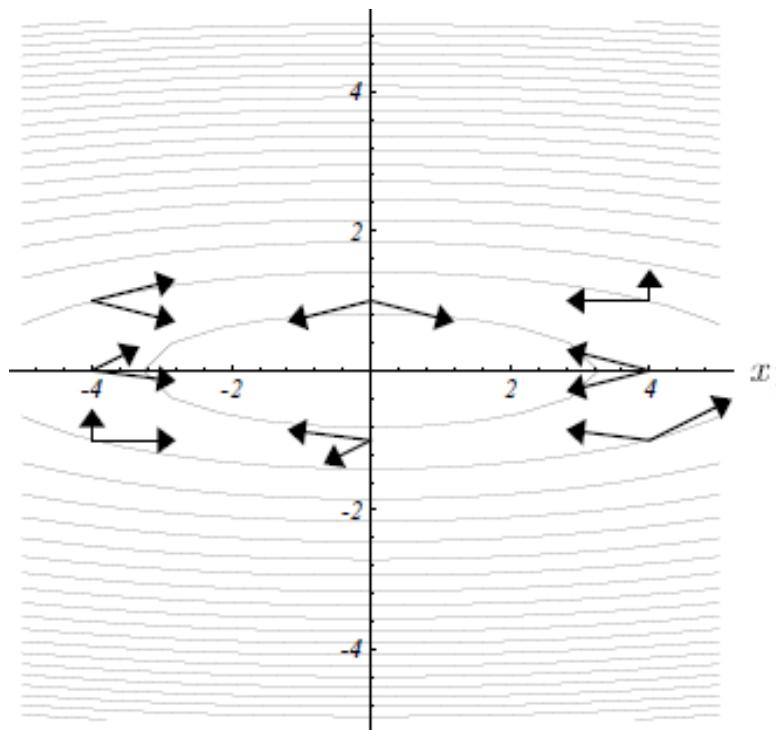
Nicht differenzierbares Beispiel  
Fixpunkt ist kein lokales Minimum

- Auf dem Weg zum Minimum werden immer wieder dieselben Richtungen gewählt  
→ ineffizient
- Mit einer Reihe von orthogonalen Richtungen wären wir wesentlich schneller beim Minimum.
- Im Falle einer quadratischen Zielfunktion
$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top A \mathbf{x} + \mathbf{b}^\top \mathbf{x}$$
sogar in  $n$  Schritten
- Hierfür bräuchten wir aber die Eigenvektoren von  $A$   
→ auch ineffizient



- Statt Orthogonalität im Euklidischen Raum, Orthogonalität mit der durch  $A$  definierten Metrik:

$$\mathbf{x}^\top A \mathbf{x} \neq 0, \quad \mathbf{x}^\top A \mathbf{y} = 0$$



A-orthogonale Vektoren für zwei unterschiedliche quadratische Zielfunktionen  
 Quelle: J. Shewchuk

- Wähle als erste Richtung den Gradienten an der Startposition  $x^0$ :

$$d^0 = -\nabla f(x^0) = b - Ax^0$$

- Wähle die optimale Schrittweite:

$$\tau^k = \frac{|\nabla f(x^k)|^2}{d^{k\top} Ad^k}$$

analytische Lösung im Fall quadratischer Funktionen, da dann  $h'(\tau) = 0$  eine lineare Gleichung ist

- Neue Lösung:

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

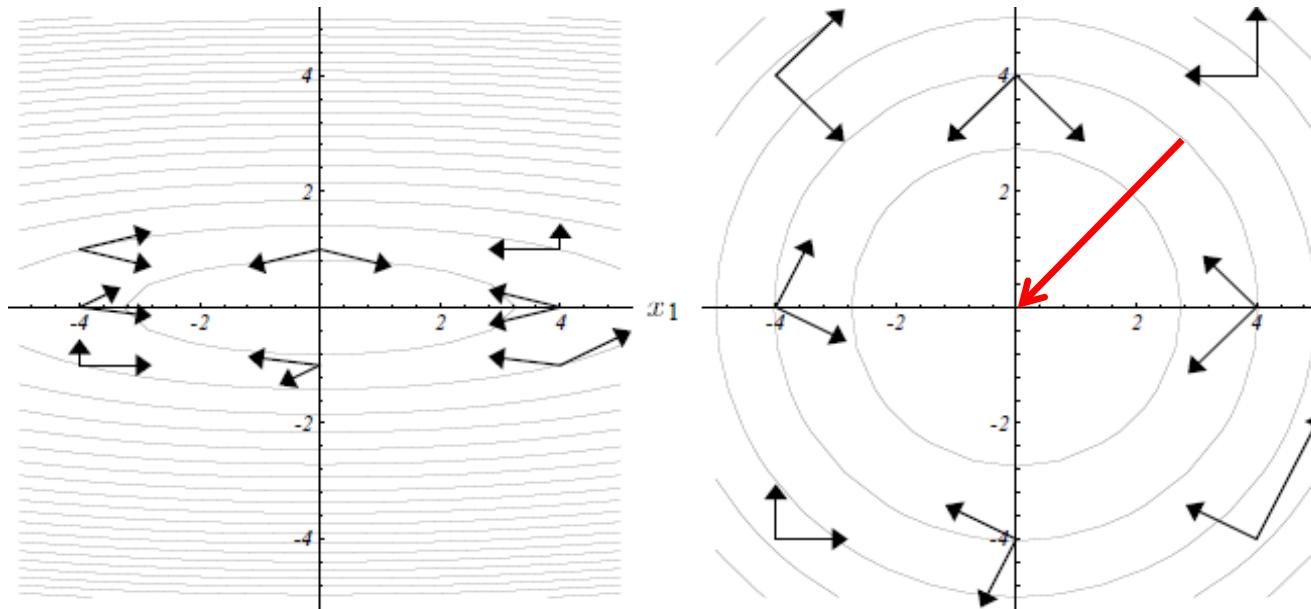
- Wähle neue Richtung

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

mit  $\beta^k = \frac{|\nabla f(x^{k+1})|^2}{|\nabla f(x^k)|^2}$  (Lösung für die  $d^{k+1}$  und  $d^k$  orthogonal sind)

## CG Verfahren zum Lösen linearer Gleichungssysteme

- Der Minimierer von  $f(x) = \frac{1}{2}x^\top Ax - b^\top x$  entspricht der Lösung des linearen Gleichungssystems  $Ax = b$
- Notwendige Bedingung für ein Minimum von  $f(x) = \frac{1}{2}x^\top Ax - b^\top x$   
 $\nabla f(x) = Ax - b = 0$
- Das CG Verfahren wird daher meist zum Lösen linearer Gleichungssysteme eingesetzt.
- Die Effizienz hängt von der **Konditionszahl** von  $A \in \mathbb{R}^{n \times n}$  ab.
- Die Konditionszahl einer Matrix ist der größte Eigenwert geteilt durch den kleinsten.
- Bei exakter Zahlendarstellung garantiert das CG Verfahren in  $n$  Schritten die exakte Lösung, bei kleiner Konditionszahl bereits wesentlich schneller.



Links: hohere Konditionszahl. CG Verfahren konvergiert nach 2 Schritten.  
Rechts: Konditionszahl = 1. CG Verfahren konvergiert in einem Schritt.

Quelle: J. Shewchuk

- Das CG Verfahren wird daher oft mit einem **Prakonditionierer** kombiniert, der vorab die Konditionszahl der Matrix reduziert.

- Wähle als erste Richtung den Gradienten an der Startposition  $x^0$ :

$$p^0 = -\nabla f(x^0)$$

- Wähle eine Schrittweite  $\tau^k$ 
  - mithilfe von Line Search (keine analytische Lösung verfügbar)

- Neue Lösung:

$$x^{k+1} = x^k + \tau^k p^k$$

- Wähle neue Richtung

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

$$\text{mit } \beta^k = \frac{|\nabla f(x^{k+1})|^2}{|\nabla f(x^k)|^2} \quad (\text{Fletcher/Reeves})$$

- Bessere Varianten (z.B. Polak/Ribiére) verfügbar

- Gradientenabstieg ist ein einfaches Verfahren zur lokalen Minimierung allgemeiner differenzierbarer Funktionen.
- Die Schrittweite muss bestimmte Bedingungen erfüllen um Konvergenz und Effizienz zu garantieren (Wolfe Bedingungen)
- Eine effizientere Variante ist das CG Verfahren. Insbesondere quadratische Funktionen lassen sich damit effizient minimieren.

1. Machen Sie sich mit Python vertraut.

2. Minimieren Sie die Funktion

$$f(x) = \frac{1}{2}x^4 + 2x^3 - 3x - 4$$

- Visualisieren Sie die Funktion. Ist sie konvex?
- Berechnen Sie den Gradienten.
- Optimieren Sie die Funktion mit Gradientenabstieg. Verwenden Sie bitte Ihre eigene Implementierung. Probieren Sie verschiedene Startpunkte aus. Implementieren Sie Backtracking Line Search zur Bestimmung der Schrittweite.
- Visualisieren Sie, was während der Optimierung geschieht. Für diese 1D-Funktion geht diese sehr schön, bei hochdimensionalen Problemen aus der Praxis nicht mehr.

3. Minimieren Sie die zweidimensionale Funktion

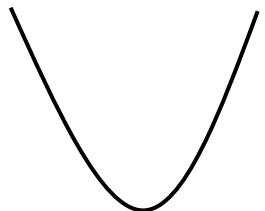
$$f(x_1, x_2) = 2x_1^2 + 2x_2^2 + 3x_1x_2 - x_1 + x_2$$

- Visualisieren Sie die Isolinien. Ist die Funktion konvex?
- Berechnen Sie den Gradienten.
- Optimieren Sie die Funktion mit Gradientenabstieg. Verwenden Sie bitte wieder Ihre eigene Implementierung.
- Optimieren Sie die Funktion analytisch. Stimmt Ihre numerische Lösung mit dem Minimum überein?

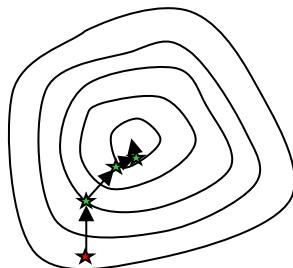
# Optimierung

---

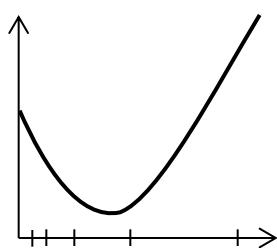
Vorlesung 3  
Newton- und Quasi-Newton-Verfahren



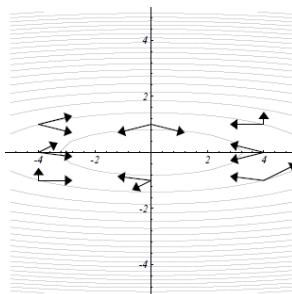
Konvexität (Garantie globaler Minima)



Gradientenabstieg  
(allgemeines Verfahren für kontinuierliche Probleme)



Line Search (zur Bestimmung der optimalen Schrittweite)



Conjugate Gradients (CG)  
(effizientes Verfahren für quadratische Funktionen)

- Iteratives Verfahren

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \tau^k \mathbf{d}^k$$

mit einem Startpunkt  $\mathbf{x}^0$ , einer Änderungsrichtung  $\mathbf{d}^k$  und einer Schrittweite  $\tau^k$

- Beim Gradientenverfahren entspricht die Änderungsrichtung dem negativen Gradienten der Zielfunktion  $f$  an der aktuellen Stelle  $\mathbf{x}^k$

$$\mathbf{d}^k := -\nabla f(\mathbf{x}^k) \quad \text{also } \mathbf{x}^{k+1} = \mathbf{x}^k - \boxed{\tau^k} \boxed{\nabla f(\mathbf{x}^k)}$$

- Gradientenverfahren verwenden nur die 1. Ableitung der Funktion. Die 2. Ableitung (Krümmung) wird ignoriert.
- Verfahren 1. Ordnung vs. Verfahren 2. Ordnung
- Annahme: die 2. Ableitung existiert, d.h.  $f \in \mathcal{C}^2$  (2x stetig differenzierbar)

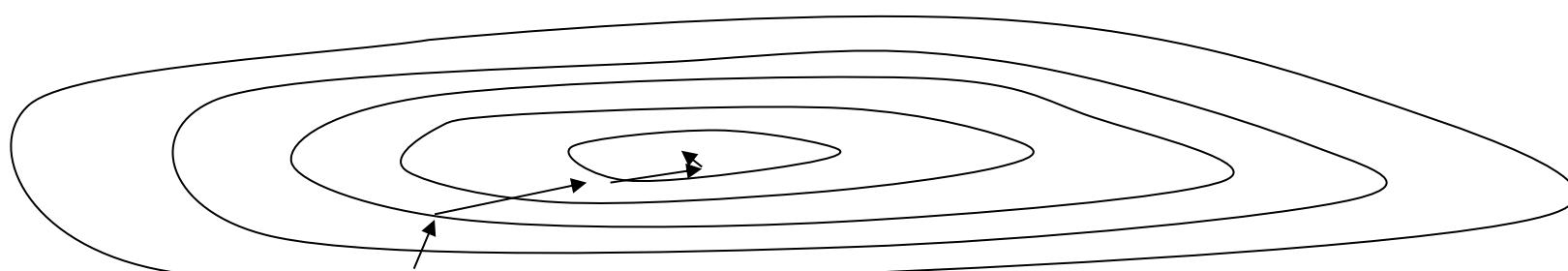
- Das **Newton-Verfahren** verwendet auch die 2. Ableitung (Krümmung):

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

- Im mehrdimensionalen Fall ist diese durch die **Hesse Matrix**  $H$  gegeben:

$$H_f(x_1, \dots, x_n) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \dots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

- Motivation: größere Schritte in Richtungen mit schwacher Krümmung

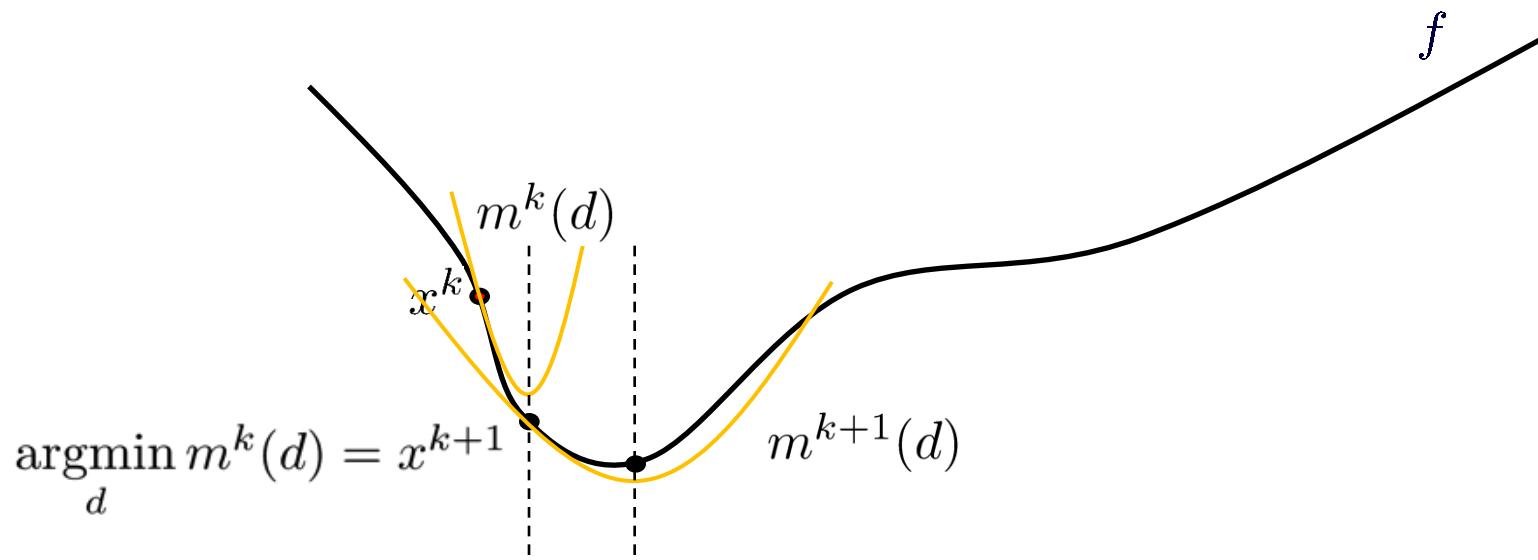


- Zielfunktion lässt sich an der Stelle  $x^k$  durch Taylor-Reihe darstellen:

$$f(x^k + d) = f(x^k) + d^\top \nabla f(x^k) + \frac{1}{2} d^\top H(x^k) d + O(d^3)$$

→ d.h. die Funktion wird lokal durch eine quadratische Funktion approximiert

$$f(x^k + d) = f(x^k) + d^\top \nabla f(x^k) + \frac{1}{2} d^\top H(x^k) d = m^k(d)$$



- Für  $x^{k+1} = x^k + d^k$  müssen wir also  $\operatorname{argmin}_d m^k(d)$  bestimmen, wobei

$$m^k(d) = f(x^k) + d^\top \nabla f(x^k) + \frac{1}{2} d^\top H(x^k) d$$

- → quadratisches Optimierungsproblem

- 1. Ableitung von  $m^k(d)$  nach  $d$ :

$$\nabla m^k(d) = \nabla f(x^k) + H(x^k) d \quad (*)$$

- Setzten der Ableitung gleich 0 ergibt

$$d^k = -H_f^{-1}(x^k) \nabla f(x^k)$$

- Bemerkung: (\*) gilt nur da  $H(x^k)$  symmetrisch ist.

- In beiden Verfahren haben wir eine Abstiegsrichtung  $\mathbf{d}^k$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \tau^k \mathbf{d}^k$$

und müssen eine gute/optimale Schrittweite  $\tau^k$  finden.

- Gradientenabstieg (Verfahren 1. Ordnung)

$$\mathbf{d}^k := -\nabla f(\mathbf{x}^k)$$

- Newton-Verfahren (Verfahren 2. Ordnung)

$$\mathbf{d}^k := -H^{-1}(f(\mathbf{x}^k))\nabla f(\mathbf{x}^k)$$

- Bestimmung der Schrittweite in beiden Fällen mittels Line Search

- Hinzunahme der Krümmung hat verschiedene Vor- und Nachteile

- Konvergenz von Folgen  $s_1, s_2, \dots, \lim_{k \rightarrow \infty} s_k = \bar{s}$
- In unserem Fall: Die Konvergenz der Punkte  $x_k$  zur Lösung des Optimierungsproblems (bzw. zur Nullstelle der Ableitung)
- Lineare Konvergenz

$$\lim_{k \rightarrow \infty} \frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|} = C < 1$$

Beispiel:  $s_k = 0.9^k$

- Superlineare Konvergenz

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

Beispiel:  $s_k = \frac{1}{k!}$

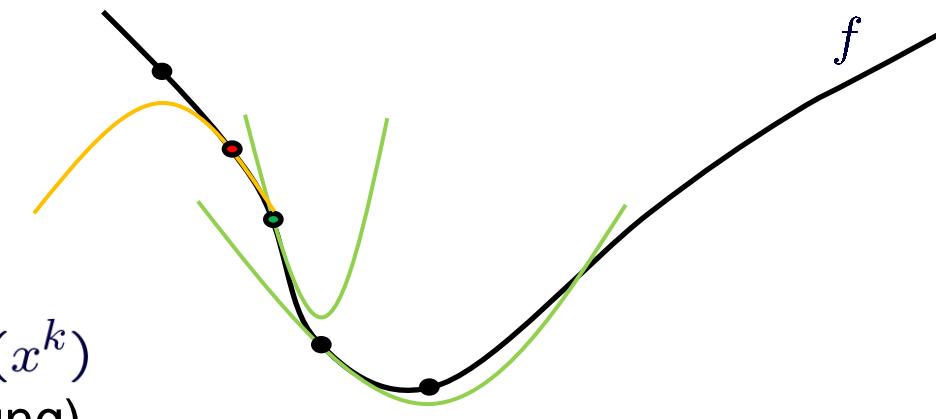
- Quadratische Konvergenz

$$\lim_{k \rightarrow \infty} \frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|^2} = C < 1$$

Beispiel:  $s_k = 0.9^{(2^k)}$

- Gradientenabstieg konvergiert linear
- Das Newton-Verfahren konvergiert für glatte, konvexe Funktionen nahe der Lösung quadratisch
- Insbesondere konvergiert das Newton-Verfahren für quadratische Funktionen in einem Schritt
- Das Newton-Verfahren ist also nominal wesentlich effizienter als Gradientenabstieg
- Zu berücksichtigen: Berechnung und Invertierung der Hesse-Matrix  
 $d^k := -\nabla f(x^k)$    vs.    $d^k := -H^{-1}(f(x^k))\nabla f(x^k)$   
→ jeder einzelne Iterationsschritt ist wesentlich aufwendiger

- Ist die Krümmung negativ, läuft das Newton-Verfahren in die falsche Richtung!
- Damit  $d^k = -H_f^{-1}(x^k)\nabla f(x^k)$  eine Abstiegsrichtung ist, muss  $H_f(x^k)$  positiv definit sein (positive Krümmung)
- Konvergenzgarantie für konvexe Funktionen (diese haben überall positive Krümmung)
- Bei nicht-konvexen Funktionen: Manipulation der Hesse Matrix, so dass alle ihre Eigenwerte positiv sind
- Beim Gradientenverfahren ist die Richtung per Definition eine Abstiegsrichtung



## Gradientenabstieg

- Lineare Konvergenz
- Konvergenz für alle glatte Funktionen
- Jede Iteration:
  - Gradientenberechnung
  - Line Search

## Newton-Verfahren

- Quadratische Konvergenz
- Konvergenz für glatte, streng konvexe Funktionen (sonst zusätzliche Maßnahmen nötig)
- Jede Iteration:
  - Gradientenberechnung
  - Berechnung Hesse Matrix
  - Invertierung der Hesse Matrix
  - Line Search (nicht nötig bei quadratischen Funktionen)

Gibt es Verfahren mit superlinearer Konvergenz aber schnelleren Iterationen?

- Zwei Grundideen:
  - Approximation der Hesse Matrix mithilfe des Gradienten (1. Ordnung)
  - Berechnung der Approximation aus dem vorherigen Iterationsschritt

- Newton-Verfahren:

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

- Quasi-Newton Verfahren:

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

- Wie erhält man eine gute Approximation der Hesse-Matrix?

- Taylor Approximation des Gradienten (ges.: Nullstelle der Ableitung!)

$$\nabla f(x^k + d) \approx \nabla f(x^k) + H_f(x^k)d$$

- Mit  $d = x^{k+1} - x^k$  erhält man

$$\nabla f(x^{k+1}) - \nabla f(x^k) \approx H_f(x^k)(x^{k+1} - x^k)$$

- Die Approximation  $B^k$  sollte dies auch erfüllen.

→ Quasi-Newton Bedingung (Sekantengleichung):

$$B^k s^k = y^k$$

mit  $s^k := x^k - x^{k-1}$

$$y^k := \nabla f(x^k) - \nabla f(x^{k-1})$$

- Um diese Gleichung erfüllen zu können muss gelten (siehe nächste Folie)

$$(s^k)^\top y^k > 0$$

→ Bedingung für die Schrittweitensuche

Beweis von  $B^k s^k = y^k$  impliziert  $(s^k)^\top y^k > 0$

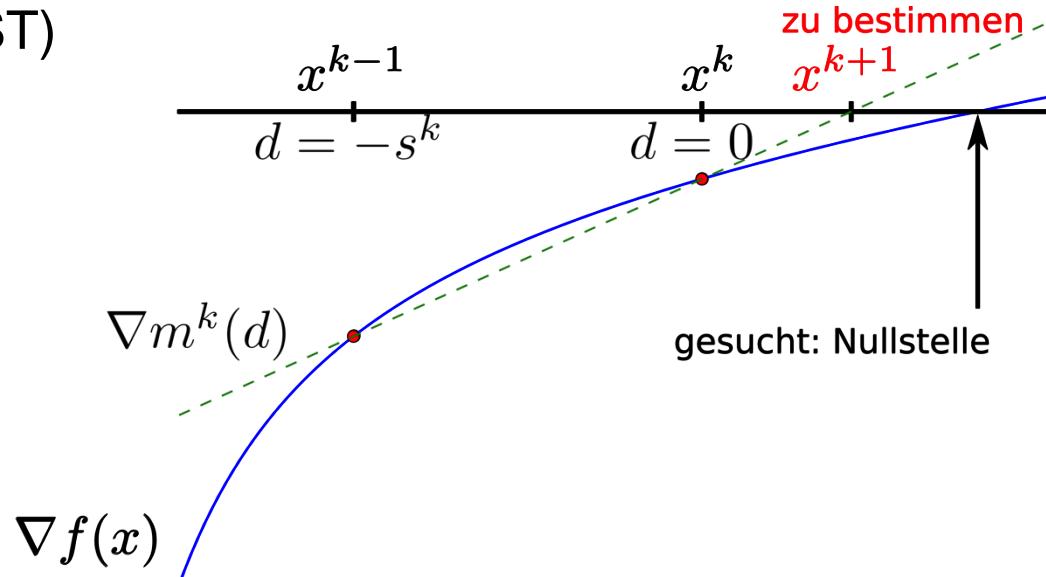
- B Positive definite

- Das heisst:  $\forall z \neq \vec{0} : z^\top B^k z > 0$
- Aus  $B^k s^k = y^k$  folgt  $(s^k)^\top B^k s^k = (s^k)^\top y^k$
- Es gilt  $(s^k)^\top B^k s^k > 0$  und damit  $(s^k)^\top y^k > 0$

# Erklärung für Sekantengleichen

- Wir suchen eine Nullstelle (NST) der Ableitung  $\nabla f(x)$
- Lineare Approximation mit Matrix  $B^k$ :

$$\nabla m^k(d) = \nabla f(x^k) + B^k d$$



- Sekantenmethode zur NST-Suche -> Bedingungen:

$$\nabla m^k(\vec{0}) = \nabla f(x^k) \text{ und } \nabla m^k(-s^k) = \nabla f(x^{k-1}) \quad \text{mit } s^k = x^k - x^{k-1}$$

- 1. Bedingung automatisch erfüllt, 2. Bedingung kann umformuliert werden da  $\nabla m^k(-s^k) = \nabla f(x^k) - B^k s^k$  gilt:

$$\begin{aligned} \nabla f(x^k) - B^k s^k &= \nabla f(x^{k-1}) \\ B^k s^k &= \nabla f(x^k) - \nabla f(x^{k-1}) =: y^k \end{aligned}$$

- Bisher folgende Bedingungen an  $B^k$ :
  - Symmetrische Matrix  $B^k = (B^k)^\top$
  - Positiv definit (alle Eigenwerte  $> 0$ )
  - Sekantengleichung  $B^k s^k = y^k$
- Dies führt immer noch zu vielen möglichen Lösungen  
→ weitere Bedingung

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

(Approximation ist möglichst ähnlich zu der im letzten Iterationsschritt)

- Verschiedene Matrixnormen führen zu verschiedenen Unterverfahren.
- Mit einer gewichteten Frobeniusnorm, bei der die Gewichtsmatrix der mittleren Hesse-Matrix entspricht, führt dies zum Verfahren von Davidon, Fletcher und Powell (ohne Beweis).



- Von Davidon 1959 aus der Not entwickelt da sein Computer grundsätzlich abstürzte bevor die Optimierung abgeschlossen war.
- Neue Approximation der Hesse-Matrix wird aus der vorherigen berechnet:

$$B^{k+1} = \left( I - \rho^k y^k (s^k)^\top \right) B^k \left( I - \rho^k s^k (y^k)^\top \right) + \rho^k y^k (y^k)^\top$$

$$\rho^k := \frac{1}{(y^k)^\top y^k}$$

- Da man ohnehin an der inversen Hesse-Matrix interessiert ist, ist es vorteilhaft direkt die inverse Approximation  $Q := B^{-1}$  zu iterieren:

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

- Die Bedingung

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

kann auch auf die inverse Approximation angewendet werden:

$$Q^k = \min_Q \|Q - Q^{k-1}\|$$

- Dies führt zum noch etwas effizienteren Verfahren von Broyden-Fletcher-Goldfarb-Shanno (BFGS):

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

- Aus Effizienzgründen ist die Reihenfolge der Operationen so zu wählen, dass keine Matrix-Matrix Multiplikationen entstehen.

(Multiplizierte erst  $Q^k$  mit  $\mathbf{y}^k$  und dann das Ergebnis mit  $s^k$ )

- Eingabe:
  - Startpunkt  $x^0$
  - Initiale Approximation  $Q^0$  (z.B.  $Q^0 = I$ )

- Iterationen:
  - Berechne Abstiegsrichtung:

$$\mathbf{d}^k = -Q^k \nabla f(\mathbf{x}^k)$$

- Bestimme Schrittweite  $\tau^k$  durch Line Search mit Wolfe Bedingungen (wichtig)
- Berechne neue Lösung:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \tau^k \mathbf{d}^k$$

- Berechne neue Approximation der inversen Hesse Matrix:

$$\mathbf{s}^k = \mathbf{x}^{k+1} - \mathbf{x}^k, \quad \mathbf{y}^k = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k), \quad \rho^k = \frac{1}{(\mathbf{y}^k)^\top \mathbf{y}^k}$$

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

- Superlineare Konvergenz. Was heißt das in der Praxis?

Gradientenabstieg: 5264 Iterationen

BFGS: 34 Iterationen

Newton: 21 Iterationen

Beispielproblem:  $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

Abbruchkriterium:  $\|\nabla f(x^k)\| \leq 10^{-5}$

- $Q^k$  ist immer positiv definit wenn  $Q^{k-1}$  positiv definit war.
- Line search sollte immer  $\tau^k = 1$  testen, da diese Schrittweite in den meisten Iterationen akzeptiert wird und zu schneller Konvergenz führt.
- Die Wolfe Bedingungen sind wichtig, damit BFGS zeitweise schlechte Approximationen  $Q$  wieder korrigiert.

- Eine sehr häufige Klasse von Funktionen:

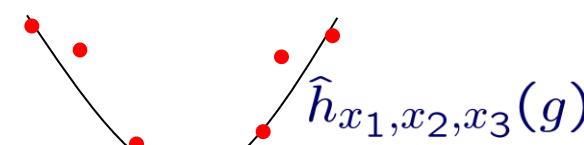
$$f(x) = \frac{1}{2} \sum_{j=1}^m r_j^2(x)$$

- Modell mit  $m$  Fehlertermen (**Residuen**)  $r_j$
- Beispiel 1: Viele Probleme aus dem maschinellen Lernen
- Beispiel 2: Kurve an Datenpunkte anpassen

Quadratische Kurve beschrieben durch drei Parameter  $x = (x_1, x_2, x_3)^\top$

Estimate  $\hat{h}_j = x_1 g_j^2 + x_2 g_j + x_3$

Gegeben  $m$  Datenpunkte  $(g_j, h_j)^\top$



- Optimierungsproblem:  $f(x) = \frac{1}{2} \sum_{j=1}^m (\underbrace{x_1 g_j^2 + x_2 g_j + x_3}_{r_j(x)} - h_j)^2$

- Problemklasse hat eine nützliche Struktur
- Wir können die Residuen als einen **Residuenvektor** schreiben:

$$r(x) = (r_1(x), \dots, r_m(x))^\top$$

- Die Zielfunktion lässt sich dann in Kurzform schreiben:

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

- Die Ableitungen der Residuen bilden die  $m \times n$  **Jacobimatrix**

$$J(x) = \begin{pmatrix} \nabla r_1(x)^\top \\ \vdots \\ \nabla r_m(x)^\top \end{pmatrix}$$

- Damit lässt sich z.B. der Gradient von  $f(x)$  kompakt darstellen:

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

- Auch die Hesse-Matrix bekommt eine besondere Form:

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

- Teil der Hesse-Matrix lässt sich aus Ableitungen 1. Ordnung berechnen!
- Spezialfall linearer Residuen (**linear least squares**):  $r_j = A_j x - b_j$  (entspricht einem quadratischen Optimierungsproblem)
- In diesem Fall  $J(x) = A$  und der zweite Term der Hesse-Matrix fällt weg
- Optimierungsproblem durch lineares Gleichungssystem beschrieben:  
$$J^\top J x = J^\top b$$

- Für nichtlineare Residuen bietet sich wieder eine Approximation der Hesse-Matrix an:

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

→ Approximation des Newton-Verfahrens nur mit 1. Ableitungen

- Anders als bei BFGS durch Ausnutzen der speziellen Problemstruktur (Summe quadrierter Residuen)
- Durch Vernachlässigen des zweiten Terms, werden die Residuen lokal an der aktuellen Lösung  $x^k$  linearisiert:

$$r_j(x^k + d^k) = r_j(x^k) + J(x^k)d^k + \frac{1}{2}(d^k)^\top \nabla^2 r_j(x^k)d^k + \dots$$

- Linearisierung der Residuen

$$r_j(x^k + d^k) \approx r_j(x^k) + J(x^k)d^k$$

führt daher zu einem linearen Least-Squares-Problem mit Parametern  $d^k$

- Lineares Gleichungssystem

$$J^\top(x^k)J(x^k)d^k = -J^\top(x^k)r(x^k)$$

- Die Lösung  $d^k$  ist eine Abstiegsrichtung, mit der  $x^k$  verbessert wird:

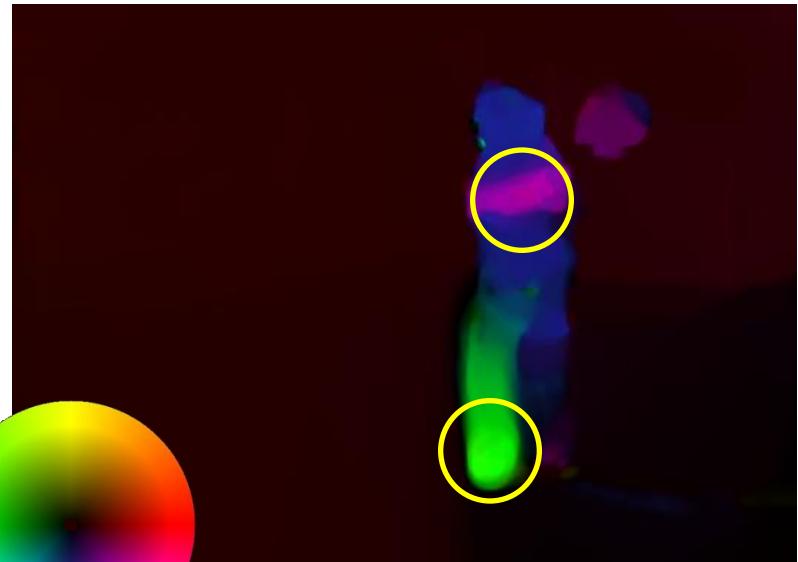
$$x^{k+1} = x^k + \tau^k d^k$$

- Gauss-Newton löst also eine Sequenz von linearen Gleichungssystemen
- Gauss-Newton funktioniert, wenn alle Eigenwerte von  $J^\top J$  deutlich positiv sind. Bei Eigenwerten nahe null → Levenberg-Marquardt

# Beispiel für Gauss-Newton-Verfahren: Bewegungsschätzung



Zwei Bilder einer Bildsequenz  
 $I_1(x, y), I_2(x, y)$



Bewegungsfeld  
 $(u, v)(x, y)$

- Optimierungsproblem:

$$f(u, v) = \int (I_2(x+u(x, y), y+v(x, y)) - I_1(x, y))^2 + \alpha (|\nabla u(x, y)|^2 + |\nabla v(x, y)|^2) \, dx \, dy$$

- Das **Newton-Verfahren** berücksichtigt neben dem Gradienten zusätzlich die Krümmung zur Bestimmung der Abstiegsrichtung
- **Quasi-Newton-Verfahren** approximieren die Krümmung und erreichen so schnelle Konvergenz bei niedrigeren Kosten pro Iteration
- Auch das **Gauss-Newton-Verfahren** approximiert die Krümmung. Hierbei wird die spezielle Struktur von Least-Squares-Problemen ausgenutzt.

# Übungsaufgabe

1. Laden Sie das Bild `puppy.png` mit `imread` aus der Library `scipy.misc` und lassen Sie es sich mit `pyplot's imshow` funktion anzeigen. Das Bild ist offenbar etwas verrauscht. Wir würden gerne ein schöneres Bild rekonstruieren, was auf das folgende Optimierungsproblem hinausläuft:

$$f(x) = \sum_{i,j} \left( \sqrt{(x_{ij} - y_{ij})^2 + 1} + \frac{1}{2} \sqrt{(x_{ij} - x_{i+1j})^2 + (x_{ij} - x_{ij+1})^2 + 1} \right)$$

wobei  $y_{ij}$  die gemessenen Bildpixel darstellen und  $x_{ij}$  die Bildpixel des verbesserten Bildes.

- Berechnen Sie den Gradienten der Funktion.
  - Optimieren Sie die Funktion mit Gradientenabstieg. Starten Sie mit dem Eingangsbild als Startpunkt, also  $x = y$  (wandeln Sie am besten das Eingangsbild in `float64` um). Bestimmen Sie die Schrittweite wieder mit Backtracking Line Search. Speichern Sie die gewählte Schrittweite bei jeder Iteration und lassen Sie sich die Schrittweiten über die Iterationen später anzeigen. Verfolgen Sie auch die Konvergenz (also den Funktionswert über die Iterationen) und schauen Sie, wie sich die Lösung  $x$  in Form des Bildes über die Iterationen verändert.
2. Versuchen sie nun die Funktion von Scipy's `optimize` optimieren zu lassen. Vergleichen sie das BFGS und das Truncated Newton Verfahren. Welches benötigt weniger schritte? Wie lange dauert ein Schritt?

# Optimierung

---

Vorlesung 4 (Erweiterung)  
Erklärung Dualität

- Formulierung 1:  $\min_x f(x)$  subject to  $c_i(x) \geq 0, i \in \mathcal{I}$
- Dazugehörige Lagrangefunktion:  $\mathcal{L}(x, \lambda) = f(x) - \sum_i \lambda_i c_i(x)$
- Äquivalente Formulierung:  $\min_x f(x)$  subject to  $c_i(x) \leq 0, i \in \mathcal{I}$
- Dazugehörige Lagrangefunktion:  $\mathcal{L}(x, \lambda) = f(x) + \sum_i \lambda_i c_i(x)$

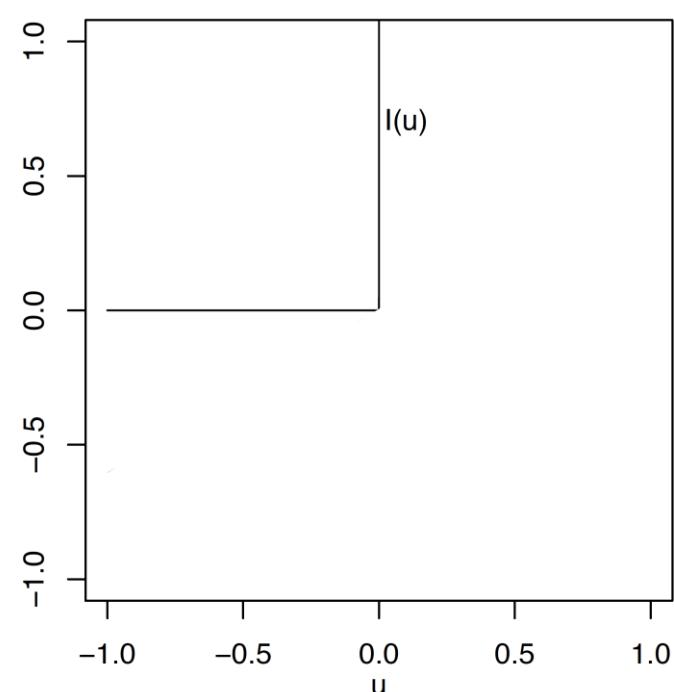
# Step Funktion

- Lagrangefunktion:  $\mathcal{L}(x, \lambda) = f(x) + \sum_i \lambda_i c_i(x)$
- Eigentlich wollen wir nicht  $\mathcal{L}(x, \lambda)$  minimieren, sondern folgende Funktion:

$$\begin{aligned} J(x) &= \begin{cases} f(x) & \text{if } c_i(x) \leq 0 \\ \infty & \text{else} \end{cases} \\ &= f(x) + \sum_i I[c_i(x)] \end{aligned}$$

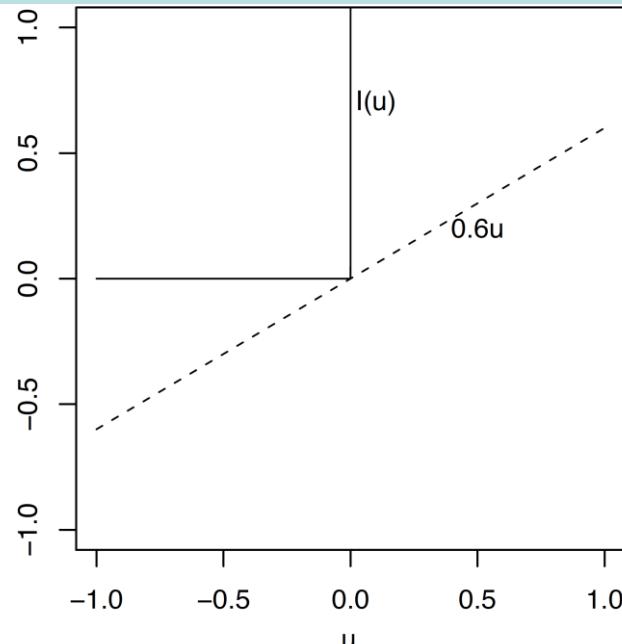
with  $I[u] = \begin{cases} 0 & \text{if } u \leq 0 \\ \infty & \text{else} \end{cases}$

- Wir suchen also:  $\min_x J(x)$
- Problem:  $I[u]$  nicht differenzierbar



# Step Funktion und Lagrange Funktion

- Daher: Approximiere  $I[u]$  durch lineare Funktion  $\lambda_i \cdot u$
- Aus  $J(x) = f(x) + \sum_i I[c_i(x)]$  wird dann:
$$\mathcal{L}(x, \lambda) = f(x) + \sum_i \lambda_i c_i(x)$$



- Aus dem Bild sieht man sofort dass die Lagrange-Funktion eine untere Schranke ist:

$$\forall \lambda \geq 0 : \mathcal{L}(x, \lambda) \leq J(x)$$

- Gegeben ein festes  $x$ , was ist das "beste"  $\lambda_i$ ?

$$c_i(x) \leq 0 \quad \Rightarrow \quad \lambda_i = 0 \quad (\text{Bem.: } \lambda > 0 \text{ verkleinert da } c_i(x) \leq 0)$$

$$c_i(x) > 0 \quad \Rightarrow \quad \lambda_i \rightarrow \infty$$

- Daher gilt (etwas überraschend?):
 
$$\max_{\lambda} \mathcal{L}(x, \lambda) = J(x)$$

- Da wir an  $\min_x J(x)$  interessiert sind, suchen wir nach

$$\min_x \max_{\lambda} \mathcal{L}(x, \lambda) = \min_x J(x)$$

- Das ist aber vielleicht schwer, daher schauen wir mal auf

$$\max_{\lambda} \min_x \mathcal{L}(x, \lambda)$$

- Daraus leiten wir das duale Optimierungsproblem

$$\max_{\lambda} g(\lambda) \text{ wobei } g(\lambda) = \min_x \mathcal{L}(x, \lambda)$$

- Aus  $\forall \lambda \geq 0 : \mathcal{L}(x, \lambda) \leq J(x)$  folgt

$$\min_x \mathcal{L}(x, \lambda) = g(\lambda) \leq \min_x J(x)$$

- Damit ist  $\max_{\lambda} g(\lambda)$  das Problem des Findens der besten unteren Schranke

- Wird das duale Problem genannt und ist einfacher, da konkav (siehe später)

- Durch die Nebenbedingungen kamen neben den primalen Variablen  $x$  die Lagrange Multiplikatoren  $\lambda$  als Variablen ins Spiel

$$\mathcal{L}(x, \lambda) = f(x) + \sum_i \lambda_i c_i(x), \quad \lambda \geq 0$$

- Sie geben an wie stark die Funktion gegen die Bedingungen drückt.

$$\nabla f(x) = \lambda^\top \nabla c(x), \quad \lambda \geq 0$$

- Alternativ können wir  $\mathcal{L}(x, \lambda)$  als zu minimierende Funktion ohne Nebenbedingungen betrachten, indem wir  $\lambda$  festhalten

$$q(\lambda) = \min_x \mathcal{L}(x, \lambda) \quad *$$

- In vielen Fällen erhält man  $-\infty$  für einige Werte von  $\lambda$ . Daher schränken wir die Definitionsmenge von  $q(\lambda)$  entsprechend ein:

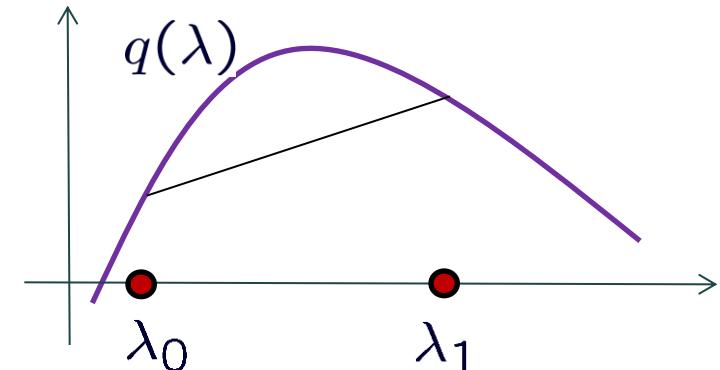
$$\mathcal{D} = \{\lambda | q(\lambda) > -\infty\}$$

- Anmerkung:** 1.) Wir betrachten nur Ungleichheitsbed.  
\*) eigentlich sollten wir inf statt min betrachten

- Die Funktion

$$q(\lambda) = \min_x \mathcal{L}(x, \lambda)$$

ist konkav.



- Beweis Schritt 1: Wir zeigen

$$\mathcal{L}(x, (1-\alpha)\lambda_0 + \alpha\lambda_1) = (1-\alpha)\mathcal{L}(x, \lambda_0) + \alpha\mathcal{L}(x, \lambda_1)$$

- Teilbeweis aus def. Lagrange:

$$\begin{aligned}
 \mathcal{L}(x, (1 - \alpha)\lambda_0 + \alpha\lambda_1) &= f(x) + (1 - \alpha)\lambda_0 c(x) + \alpha\lambda_1 c(x) \\
 &= (1 - \alpha)f(x) + \alpha f(x) + (1 - \alpha)\lambda_0 c(x) + \alpha\lambda_1 c(x) \\
 &= (1 - \alpha)\mathcal{L}(x, \lambda) + \alpha\mathcal{L}(x, \lambda)
 \end{aligned}$$

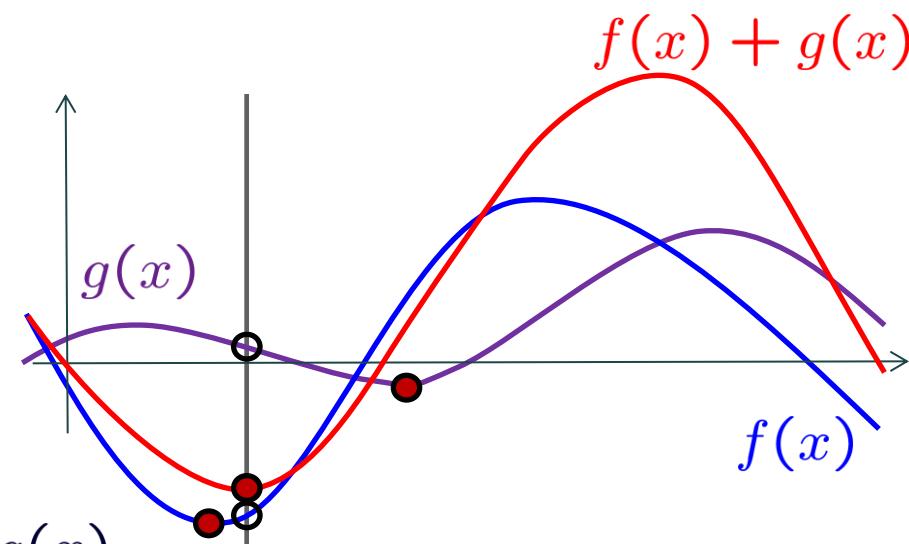
- Damit haben wir:

$$\mathcal{L}(x, (1-\alpha)\lambda_0 + \alpha\lambda_1) = (1-\alpha)\mathcal{L}(x, \lambda_0) + \alpha\mathcal{L}(x, \lambda_1)$$

- Beweis (fort.)
- Das Infimum einer **Summe** ist größer gleich der Summe von Infima, daher:

$$q((1 - \alpha)\lambda_0 + \alpha\lambda_1) \geq (1 - \alpha)q(\lambda_0) + \alpha q(\lambda_1)$$

- Damit ist Konkavität bewiesen



$$\inf(f(x) + g(x)) \geq \inf f(x) + \inf g(x)$$

- Aus dem primalen Optimierungsproblem

$$\min_x f(x), \quad c_i(x) \leq 0$$

können wir also ein **duales Optimierungsproblem** ableiten

$$\max_{\lambda} \min_x \mathcal{L}(x, \lambda), \quad \lambda \geq 0$$

- In manchen Fällen lässt sich das duale Problem leichter lösen als das primale Problem.
- Sogenannte Primal-Dual-Verfahren optimieren gleichzeitig das primale und das duale Problem.
- Es gibt noch andere Formen der Dualität, z.B. die Fenchel Dualität.

- **Schwache Dualität:**

Für alle gültigen Lösungen  $x$  und  $\lambda$  gilt  $q(\lambda) \leq f(x)$

**Beweis:**  $q(\lambda) = \inf_x f(x) + \lambda^\top c(x) \leq f(x) + \underbrace{\lambda^\top c(x)}_{\leq 0} \leq f(x)$

d.h. das duale Problem liefert immer eine untere Schranke für das primale Problem

- **Starke Dualität:**

- $f(x)$  sei konvex und die gültige Menge sei eine konvexe Menge
- $\hat{\lambda}$  bezeichne das Optimum von  $q(\lambda)$  mit  $\hat{x} = \operatorname{arginf}_x \mathcal{L}(x, \hat{\lambda})$
- $\mathcal{L}(x, \hat{\lambda})$  sei streu konvex

Dann gilt:

$$q(\hat{\lambda}) = \mathcal{L}(\hat{x}, \hat{\lambda}) = f(\hat{x})$$

d.h. das Optimum des dualen Problems minimiert auch das primale Problem.

- Konvexes Beispielproblem:

$$\min_{x_1, x_2} \frac{1}{2}(x_1^2 + x_2^2), \quad 1 - x_1 \leq 0$$

- Lagrange-Funktion:

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

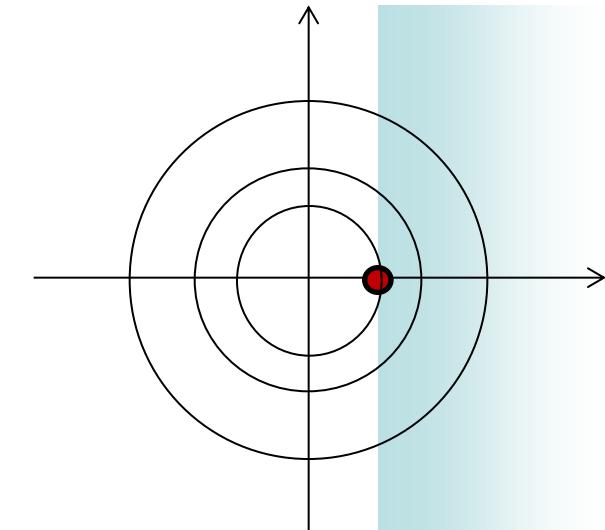
- Bedingungen für ein Minimum:

$$x_1 - \lambda = 0, \quad x_2 = 0,$$

- Damit lässt sich  $x_1, x_2$  in der Lagrange-Funktion eliminieren (geht nicht immer analytisch). Es bleibt das duale Problem

$$\max q(\lambda) = \max -\frac{1}{2}\lambda^2 + \lambda, \quad \lambda \geq 0$$

mit der Lösung  $\lambda = 1$



- Lösung des primalen Problems (starke Dualität):  $x_1 = \lambda = 1, x_2 = 0$

- **Lineare Programme**

$$\min_x w^\top x, \quad Ax - b \geq 0$$

- **Quadratische Programme**

$$\min_x \frac{1}{2} x^\top Q x + w^\top x, \quad Ax - b \geq 0$$

mit  $Q$  symmetrisch und positiv definit

- Optimalitätsbedingungen für Probleme mit Nebenbedingungen unterscheiden ob eine Bedingung aktiv oder inaktiv ist.
- Beide Fälle lassen sich mithilfe der Lagrange-Funktion ausdrücken.
- Über die Lagrange-Funktion lässt sich auch ein duales Problem herleiten.
- Die Lösung des dualen Problems bildet mindestens eine untere Schranke für das ursprüngliche Problem (schwache Dualität).
- Für konvexe Funktionen mit konvexen Nebenbedingungen führt die Lösung des dualen Problems zur exakten Lösung des ursprünglichen Problems (starke Dualität).

- Optimierungsproblem:

$$\min_x (x_1 - 3)^2 + (x_2 - 1)^2$$

$$1 - x_1 - x_2 \geq 0 \quad 1 + x_1 - x_2 \geq 0$$

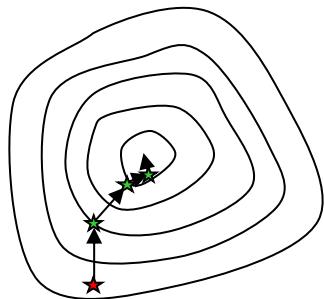
$$1 - x_1 + x_2 \geq 0 \quad 1 + x_1 + x_2 \geq 0$$

- Machen Sie eine Skizze des erlaubten Suchraums.
- Berechnen Sie für verschiedene Punkte des Suchraums den Gradienten der Funktion und tragen Sie ihn in die Skizze ein.
- Wo befindet sich das Minimum?
- Wie sehen die Lagrange Multiplikatoren im Optimum aus? Welche Nebenbedingungen sind im Optimum aktiv, welche nicht?

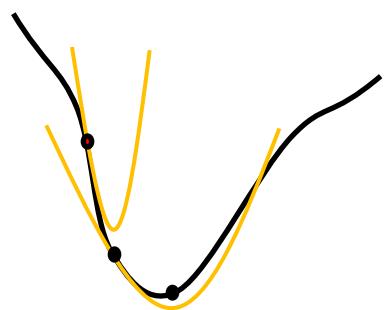
# Optimierung

---

Vorlesung 4  
Optimierung mit Nebenbedingungen



Gradientenabstieg  
(allgemeines Verfahren für kontinuierliche Probleme)



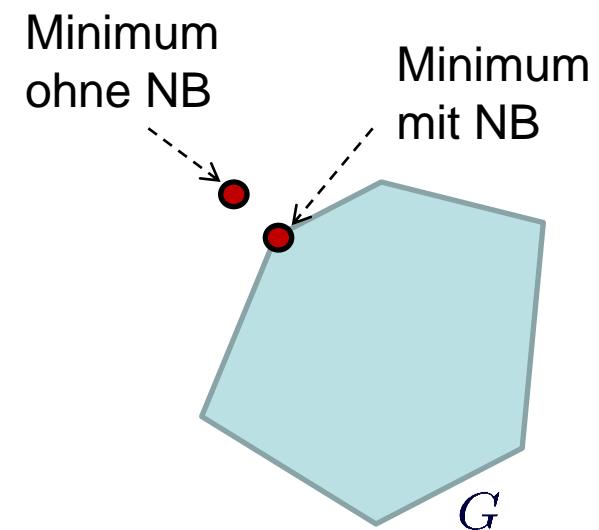
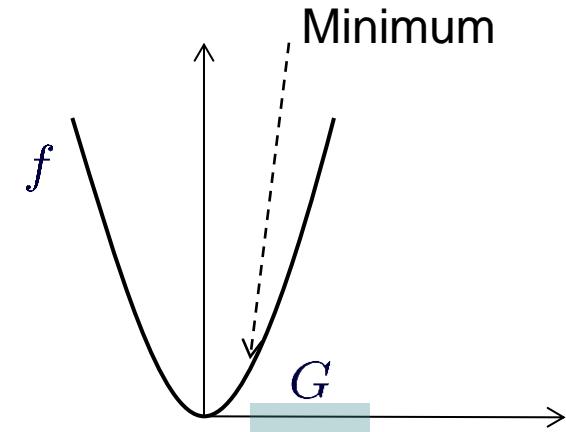
Newton-Verfahren  
(schnelle Konvergenz durch Ausnutzen der Krümmung)

$$d^k = -Q^k \nabla f(x^k)$$

Quasi-Newton-Verfahren (BFGS)  
(effiziente Approximation der Krümmung)

# Nebenbedingungen

- Bisher war der Suchraum für ein Optimum der gesamte Raum  $\mathbb{R}^n$
- In vielen Fällen ist der Suchraum  $G \subset \mathbb{R}^n$  eingeschränkt durch Nebenbedingungen, z.B.  
 $x \leq 5$   
 $x \geq 2$
- Die Lösung des Problems ohne Nebenbedingung ist immer eine **untere Schranke** für das Problem mit Nebenbedingung:  
 $f(x \in \mathbb{R}^n) \leq f(x \in G)$
- Oft (aber nicht immer) liegt das Optimum auf dem Rand von  $G$ .

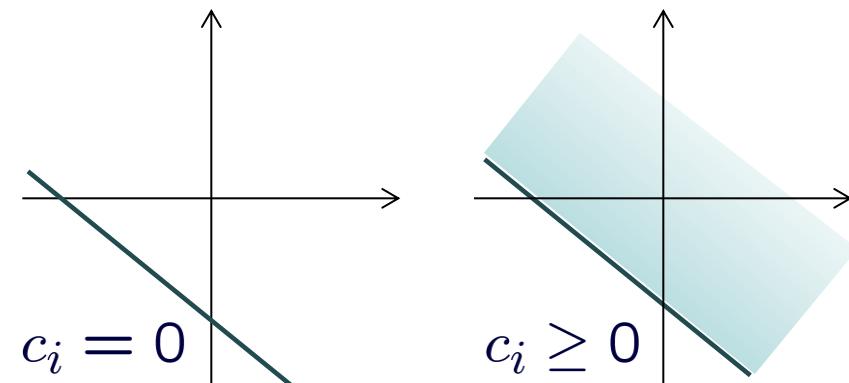


- Jedes Optimierungsproblem lässt sich allgemein schreiben als:

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0, & i \in \mathcal{E} \\ c_i(x) \geq 0, & i \in \mathcal{I} \end{cases}$$

- Zwei Arten von Bedingungen:
  - Gleichheitsbedingungen
  - Ungleichheitsbedingungen
- Jede Bedingung lässt sich in diese Form bringen, z.B.

$$5x_1 + 3x_2 \leq 2 \quad \rightarrow \quad 2 - 5x_1 - 3x_2 \geq 0$$



# Beispiel mit einer Gleichheitsbedingung

- Das Problem:  $f(x) = x_1 + x_2$      $c(x) = x_1^2 + x_2^2 - 2 = 0$
- Gradienten von  $\nabla f$  und  $\nabla c$
- Schritt  $d$  erlaubt falls  $\nabla c^T d = 0$  und  $\nabla f^T d < 0$   
also Abstieg in  $f$  und keine Änderung in  $c$

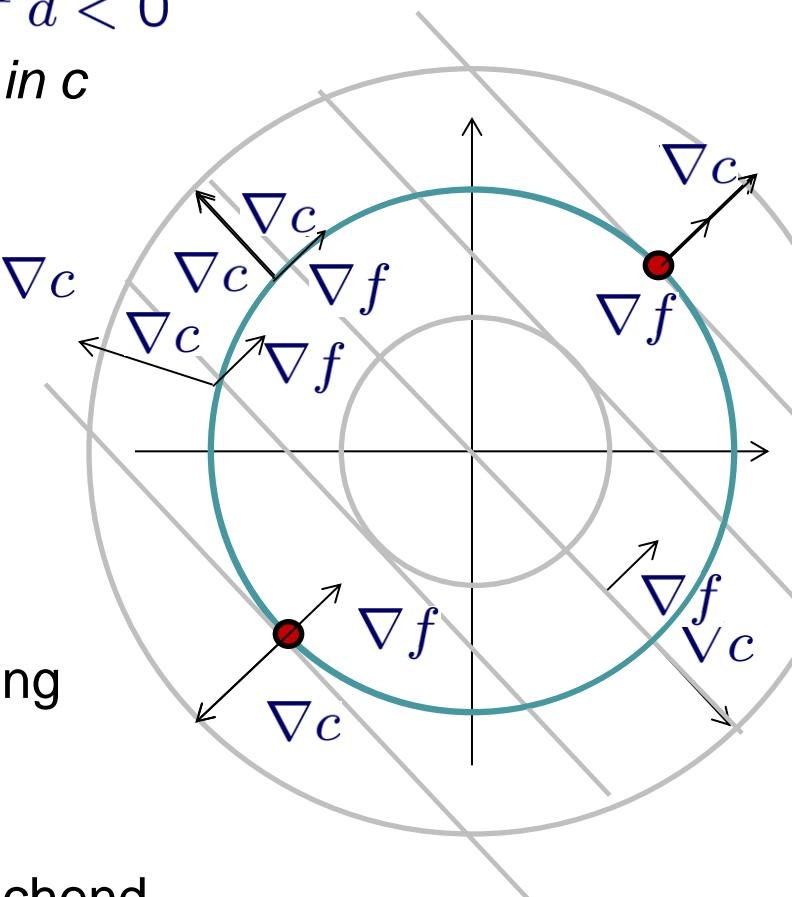
- Im Minimum sind die Gradienten  $\nabla f$  und  $\nabla c$  offenbar parallel, d.h.

$$\nabla f(x) = \lambda \nabla c(x)$$

für einen bestimmten skalaren Wert

- In der Tat ist dies eine notwendige Bedingung für ein Minimum

- Offensichtlich ist die Bedingung nicht hinreichend, denn sie ist auch im Maximum erfüllt



- Wir können die notwendige Bedingung für ein Minimum mithilfe der **Lagrange Funktion** ausdrücken

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

- Die Bedingung lautet dann

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

für eine geeignete Wahl von  $\lambda$

- Man bezeichnet  $\lambda$  auch als **Lagrange Multiplikator**
- In manchen Fällen nimmt dieser auch den Wert 0 an.  
Was bedeutet das?

## Nebenbedingung ohne Relevanz

- Betrachten wir das Problem

$$f(x) = (x_1 + x_2)^2$$

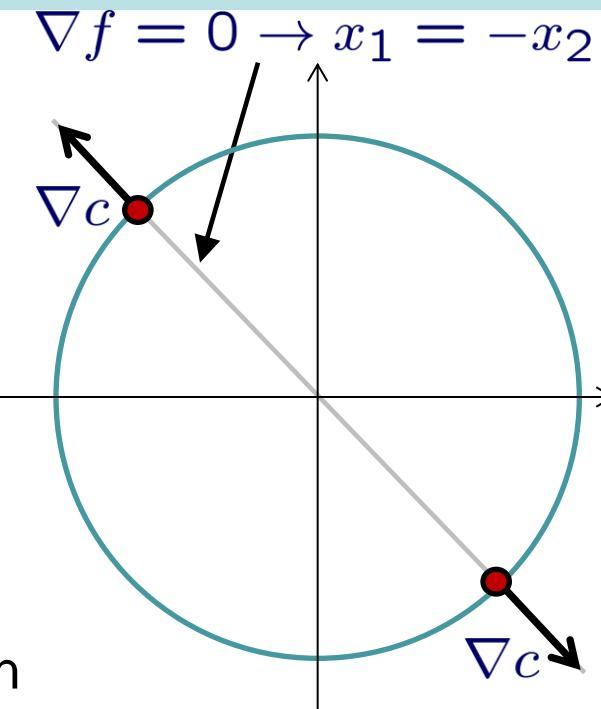
$$c(x) = x_1^2 + x_2^2 - 2 = 0$$

- Wir haben zwei (gleich gute) Minima.
- In beiden Fällen ist  $\nabla f = 0$ , d.h. unabhängig von  $c$  kann  $f$  lokal nicht weiter minimiert werden

- Die notwendige Bedingung

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

wird für  $\lambda = 0$  erfüllt.



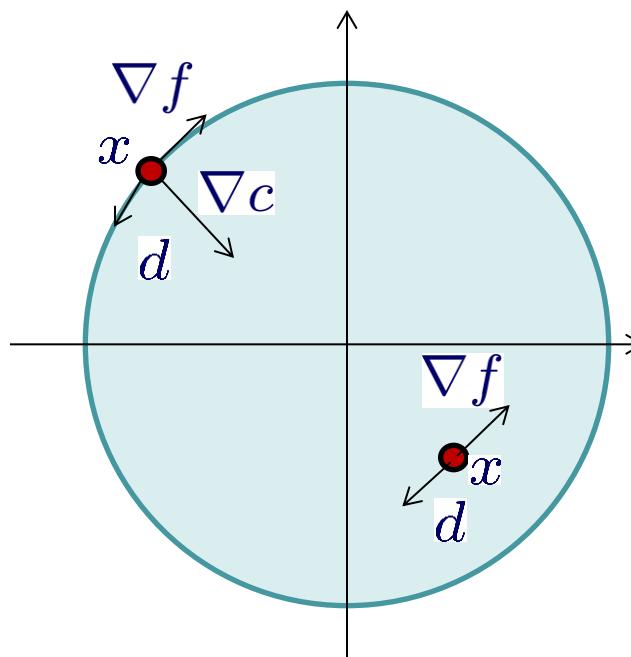
## Eine Ungleichheitsbedingung

- Das Problem:

$$f(x) = x_1 + x_2$$

$$c(x) = 2 - x_1^2 - x_2^2 \geq 0$$

- Wir betrachten einen Punkt  $x$  und fragen uns wann ein kleiner Schritt  $d$  zulässig ist.



## Ungleichheitsbedingung I: Minimum am Rand

- Das Problem:

$$f(x) = x_1 + x_2$$

$$c(x) = 2 - x_1^2 - x_2^2 \geq 0$$

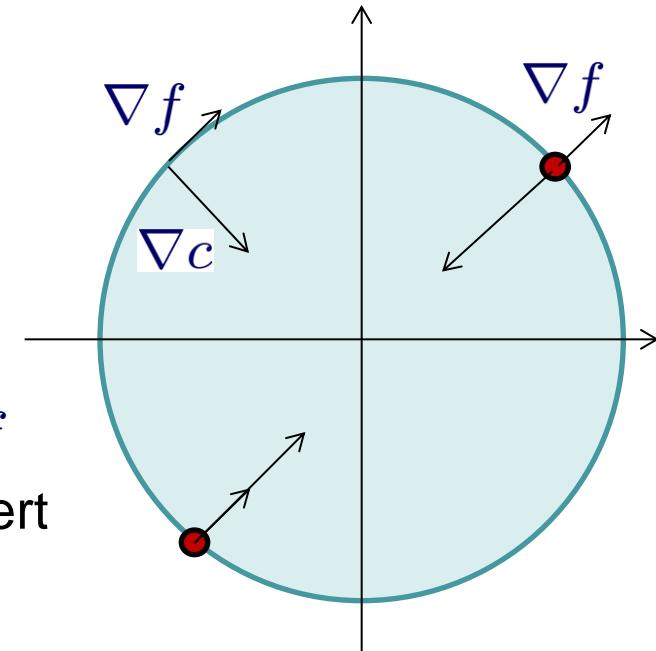
- Minimum am Rand: wir wollen in Richtung  $-\nabla f$  aber ein weiterer Abstieg wird durch  $c$  verhindert

$$c(x) = 0$$

- Die Gradienten  $\nabla f$  und  $\nabla c$  zeigen im Minimum notwendigerweise in die gleiche Richtung

$$\nabla f(x) = \lambda \nabla c(x), \quad \lambda \geq 0$$

- Das Vorzeichen von  $\lambda$  spielt diesmal eine Rolle. Würden  $\nabla f$  und  $\nabla c$  in entgegengesetzte Richtung zeigen, könnte man  $f$  in Richtung von  $\nabla c$  weiter minimieren.



## Beispiel mit einer Ungleichheitsbedingung

- Neues Problem:

$$f(x) = x_1^2 + x_2^2$$

$$c(x) = 2 - x_1^2 - x_2^2 \geq 0$$

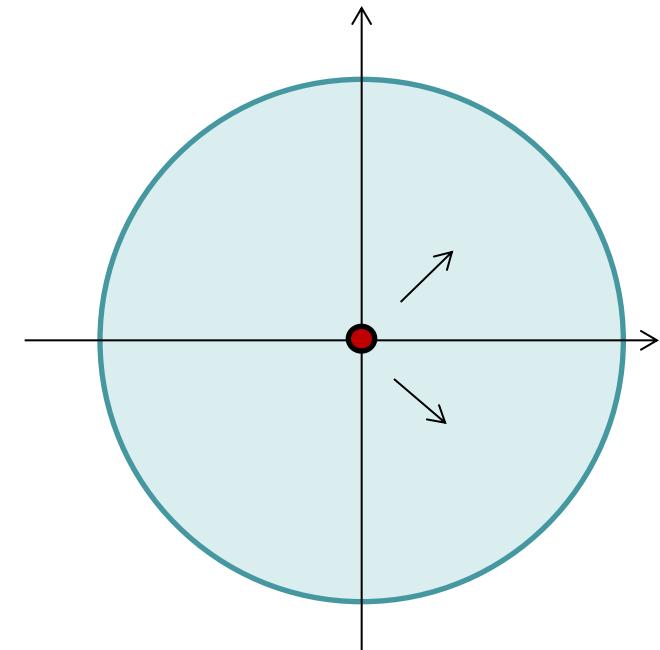
- Das Minimum liegt nun im Inneren von  $G$ , d.h.  $c$  hat keinen Einfluss und  $\nabla f(x) = 0$

- Die Bedingung

$$\nabla f(x) = \lambda \nabla c(x), \quad \lambda \geq 0$$

ist für  $\lambda = 0$  erfüllt.

- Die Nebenbedingung ist hier im Minimum **inaktiv**.



- Wir können allgemein wieder die Lagrange Funktion

$$\mathcal{L}(x, \lambda) = f(x) - \lambda c(x), \quad \lambda \geq 0$$

verwenden und erhalten die **Optimalitätsbedingung**

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

- Zusätzlich muss die **Komplementaritätsbedingung** gelten:

$$\lambda c(x) = 0$$

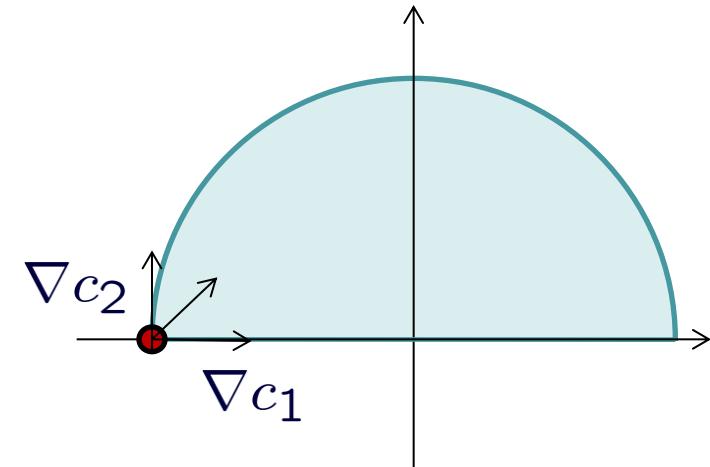
- Entweder liegt das Optimum auf dem Rand (dann ist  $c(x) = 0$ ) oder  $c$  ist im Optimum inaktiv (dann ist  $\lambda = 0$ ) oder beides.
- Sämtliche Fälle sind also sehr elegant über die Lagrange Funktion und die Komplementaritätsbedingung abgedeckt.
- **Strenge Komplementarität** liegt vor wenn  $c$  und  $\lambda$  nicht gleichzeitig null sind

- Problem mit zwei Bedingungen:

$$f(x) = x_1 + x_2$$

$$c_1(x) = 2 - x_1^2 - x_2^2 \geq 0$$

$$c_2(x) = x_2 \geq 0$$



- In diesem Fall sind beide Nebenbedingungen im Minimum aktiv

$$\nabla f(x) - \lambda_1 \nabla c_1(x) - \lambda_2 \nabla c_2(x) = 0, \quad \lambda_1 \geq 0, \lambda_2 \geq 0$$

$$\text{d.h. } c_1(x) = c_2(x) = 0, \quad \lambda_1 > 0, \lambda_2 > 0$$

- Die Menge der aktiven Nebenbedingungen wird als die **aktive Menge** (active set) bezeichnet

- Einige Nebenbedingungen können überflüssig sein, da sie bereits von anderen Nebenbedingungen abgedeckt sind, z.B.:

$$c_1(x) = x_2 \geq 0$$

$$c_2(x) = x_2 \geq 2$$

- $c_2$  deckt automatisch auch  $c_1$  ab.  $c_1$  ist daher nie aktiv.
- Da die Komplexität von der Anzahl der Nebenbedingungen abhängt, ist es meist ratsam redundante Bedingungen vorab zu eliminieren.

- Zusammenfassend sind die notwendigen Bedingungen für ein lokales Optimum:

$$\nabla_x \mathcal{L}(x, \lambda) = 0$$

$$c_i(x) = 0, \quad \forall i \in \mathcal{E}$$

$$c_i(x) \geq 0, \quad \forall i \in \mathcal{I}$$

$$\lambda_i \geq 0, \quad \forall i \in \mathcal{I}$$

$$\lambda_i c_i(x) = 0, \quad \forall i \in \mathcal{E} \cup \mathcal{I}$$

- Diese werden als Karush-Kuhn-Tucker (KKT) Bedingungen bezeichnet.
- Im folgenden werden wir alle Bedingungen  $c_i(x)$  in einen Funktionsvektor  $c(x) := (c_1(x), \dots, c_m(x))^T$  und entsprechend  $\lambda \in \mathbb{R}^m$  zusammenfassen.

- Durch die Nebenbedingungen kamen neben den primalen Variablen  $x$  die Lagrange Multiplikatoren  $\lambda$  als Variablen ins Spiel

$$\mathcal{L}(x, \lambda) = f(x) - \lambda^\top c(x), \quad \lambda \geq 0$$

- Sie geben an wie stark die Funktion gegen die Bedingungen drückt.

$$\nabla f(x) = \lambda^\top \nabla c(x), \quad \lambda \geq 0$$

- Alternativ können wir  $\mathcal{L}(x, \lambda)$  als zu minimierende Funktion ohne Nebenbedingungen betrachten, indem wir  $\lambda$  festhalten

$$q(\lambda) = \inf_x \mathcal{L}(x, \lambda)$$

- In vielen Fällen erhält man  $-\infty$  für einige Werte von  $\lambda$ . Daher schränken wir die Definitionsmenge von  $q(\lambda)$  entsprechend ein:

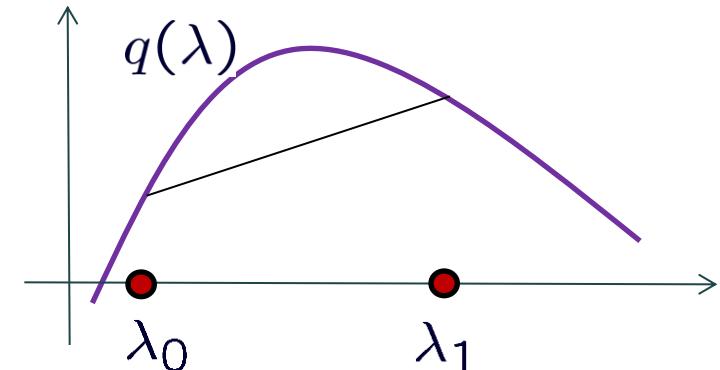
$$\mathcal{D} = \{\lambda | q(\lambda) > -\infty\}$$

- **Anmerkung:** Wir betrachten nur Ungleichheitsbed.

- Die Funktion

$$q(\lambda) = \inf_x \mathcal{L}(x, \lambda)$$

ist konkav.



- Beweis Schritt 1: Wir zeigen

$$\mathcal{L}(x, (1-\alpha)\lambda_0 + \alpha\lambda_1) = (1-\alpha)\mathcal{L}(x, \lambda_0) + \alpha\mathcal{L}(x, \lambda_1)$$

- Teilbeweis aus def. Lagrange:

$$\begin{aligned}
 \mathcal{L}(x, (1 - \alpha)\lambda_0 + \alpha\lambda_1) &= f(x) - (1 - \alpha)\lambda_0 c(x) - \alpha\lambda_1 c(x) \\
 &= (1 - \alpha)f(x) + \alpha f(x) - (1 - \alpha)\lambda_0 c(x) - \alpha\lambda_1 c(x) \\
 &= (1 - \alpha)\mathcal{L}(x, \lambda) + \alpha\mathcal{L}(x, \lambda)
 \end{aligned}$$

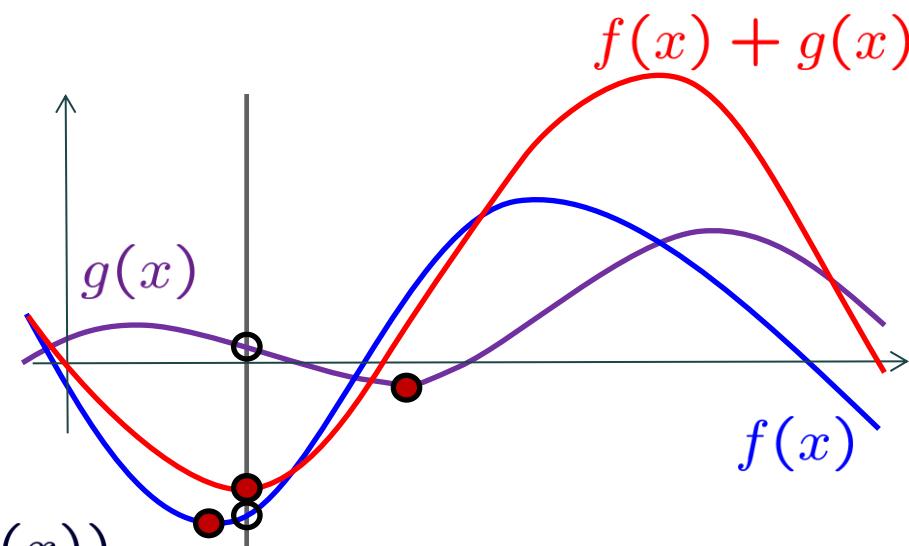
- Damit haben wir:

$$\mathcal{L}(x, (1-\alpha)\lambda_0 + \alpha\lambda_1) = (1-\alpha)\mathcal{L}(x, \lambda_0) + \alpha\mathcal{L}(x, \lambda_1)$$

- Beweis (fort.)
- Das Infimum einer **Summe** ist größer gleich der Summe von Infima, daher:

$$q((1 - \alpha)\lambda_0 + \alpha\lambda_1) \geq (1 - \alpha)q(\lambda_0) + \alpha q(\lambda_1)$$

- Damit ist Konkavität bewiesen  
(siehe Abb. oben)



$$\inf f(x) + \inf g(x) \geq \inf(f(x) + g(x))$$

- Aus dem primalen Optimierungsproblem

$$\min_x f(x), \quad c(x) \geq 0$$

können wir also ein **duales Optimierungsproblem** ableiten

$$\max_{\lambda} \inf_x \mathcal{L}(x, \lambda), \quad \lambda \geq 0$$

- In manchen Fällen lässt sich das duale Problem leichter lösen als das primale Problem.
- Sogenannte Primal-Dual-Verfahren optimieren gleichzeitig das primale und das duale Problem.
- Es gibt noch andere Formen der Dualität, z.B. die Fenchel Dualität.

- **Schwache Dualität:**

Für alle gültigen Lösungen  $x$  und  $\lambda$  gilt  $q(\lambda) \leq f(x)$

**Beweis:**  $q(\lambda) = \inf_x f(x) - \lambda^\top c(x) \leq f(x) - \underbrace{\lambda^\top c(x)}_{\geq 0} \leq f(x)$

d.h. das duale Problem liefert immer eine untere Schranke für das primale Problem

- **Starke Dualität:**

- $f(x)$  sei konvex und die gültige Menge sei eine konvexe Menge
- $\hat{\lambda}$  bezeichne das Optimum von  $q(\lambda)$  mit  $\hat{x} = \operatorname{arginf}_x \mathcal{L}(x, \hat{\lambda})$
- $\mathcal{L}(x, \hat{\lambda})$  sei streu konvex

Dann gilt:

$$q(\hat{\lambda}) = \mathcal{L}(\hat{x}, \hat{\lambda}) = f(\hat{x})$$

d.h. das Optimum des dualen Problems minimiert auch das primale Problem.

- Konvexes Beispielproblem:

$$\min_{x_1, x_2} \frac{1}{2}(x_1^2 + x_2^2), \quad x_1 - 1 \geq 0$$

- Lagrange-Funktion:

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

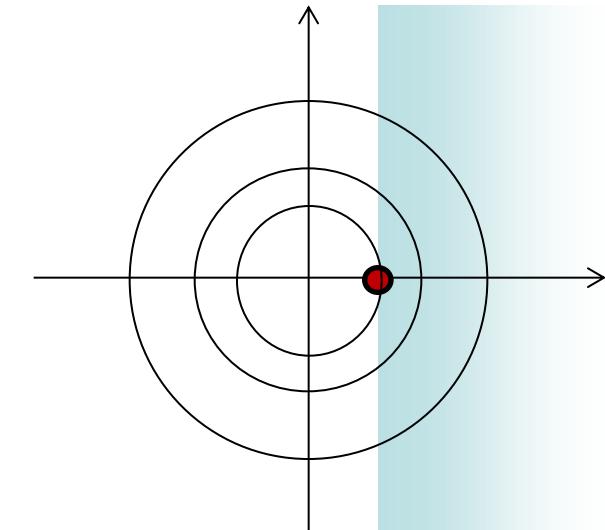
- Bedingungen für ein Minimum:

$$x_1 - \lambda = 0, \quad x_2 = 0,$$

- Damit lässt sich  $x_1, x_2$  in der Lagrange-Funktion eliminieren (geht nicht immer analytisch). Es bleibt das duale Problem

$$\max q(\lambda) = \max -\frac{1}{2}\lambda^2 + \lambda, \quad \lambda \geq 0$$

mit der Lösung  $\lambda = 1$



- Lösung des primalen Problems (starke Dualität):  $x_1 = \lambda = 1, x_2 = 0$

- **Lineare Programme**

$$\min_x w^\top x, \quad Ax - b \geq 0$$

Duales Problem:

$$\max_\lambda b^\top \lambda, \quad A^\top \lambda = w, \quad \lambda \geq 0$$

- **Quadratische Programme**

$$\min_x \frac{1}{2} x^\top Q x + w^\top x, \quad Ax - b \geq 0$$

mit  $Q$  symmetrisch und positiv definit

Duales Problem:

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

- Optimalitätsbedingungen für Probleme mit Nebenbedingungen unterscheiden ob eine Bedingung aktiv oder inaktiv ist.
- Beide Fälle lassen sich mithilfe der Lagrange-Funktion ausdrücken.
- Über die Lagrange-Funktion lässt sich auch ein duales Problem herleiten.
- Die Lösung des dualen Problems bildet mindestens eine untere Schranke für das ursprüngliche Problem (schwache Dualität).
- Für konvexe Funktionen mit konvexen Nebenbedingungen führt die Lösung des dualen Problems zur exakten Lösung des ursprünglichen Problems (starke Dualität).

- Optimierungsproblem:

$$\min_x (x_1 - 3)^2 + (x_2 - 1)^2$$

$$1 - x_1 - x_2 \geq 0 \quad 1 + x_1 - x_2 \geq 0$$

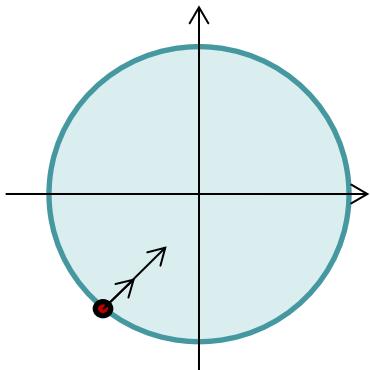
$$1 - x_1 + x_2 \geq 0 \quad 1 + x_1 + x_2 \geq 0$$

- Machen Sie eine Skizze des erlaubten Suchraums.
- Berechnen Sie für verschiedene Punkte des Suchraums den Gradienten der Funktion und tragen Sie ihn in die Skizze ein.
- Wo befindet sich das Minimum?
- Wie sehen die Lagrange Multiplikatoren im Optimum aus? Welche Nebenbedingungen sind im Optimum aktiv, welche nicht?

# Optimierung

---

Vorlesung 5  
Lineare Programmierung



## Optimierung mit Nebenbedingungen

$$\nabla_x \mathcal{L}(x, \lambda) = 0$$

$$c_i(x) = 0, \quad \forall i \in \mathcal{E}$$

$$c_i(x) \geq 0, \quad \forall i \in \mathcal{I}$$

$$\lambda_i \geq 0, \quad \forall i \in \mathcal{I}$$

$$\lambda_i c_i(x) = 0, \quad \forall i \in \mathcal{E} \cup \mathcal{I}$$

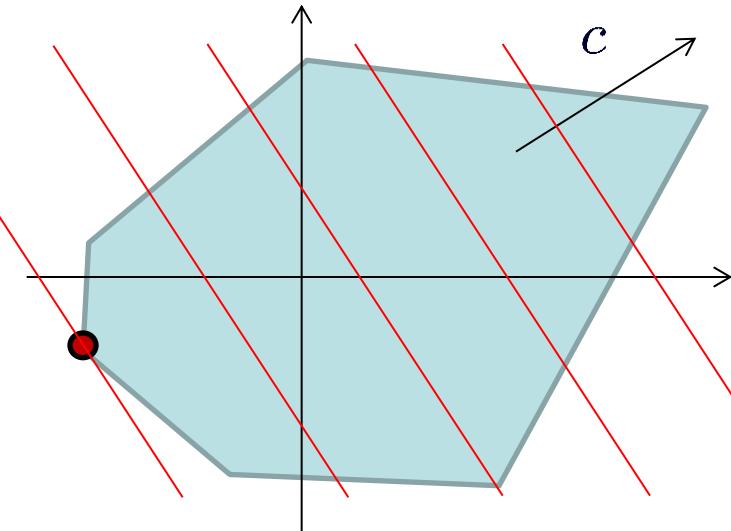
KKT Bedingungen für ein Minimum

$$q(\hat{\lambda}) = \mathcal{L}(\hat{x}, \hat{\lambda}) = f(\hat{x}) \quad \text{Lagrange Dualität}$$

- Der einfachste Fall von Optimierung mit Nebenbedingung sind lineare Programme:

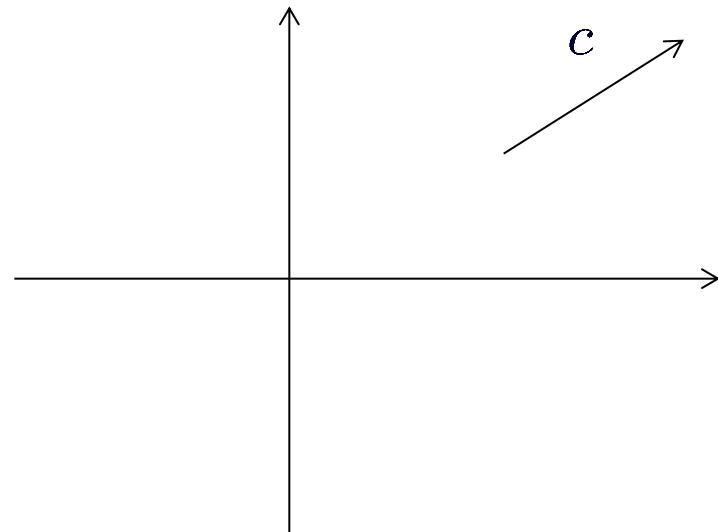
$$\min_x c^T x, \quad Ax - b \geq 0$$

- Sowohl die Zielfunktion  $f(x)$  ist linear als auch alle Nebenbedingungen.
- Das Problem ist offensichtlich konvex und es gilt daher strenge Dualität (mehr dazu später)
- Die Nebenbedingungen spannen einen konvexen Polyeder auf. Falls ein Minimum existiert, liegt es auf dem Rand des Polyeders.



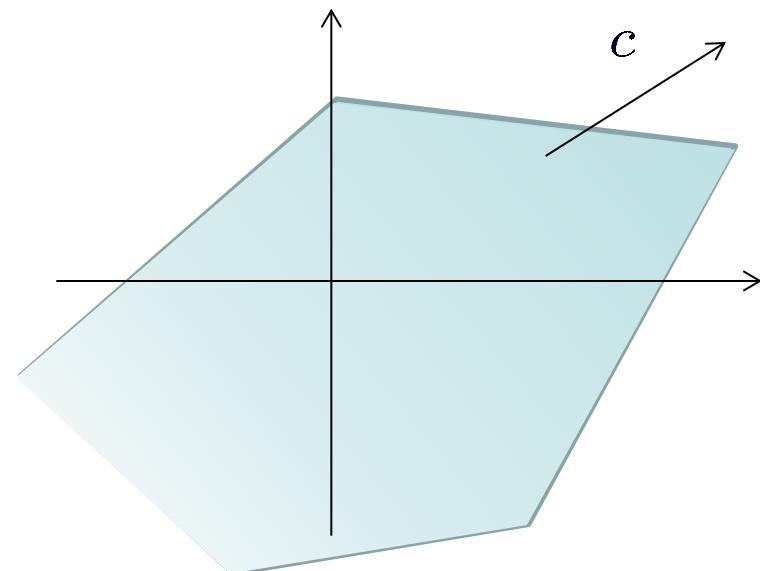
1. Ein lineares Programm ist **ungültig**, wenn der zulässige Polyeder die leere Menge darstellt.

D.h. einige Nebenbedingungen schließen sich gegenseitig aus.



2. Ein lineares Programm ist **unbeschränkt**, wenn der Polyeder in negativer Richtung von  $c$  offen ist.

Die Zielfunktion kann dann beliebig kleine Werte annehmen.



- Bedingt durch das Simplex-Verfahren zur Optimierung linearer Programme hat sich die Formulierung in der **Standardform** durchgesetzt:

$$\min_x c^\top x, \quad Ax = b, \quad x \geq 0$$

- Jedes lineare Programm lässt sich in diese Form überführen.
- Um Ungleichheitsbedingungen in die geforderte Gleichheitsbedingung umzuformulieren, bedient man sich sogenannter **Schlupfvariablen**  $\xi$

$$Ax - b \geq 0 \quad \rightarrow \quad Ax - \xi - b = 0, \quad \xi \geq 0$$

- Durch Aufteilen von  $x = x^+ - x^-$  in den negativen und den positiven Teil, lässt sich das Problem in die Standardform erweitern

$$\min_x \begin{pmatrix} c \\ -c \\ 0 \end{pmatrix}^\top \begin{pmatrix} x^+ \\ x^- \\ \xi \end{pmatrix}, \quad (A \quad -A \quad -I) \begin{pmatrix} x^+ \\ x^- \\ \xi \end{pmatrix} = b, \quad \boxed{\begin{pmatrix} x^+ \\ x^- \\ \xi \end{pmatrix}} \geq 0$$

$x$

- Lagrange-Funktion

$$\mathcal{L}(x, \lambda, s) = c^\top x - \lambda^\top (Ax - b) - s^\top x$$

mit separaten Lagrange-Multiplikatoren  $\lambda$  und  $s$

- Karush-Kuhn-Tucker Bedingungen (siehe letzte Vorlesung):

$$A^\top \lambda + s = c \quad (\nabla_x \mathcal{L} = 0)$$

$$Ax = b$$

$$x \geq 0$$

$$s \geq 0$$

$$x^\top s = 0 \quad (\text{Komplementaritätsbedingung})$$

- Da das Problem konvex ist, sind diese notwendigen Bedingungen auch hinreichend.

- Das duale Problem lautet

$$\max_{\lambda} b^T \lambda, \quad A^T \lambda + s = c, \quad s \geq 0$$

- Für lineare Programme ist die Beziehung zwischen dem primalen und dem dualen Problem besonders stark ausgeprägt.
- Unter Ausnutzung der KKT Bedingungen können wir z.B. im Optimum folgende Umformung vornehmen:

$$\begin{aligned} c^T x &= (A^T \lambda + s)^T x && \text{(Verwendung der 1. KKT Bedingung)} \\ &= (Ax)^T \lambda + s^T x \\ &= (Ax)^T \lambda && \text{(verwende } x^T s = 0 \text{ )} \\ &= b^T \lambda && \text{(verwende } Ax = b \text{ )} \end{aligned}$$

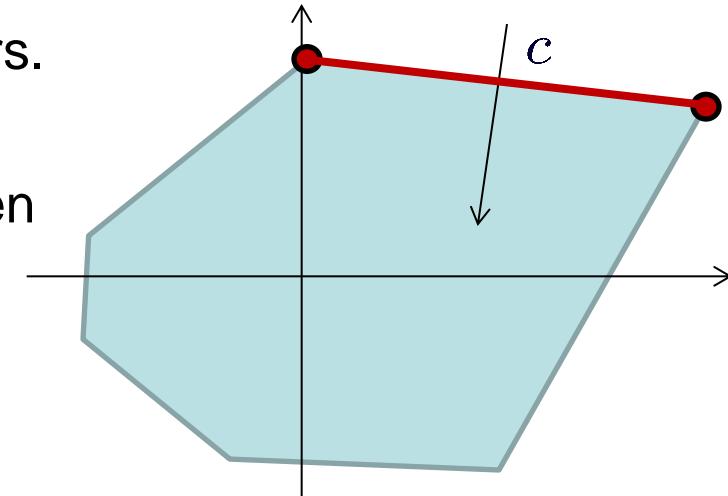
d.h. die Zielfunktionen nehmen im Optimum den gleichen Wert an  
(starke Dualität)

- Starke Dualität ist keine Überraschung, da lineare Programme die entsprechenden Bedingungen aus der letzten Vorlesung erfüllen.
- Es können folgende Aussagen über die Lösbarkeit gemacht werden:
  - Wenn eines der Probleme lösbar ist, ist auch das andere lösbar (direkte Konsequenz der starken Dualität)
  - Wenn das eine Problem unbeschränkt ist, ist das andere ungültig.
- Außerdem ist die Primal-Dual-Beziehung bei linearen Programmen symmetrisch:

Das duale Problem des dualen Problems ist wieder das primale Problem.

- Entsprechend sind die optimalen Variablen des einen Problems gerade die optimalen Lagrange-Multiplikatoren des anderen Problems.

- Optima liegen immer auf Ecken des Polyeders.



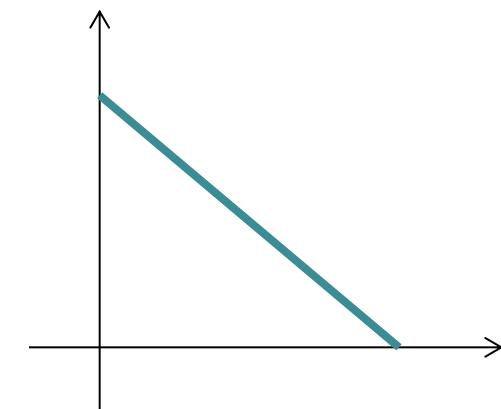
- Mehrere Optima sind möglich. Optimale Ecken sind dabei immer in der Menge der Optima enthalten.

- Grundlage des Simplex-Verfahrens:  
Ausgehend von einer Ecke, gehe zu einer benachbarten Ecke mit niedrigerer Energie.

- Vorsicht! Die bildliche Anschauung ändert sich durch Verwendung der Standardform.

- Polyeder → Polytop in höherdimensionalem Raum

Jede Schlupfvariable  $\xi$  vergrößert die Dimension des Lösungsraums.



- Lagrange-Funktion enthält die primalen und dualen Variablen

$$\mathcal{L}(x, \lambda, s) = c^\top x - \lambda^\top (Ax - b) - s^\top x \quad x, s \in \mathbb{R}^n, \lambda \in \mathbb{R}^m$$

- Von zentraler Bedeutung ist die Matrix  $A \in \mathbb{R}^{m \times n}$

- Anzahl der Zeilen  $m$ : Anzahl der Gleichheitsbedingungen

- Anzahl Spalten  $n$ : Dimension von  $x$  (inkl. Schlupfvariablen)

- Annahme:  $n > m$  und der Rang von  $A$  ist  $m$

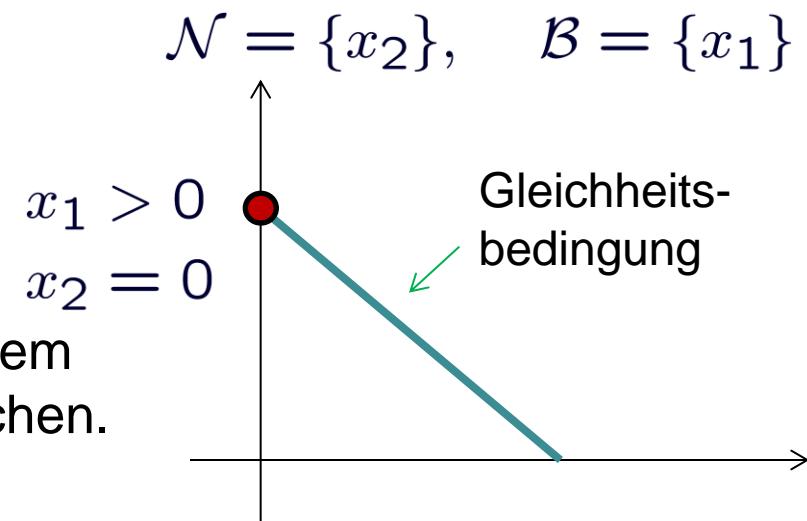
- Ist dies nicht erfüllt, heißt das:

- Die zulässige Menge  $G$  ist ein einzelner Punkt oder leer (warum?)

oder

- Nebenbedingungen sind linear abhängig. Dem können wir durch Eliminierung entsprechender Gleichungen abhelfen.

- Das Simplex-Verfahren gehört zu den **Active-Set-Verfahren**.
- In jedem Schritt teilt es die Variablen  $x_i$  in die **inaktive Menge (Basis)  $\mathcal{B}$**  und die **aktive Menge  $\mathcal{N} = \{1, \dots, n\} - \mathcal{B}$**  auf.
- Die Basis  $\mathcal{B}$  enthält immer genau  $m$  Indizes. Für die entsprechenden  $x_i$  muss die Gleichheitsbedingung  $Ax = b$  erfüllt sein.
- Aus  $i \in \mathcal{N}$  folgt hingegen  $x_i = 0$  (die Bedingung  $x_i \geq 0$  ist aktiv)
- Die entsprechende Lösung liegt immer auf den Ecken des Polytops und nennt sich **Basislösung**.
- Wir suchen die optimale Basislösung, indem wir Indizes zwischen  $\mathcal{B}$  und  $\mathcal{N}$  austauschen.



- Die Mengen  $\mathcal{B}$  und  $\mathcal{N}$  zerlegen entsprechend auch die Matrix  $A$  (Beispiel)

$$x_3 = 0 \quad x_5 = 0$$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{pmatrix}$$

- Wir erhalten eine invertierbare Matrix  $B = \begin{pmatrix} a_{11} & a_{12} & a_{14} \\ a_{21} & a_{22} & a_{24} \\ a_{31} & a_{32} & a_{34} \end{pmatrix}$   
sowie eine Matrix  $N = \begin{pmatrix} a_{13} & a_{15} \\ a_{23} & a_{25} \\ a_{33} & a_{35} \end{pmatrix}$

- Bestimmung der primalen Variablen  $x = (x_B, x_N)$

Die Gleichheitsbedingungen müssen erfüllt sein:

$$Ax = Bx_B + Nx_N = b$$

$$x_N = 0 \quad \text{daher} \quad x_B = B^{-1}b$$

- Bestimmung der dualen Variablen  $\lambda, s = (s_B, s_N)$

Wir haben  $s_B = 0$  (Komplementarität)

Aus der KKT-Bedingung  $A^\top \lambda + s = c$  erhalten wir durch Aufteilung

$$B^\top \lambda = c_B \quad \text{und} \quad N^\top \lambda + s_N = c_N$$

Damit erhalten wir:

$$\lambda = B^{-\top} c_B \quad s_N = c_N - N^\top \lambda$$

## Welche der KKT-Bedingungen sind erfüllt?

- Die KKT-Bedingungen sind nur im Optimum alle erfüllt. Noch sind wir nicht im Optimum. Welche der Bedingungen sind also verletzt?

$$A^\top \lambda + s = c$$

$$Ax = b$$

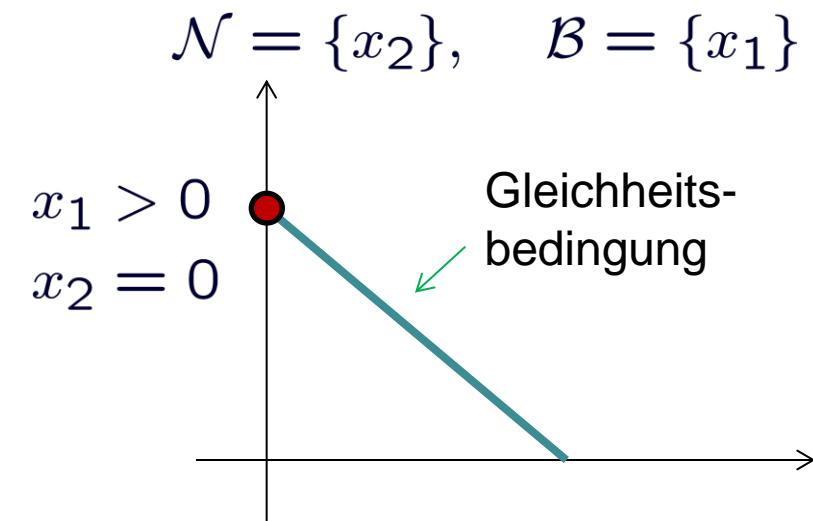
$x \geq 0$  (Start mit Basislösung, daher per Definition erfüllt)

$$s \geq 0$$

$$x^\top s = 0$$

- Falls alle  $s_N \geq 0$ , sind alle Optimalitätsbedingungen erfüllt  
→ Abbruchkriterium
- Andernfalls ist jeder Index  $j \in \mathcal{N}$  für den  $s_j < 0$  ein Kandidat, um einen verbesserten Eckpunkt zu generieren.
- Einfachste Strategie: wähle  $q$ , so dass  $s_q$  minimal ist.

- Wir kennen nun also den Index  $q \in \mathcal{N}$ , den wir in die Basis neu aufnehmen möchten.
- Da die Basis immer aus genau  $m$  Elementen besteht, muss ein anderer Index  $p \in \mathcal{B}$  nach  $\mathcal{N}$  wechseln.
- Um herauszufinden welcher, erhöhen wir  $x_q$  (das bisher 0 war) unter Einhaltung der Gleichheitsbedingungen.
- Dabei erreicht irgendwann ein  $x_i, i \in \mathcal{B}$  den Wert 0.  
→ Setze  $p = i$  und entferne das Element aus der Basis
- Falls dies nicht passiert, ist das Problem unbeschränkt.



1. Finde eine initiale Basislösung
2. Berechne alle primalen und dualen Variablen für diese Basislösung
3. Solange nicht alle  $s_N \geq 0$ :
  - Wähle  $q$  mit  $s_q < 0$
  - Erhöhe  $x_q$  unter Einhaltung von  $Ax = b$  bis  $x_p = 0$  für ein  $p \in \mathcal{B}$  (Pivoting)
  - Entferne  $p$  aus der Basis
  - Update aller Variablen für die neue Basis
- Noch offen:
  - Wie finden wir eine initiale Basislösung?
  - Wie funktioniert Pivoting genau?
  - Wie berechnen wir ein effizientes Update von  $B^{-1}$ ?
  - Was passiert im Fall einer **degenerierten Basis** ( $x_i = s_i = 0$ ,  $i \in \mathcal{B}$ )?

- Um eine initiale Basislösung (Ecke) zu finden, müssen wir ein (einfacheres) lineares Programm lösen:

$$\min_z e^T z, \quad Ax + Ez = b, \quad x \geq 0, \quad z \geq 0$$

mit der Diagonalmatrix  $E$  mit Einträgen  $E_{ii} = \text{sign } b_i$  und  $e := (1, \dots, 1)^T$

- Eine Basislösung für dieses lineare Programm ist einfach zu sehen:

$$x = 0, \quad z_i = |b_i|$$

d.h.  $x$  wird zunächst auf 0 gesetzt und die Gleichheitsbedingung mithilfe der neuen Variablen  $z_i$  erfüllt.

- Indem die Summe der  $z_i$  minimiert wird (Minimum bei 0), muss sich wieder  $x$  um die Gleichheitsbedingung kümmern.
- Im Minimum haben wir eine Basislösung für das eigentliche Problem.

- $Ax = b$  muss weiter eingehalten werden, d.h. beim Erhöhen von  $x_q$  müssen einige der  $x_i, i \in \mathcal{B}$  geeignet reduziert werden.
- Vor und nach der Änderung muss gelten:

$$Ax = Bx_B = Bx'_B + \boxed{A_q x'_q} \quad q\text{te Spalte von } A$$

- Durch Umformung erhalten wir also die notwendige Änderung von  $x_B$  beim Erhöhen von  $x_q$ :

$$x'_B = x_B - B^{-1} A_q x'_q \quad d := B^{-1} A_q$$

- Mit dieser Information finden wir das kritische Element  $p$

$$p = \operatorname{argmin}_{i \in \mathcal{B}} \frac{x_i}{d_i}$$

sowie  $x'_q = \frac{x_p}{d_p}$

- Die aufwendigste Operation ist das Invertieren der Matrix  $B$  in den Operationen

$$B^\top \lambda = c_B \quad Bd = A_q$$

- Nach dem Update der Basis möchten wir  $B$  nicht erneut invertieren.
- Besser: Faktorisierung von  $B$  in einer untere und obere Dreiecksmatrix:  
$$B = LU$$
- $L$  und  $U$  können nach dem Basiswechsel effizient aktualisiert werden  
(siehe Nocedal-Wright pp. 372)

- Wir gingen bisher immer davon aus, dass  $x_i > 0, \forall i \in \mathcal{B}$
- Es kann aber auch sein, dass  $x_i = 0, i \in \mathcal{B}$
- Eine Basis mit einem solchen Element heißt **degeneriert**. Das Gesamtproblem heißt degeneriert, wenn es eine degenerierte Basis gibt.
- Problem: Beim Austausch eines degenerierten Basiselements wird die Zielfunktion möglicherweise nicht reduziert, nämlich dann wenn beim Pivoting  $x_i = 0$  und  $d_i \neq 0$ .
- Der Basiswechsel sorgt dafür, dass das Verfahren dennoch terminiert, denn mit der anderen Basis gilt eventuell  $x_i = 0$  und  $d_i = 0$ .
- Es muss jedoch sichergestellt werden, dass das Verfahren nicht zweimal die gleiche degenerierte Basis wählt (Zyklus).

- Das Simplex-Verfahren ist in fast allen Fällen sehr effizient (ca.  $2m$  Iterationen)
- Die Worst-Case-Komplexität ist jedoch exponentiell in der Anzahl der Dimensionen  $n$
- Dies gilt auch für alle anderen Varianten des Algorithmus, die bisher bekannt sind.
- **Innere-Punkt-Methoden** liefern bessere Garantien und sind bei sehr großen Problemen üblicherweise effizienter.
- Bei kleinen und mittelgroßen linearen Programmen ist der Simplex-Algorithmus im Normalfall die beste Wahl.

- Lineare Programme lassen sich effizient mit dem Simplex-Algorithmus optimieren.
- Mindestens eines der globalen Optima ist eine Ecke des zulässigen Polyeders.
- Das Simplex-Verfahren wandert die Ecken des Polyeders ab, bis es die optimale Ecke gefunden hat.
- Die primalen und dualen Variablen werden dabei gleichzeitig optimiert.
- Das Verfahren unterscheidet zwischen der aktiven und inaktiven Menge der Ungleichheitsbedingungen.
- Ein Simplexschritt tauscht jeweils zwei Elemente zwischen diesen beiden Mengen aus.

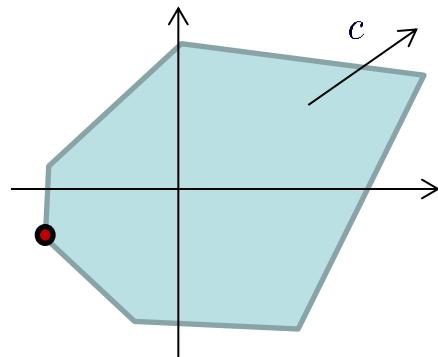
- J. Nocedal, S. J. Wright: Numerical Optimization, Springer, 2006.  
Kapitel 13 behandelt den Simplex-Algorithmus im Detail.

1. Ein Server hat einen Plattendurchsatz von 100MB/s, eine Netzwerkkapazität von 1000MBit/s und einen Hauptspeicher mit 8000MB. Er soll Jobs dreier verschiedener Arten verarbeiten. Jobs vom Typ 1 haben die höchste Priorität von 20 und benötigen 8MB/s Plattendurchsatz, 50MBit/s vom Netzwerk und 50MB Speicher (wir unterscheiden hier nicht zwischen mittlerer und Peakauslastung). Von Typ 1 stehen 10 in der Queue. Die 5 Jobs vom Typ 2 haben Priorität 8 und benötigen 5MB/s, 2MBit/s und 800MB. Außerdem gibt es noch 50 Jobs vom Typ 3 mit Priorität 2, welche 2MB/s Platte und 40MB Speicher benötigen. Alle Jobs dauern gleich lange. Wir möchten die Jobs so auswählen, dass die Summe der Prioritäten maximal wird. Formulieren Sie die Aufgabe als lineares Programm.
2. Bringen Sie die Formulierung in die Standardform.
3. Lösen Sie das Problem mit Scipy `linprog` unter Verwendung des Simplex-Algorithmus.

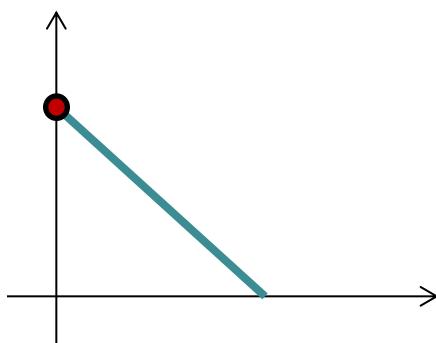
# Optimierung

---

Vorlesung 6  
Quadratische Programmierung



Lineare Programmierung



Simplex-Algorithmus

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \end{pmatrix}$$

Active-Set-Verfahren

- Lineares Programm:

$$\min_x c^\top x, \quad Ax - b \geq 0$$

- Quadratisches Programm:

$$\min_x \frac{1}{2} x^\top Q x + c^\top x$$

$$a_i^\top x = b_i, \quad i \in \mathcal{E}$$

$$a_i^\top x \geq b_i, \quad i \in \mathcal{I}$$

→ quadratische Zielfunktion  $f(x)$  mit linearen Nebenbedingungen

- Die  $n \times n$  Matrix  $Q$  ist symmetrisch und offensichtlich die Hesse-Matrix von  $f(x)$

# Konvexe vs. nicht-konvexe quadratische Programme

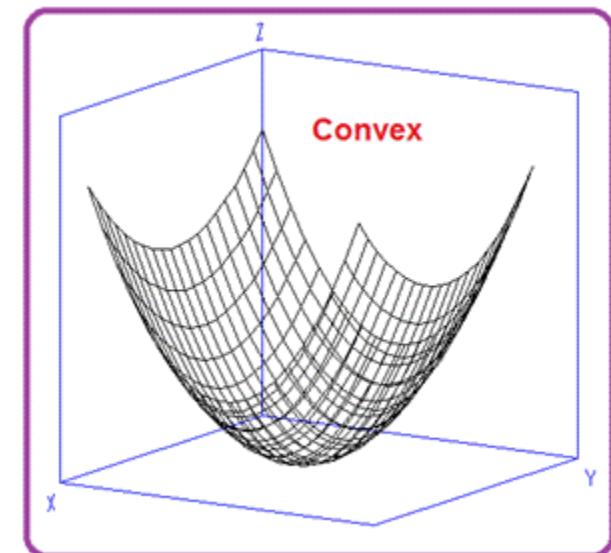
$$\min_x \frac{1}{2} x^\top Q x + c^\top x$$

- Wenn  $Q$  positiv semi-definit ist, ist das quadratische Programm konvex.

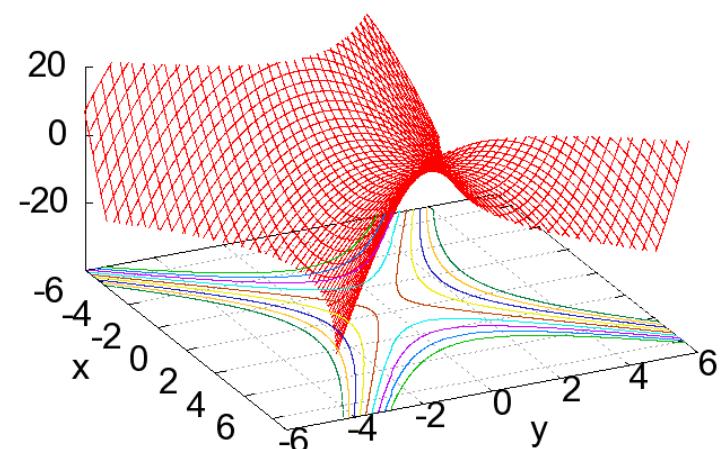
Jedes  $x$ , das die KKT Bedingungen erfüllt, ist ein globales Minimum.

- Ist  $Q$  positiv definit, ist das globale Minimum sogar eindeutig.
- Wenn  $Q$  indefinit ist, ist das quadratische Programm nicht konvex.

Es gibt stationäre Punkte, welche die KKT Bedingungen erfüllen, aber kein Minimum sind.



$$f(x,y) = x^2 - 2y^2 + xy + x + y + 1$$



Quelle: Google Images

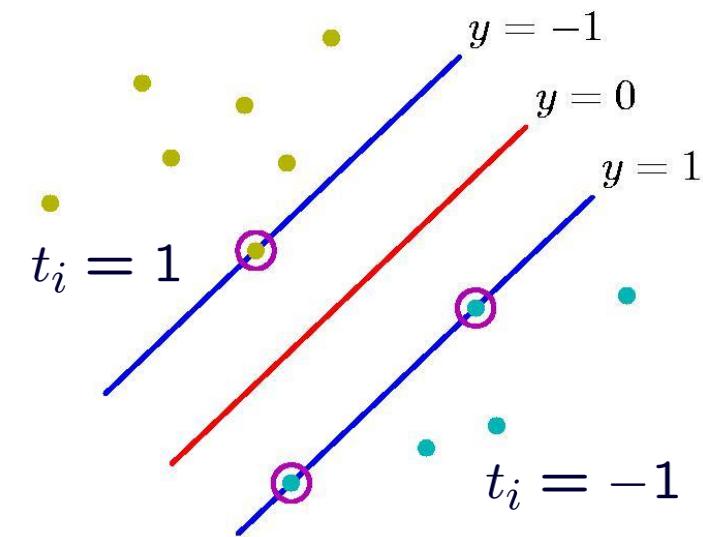
- Gegeben: Datenpunkte  $a_i \in \mathbb{R}^n$  mit Label  $t_i \in \{-1, 1\}$
- Ziel: Finden der einfachsten Hyperebene

$$y = a^\top x + b$$

so dass alle Datenpunkte auf der richtigen Seite der Hyperebene liegen

$$\text{sign}(y(a_i)) = t_i$$

und einen Mindestabstand von 1 zur Hyperebene haben.



Quelle: Christopher Bishop

- Quadratisches Programm:

$$\min_x x^\top x \quad (\text{Einfachheit der Hyperebene})$$

$$t_i(a_i^\top x + b) \geq 1, \quad \forall i \quad (\text{Vorzeichen und Mindestabstand zur Hyperebene})$$

- Das Optimierungsproblem ist hier offensichtlich konvex

$$\min_x x^\top x \quad t_i(a_i^\top x + b) \geq 1, \quad \forall i$$

da die Hesse Matrix  $Q$  die Einheitsmatrix ist.

→ eindeutiges globales Minimum

- Die Nebenbedingungen sind meist redundant

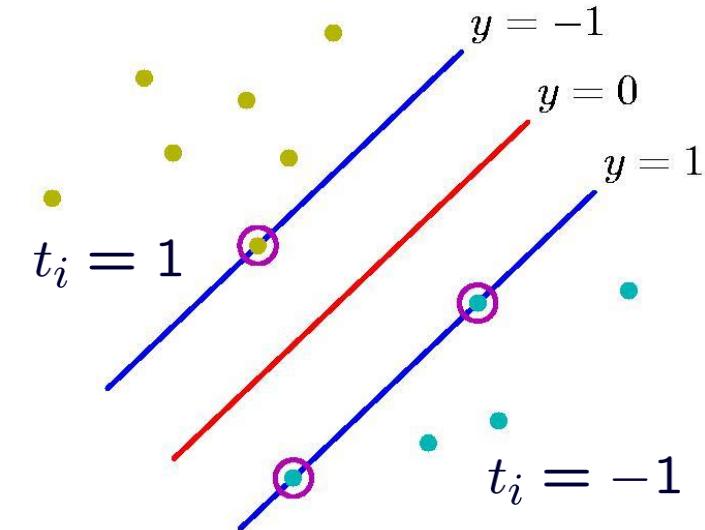
→ Reduktion der Nebenbedingungen

- In der Praxis:

– Millionen Datenpunkte, Reduktion relativ teuer

– Duales Problem hat spezielle Form mit sehr einfachen Nebenbedingungen

Daher einfaches iteratives Verfahren (Projected Coordinate Descent)  
effizienter als das folgende (allgemeinere) Verfahren zur quadratischen  
Programmierung  
→ Projektionsmethoden in der nächsten Vorlesung



Quelle: Christopher Bishop

- In dieser Vorlesung nur streng konvexe quadratische Programme
- Außerdem betrachten wir zunächst nur Probleme mit Gleichheitsbedingungen:

$$\min_x \frac{1}{2} x^\top Q x + c^\top x, \quad Ax = b$$

Annahme:  $A$  hat vollen Rang (Reduktion der Nebenbedingungen)

- Lösung hilft uns beim Lösen des (schwierigeren) Problems mit Ungleichheitsbedingungen.
- Optimalitätsbedingungen (KKT Bedingungen):

$$Qx - A^\top \lambda = -c$$

$$Ax = b$$

- KKT System in Matrixform:

$$\begin{pmatrix} Q & -A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} -c \\ b \end{pmatrix}$$

- Da  $Q$  positiv definit ist, müssen wir nur dieses Gleichungssystem lösen, z.B. effizient mit der **Schur-Komplement-Methode**:

- Multipliziere erste Gleichung mit  $AQ^{-1}$

$$\cancel{AQ^{-1}}Qx - AQ^{-1}A^\top\lambda = -AQ^{-1}c$$

- Ziehe zweite Gleichung von erster Gleichung ab

$$-AQ^{-1}A^\top\lambda = -AQ^{-1}c - b$$

- Löse Gleichungssystem in  $\lambda$

$$AQ^{-1}A^\top\lambda = AQ^{-1}c + b$$

- Löse Gleichungssystem in  $x$

$$Qx = A^\top\lambda - c$$

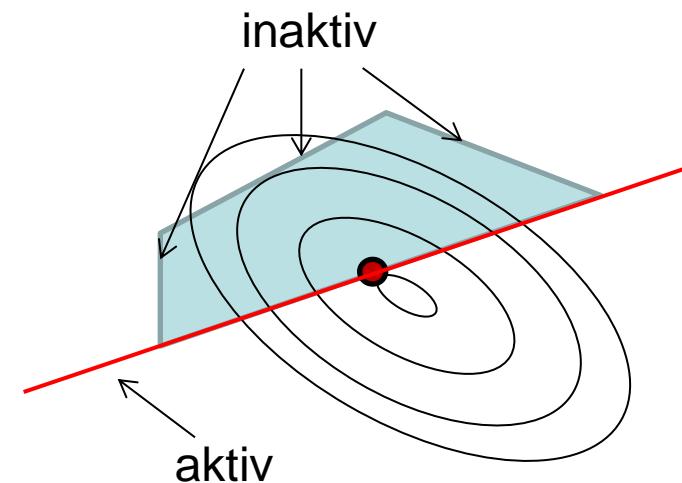
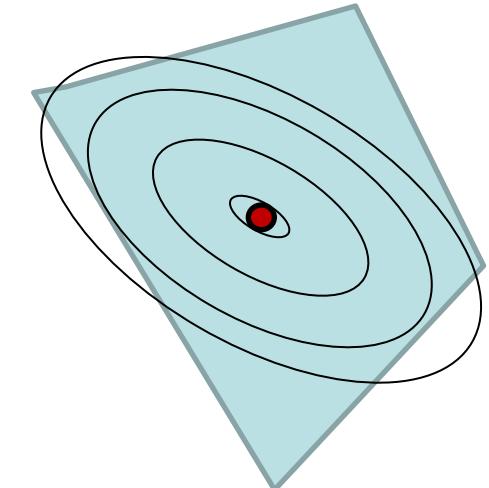
- Es gibt verschiedene Ansätze ein quadratisches Programm mit Gleichheitsbedingungen zu lösen, die je nach Problemstellung effizienter oder allgemeiner einsetzbar sind als die Schur-Komplement-Methode.

Beispiele:

- **Symmetrisch Indefinite Faktorisierung:** eine spezielle Faktorisierungsmethode, die auch mit indefiniten Matrizen umgehen kann
  - Eliminierung von Variablen anhand der Gleichheitsbedingungen und Optimierung (ohne Nebenbedingungen) im verbleibenden Nullraum (**Null-Space-Methode**)
- Das Grundproblem ist nicht nur für die nun folgenden quadratischen Programme mit Ungleichheitsbedingungen wichtig sondern auch für allgemeine nichtlineare Programme.  
→ Sequential Quadratic Programming in der nächsten Vorlesung

$$\min_x \frac{1}{2}x^\top Qx + c^\top x, \quad Ax \geq b$$

- Wie bei der linearen Programmierung können wir ein Active-Set-Verfahren verwenden.
- Unterschied: Die optimale Lösung muss nicht auf dem Rand der gültigen Menge liegen
- Beobachtung: Wüssten wir welche Nebenbedingungen im Minimum aktiv sind, hätten wir wieder Gleichheitsbedingungen.
- Die Schwierigkeit liegt also darin, die Menge der aktiven Nebenbedingungen zu finden.



- DEFINITION. Geg. Eine Menge von Nebenbedingungen (NB)

$$g_1(x) \geq 0, \dots, g_m(x) \geq 0$$

und ein Punkt  $x^*$  in der zulässigen Region. Eine NB  $g_i(x) \geq 0$  wird **aktiv** genannt falls  $g_i(x^*) = 0$  ist, ansonsten ist sie **inaktiv**.

- BEMERKUNG: Gleichheitsbedingungen sind immer aktiv.
- Bsp Simplex-Verfahren letzte Stunde:  $\min_x c^\top x, \quad Ax = b, \quad Ex \geq 0$  wobei  $E$  die Einheitsmatrix ist. Die aktive Menge  $\mathcal{N}$  bestand aus den Indizes  $i \in \mathcal{N}$  für die  $x_i = 0$  gilt. Damit sind das die i-ten Zeilen von  $Ex \geq 0$  (entspricht i-ter Nebenbedingung).

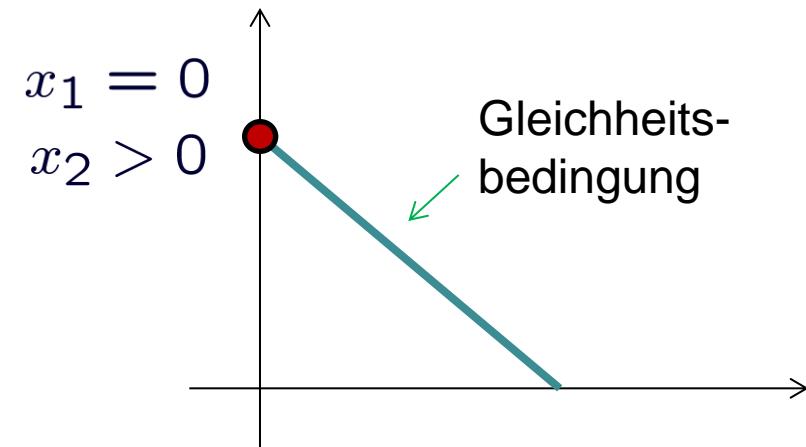
# Unterschied zwischen linearem und quadratischem Programm

- Mithilfe der Standardform erhielten wir beim linearen Programm Gleichheitsbedingungen

$$\min_x c^T x, \quad Ax = b, \quad Ex \geq 0$$

und nur einfache Ungleichheit in  $x$

Die aktive Menge ließ sich durch Austausch der aktiven  $x_i$  (= i-te Zeile von  $Ex \geq 0$ ) steuern.

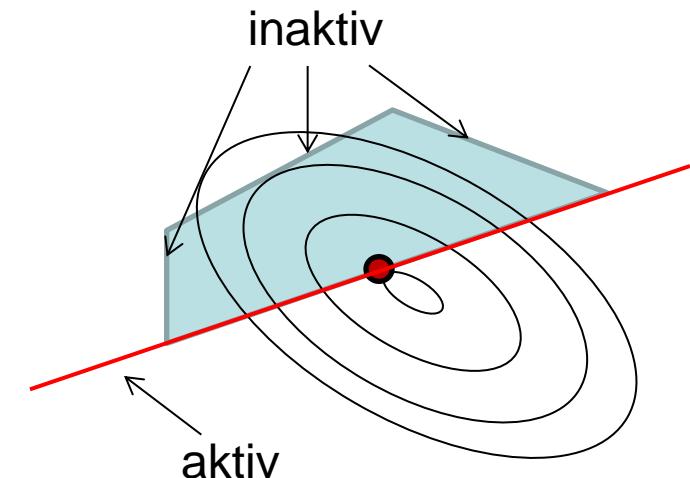


- Das quadratische Programm

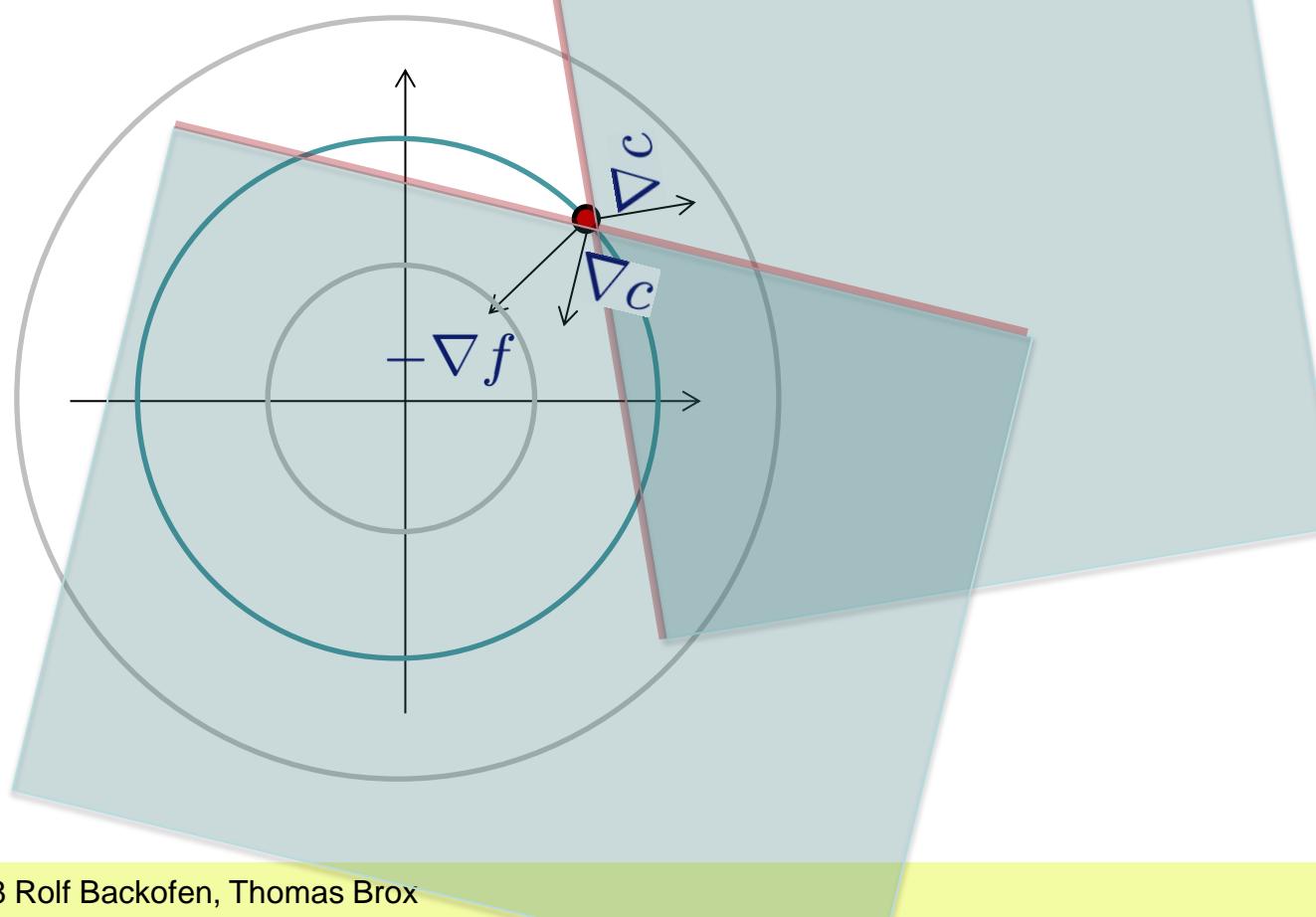
$$\min_x \frac{1}{2} x^T Q x + c^T x, \quad Ax \geq b$$

lässt sich nicht in eine solche Form bringen.

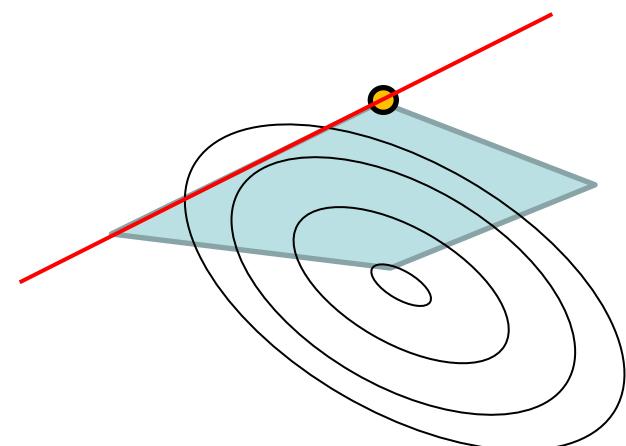
Unbekannt wie viele Bedingungen aktiv sind  
(Lösung muss keine Ecke sein)



- Das Problem:  $f(x) = x_1^2 + x_2^2$
- Gradienten von  $\nabla f$  und  $\nabla c$
- Lagrange Multiplikator negativ: weiter Optimierbar
- Lagrange Multiplikator positiv: Konstraint aktiv.
- Unterschied: spitzer/Stumpfer Winkel



- Um die Ungleichheitsbedingungen in Gleichheitsbedingungen zu konvertieren führen wir eine **Arbeitsmenge**  $\mathcal{W}$  ein.
- In ihr können nur Bedingungen enthalten sein, die in einem aktuellen Punkt  $x^k$  aktiv sind.
- Es müssen jedoch nicht alle aktiven Bedingungen enthalten sein.
- Die Arbeitsmenge kann auch leer sein.
- Die Bedingungen in  $\mathcal{W}$  müssen linear unabhängig sein.
- Die Bedingungen in der Arbeitsmenge werden in  $x^k$  zu Gleichheitsbedingungen -> entspricht Zeilen von  $Ax \geq b$  für die gilt:  
 $a_i^T x = b_i$       ( $a_i$  ist die i-te Zeile von  $A$ )



- Wir können nun einen Schritt  $d$  machen, der  $f(x)$  unter den Gleichheitsbedingungen in  $\mathcal{W}$  minimiert, d.h. wir minimieren

$$\min_x \frac{1}{2}x^\top Qx + c^\top x, \quad a_i^T x = b_i, \quad \forall i \in \mathcal{W}$$

Wir wissen bereits, wie wir dieses Problem lösen können (z.B. mit der Schur-Komplement-Method)

- Das Minimierungsproblem kann auch in  $d$  ausgedrückt werden:

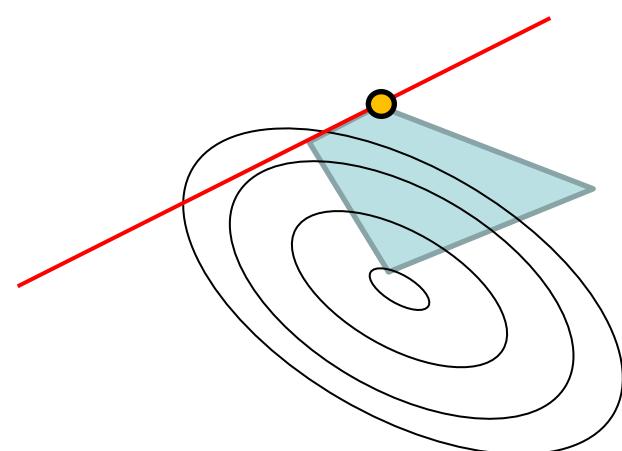
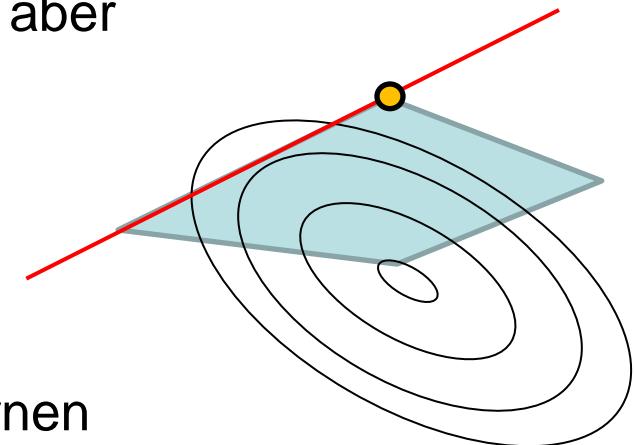
$$\min_d \frac{1}{2}d^\top Qd + g_k^\top d, \quad a_i^T d = 0, \quad \forall i \in \mathcal{W}$$

mit

$$d = x - x^k, \quad g_k = Qx^k + c$$

- Wir können also den optimalen Schritt  $d$  finden, der  $f(x)$  unter den Bedingungen in  $\mathcal{W}$  minimiert.
- Wäre  $\mathcal{W}$  die aktive Menge im Optimum, wären wir nun fertig.

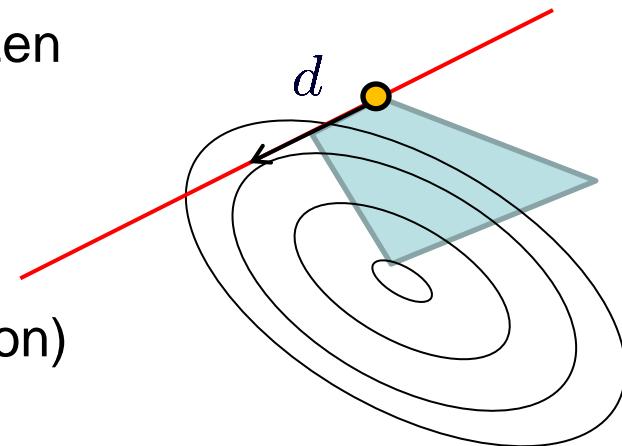
- Der Schritt  $d$  hält die Bedingungen in  $\mathcal{W}$  ein, aber
  1. Diese Bedingungen sollten vielleicht nicht alle aktiv sein
  2. Es ist nicht sichergestellt, dass die anderen Bedingungen eingehalten werden
- Wir müssen also Bedingungen aus  $\mathcal{W}$  entfernen (welche?)
- Außerdem müssen wir sicherstellen, dass der Schritt  $d$  nicht aus der gültigen Menge herausläuft (wie?)
- Die verletzte (**blockierende**) Bedingung wird in die Arbeitsmenge aufgenommen.



- Wir können unseren Schritt in der Länge verkürzen

$$x^{k+1} = x^k + \alpha d$$

- Ohne verletzte Bedingung wäre  $\alpha = 1$  optimal (Newton-Verfahren, quadratische Funktion)



- Bedingungen  $i \notin \mathcal{W}$  mit  $a_i^\top d \geq 0$  sind unkritisch, da in dem Fall

$$a_i^\top (x^k + d) \geq a_i^\top x^k \geq b_i$$

- Wir müssen also  $\alpha$  so wählen, dass auch für alle Bedingungen  $i \notin \mathcal{W}$  mit  $a_i^\top d < 0$  gilt:

$$a_i^\top (x^k + \alpha d) \geq b_i$$

$$\Rightarrow \alpha \leq \frac{b_i - a_i^\top x^k}{a_i^\top d} \quad \Rightarrow \quad \alpha = \min \left( 1, \min_{i \notin \mathcal{W}, a_i^\top d < 0} \left( \frac{b_i - a_i^\top x^k}{a_i^\top d} \right) \right)$$

- In unserem Beispiel können wir offenbar  $f(x)$  mit der aktuellen Menge  $\mathcal{W}$  nicht weiter optimieren

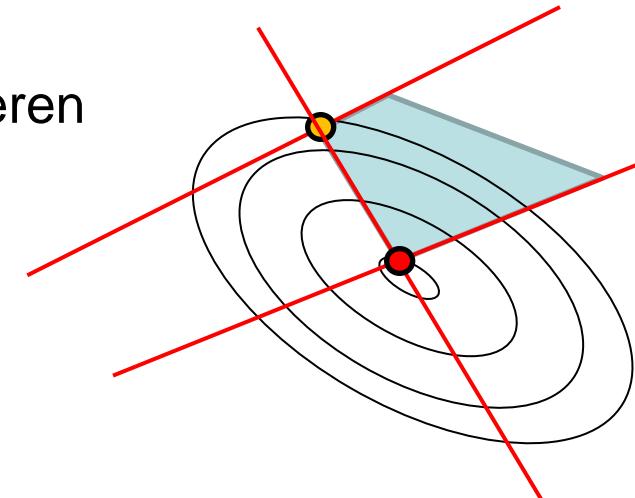
- Sind die KKT-Bedingungen erfüllt?

$$Qx - A^\top \lambda = -c \quad \checkmark$$

$$Ax \geq b \quad \checkmark$$

$$\lambda^\top (Ax - b) = 0 \quad \checkmark$$

$$\lambda \geq 0$$



- Falls  $\lambda \geq 0$  haben wir das Minimum gefunden, ansonsten zeigen die negativen  $\lambda_i$  an, welche Bedingungen aus  $\mathcal{W}$  entfernt werden müssen.  
(siehe Vorlesung 4, Folie 9 zu Lagrange-Mult. bei Ungleichungen)
- Danach lässt sich  $f(x)$  weiter optimieren.

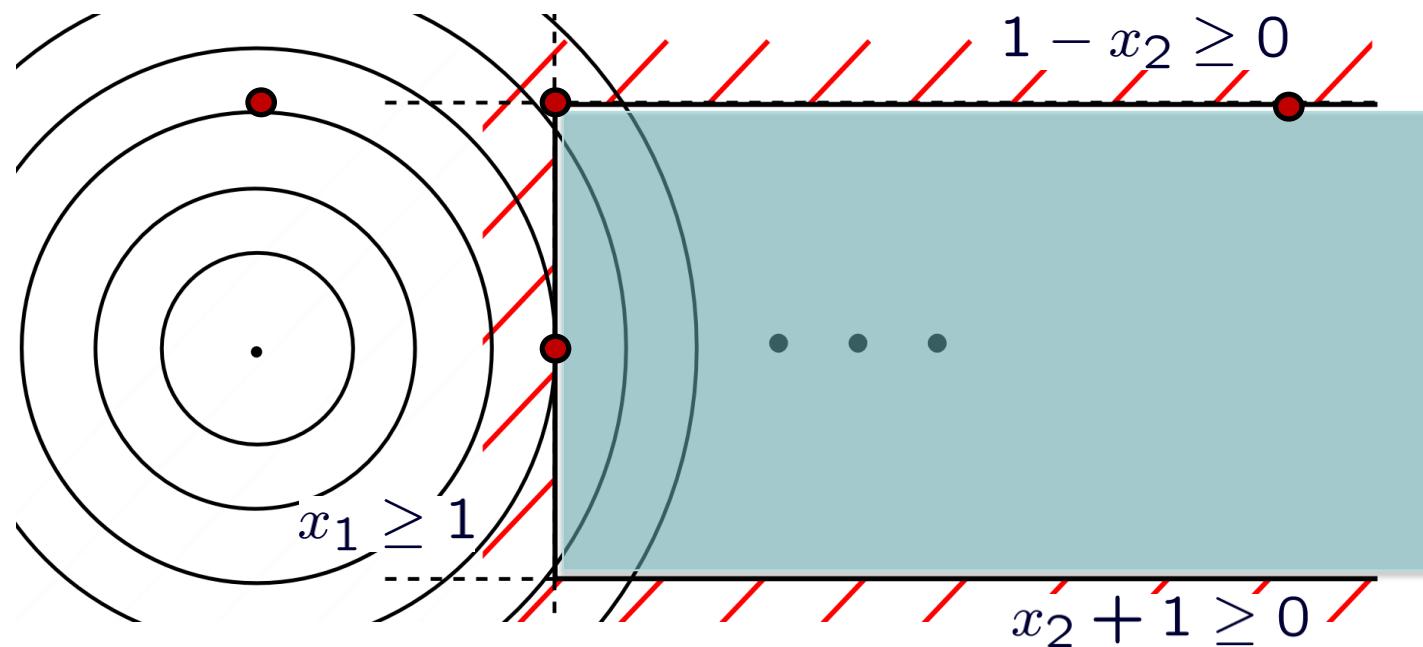
- Problem:

$$\min ||x||_2^2$$

subject to  $x_1 \geq 1$  (1)

$$x_2 + 1 \geq 0 \quad (2)$$

$$1 - x_2 \geq 0 \quad (3)$$



- Active Set:  $\{3\} \rightarrow \{3, 1\} \rightarrow \{1\}$

- Die Startlösung  $x^0$  muss in der gültigen Menge liegen. Wie beim Simplex-Verfahren kann eine solche Lösung mit einem linearen Programm bestimmt werden (siehe Vorlesung 5):

$$\min_{x,z} e^\top z, \quad a_i^\top x + z_i \geq b_i \quad z \geq 0$$

$$e := (1, \dots, 1)^\top$$

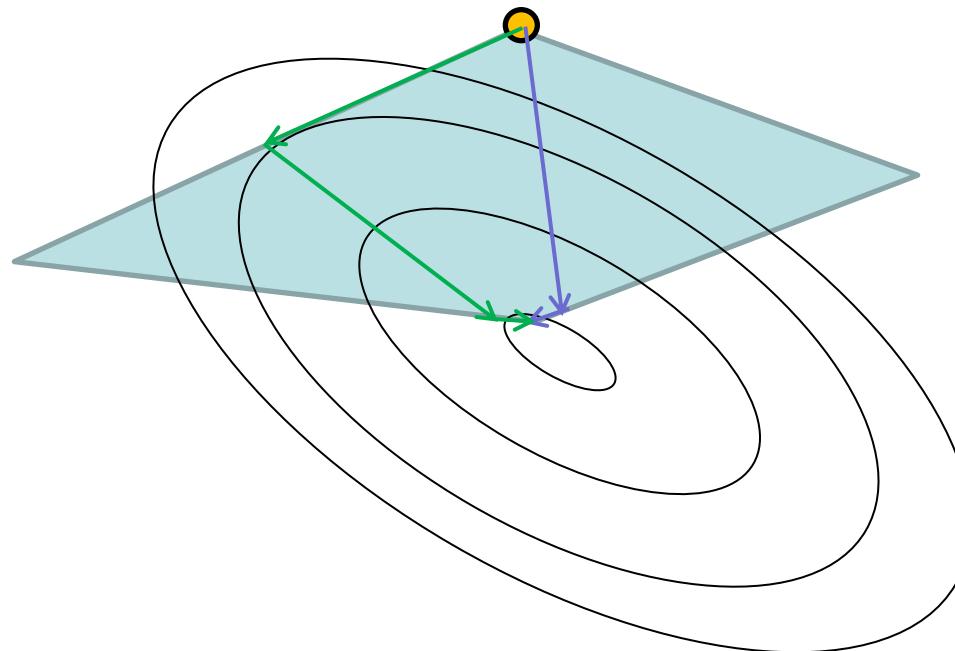
- Für beliebige Wahl von  $x = x_s$  ist

$$z_i = \max(b_i - a_i^\top x_s, 0)$$

eine gültige Lösung

- Durch Minimierung von  $e^\top z$  erhält man  $z = 0$  und ein  $x$ , das in der gültigen Menge des quadratischen Programms liegt.

- Als erste Arbeitsmenge kann eine linear unabhängige Untermenge der aktiven Bedingungen gewählt werden.
- Bei einer großen Anzahl von Bedingungen kann auch  $\mathcal{W} = \emptyset$  eine gute Wahl sein.
- Die Wahl der Arbeitsmenge beeinflusst den Weg zum Minimum.



- Dimension von  $x : n$   
Anzahl linear unabhängiger Nebenbedingungen:  $m \leq n$   
Anzahl Nebenbedingungen:  $M$
- In jedem Schritt müssen wir ein quadratisches Programm mit Gleichheitsbedingungen (Arbeitsmenge) lösen.
  - Invertierung von  $Q$ :  $\mathcal{O}(n^3)$   
(eventuell günstiger, falls  $Q$  z.B. eine Diagonalmatrix oder dünn besetzt ist)
  - Lösen des Gleichungssystems
$$AQ^{-1}A^\top \lambda = AQ^{-1}c + b$$
hat allgemein Komplexität:  $\mathcal{O}(mn^2)$
- Außerdem muss in jedem Schritt die Gültigkeit aller Nebenbedingungen überprüft werden:  $\mathcal{O}(Mn)$

- Da wir in jeder Iteration maximal eine Bedingung hinzufügen oder entfernen, brauchen wir mindestens so viele Iterationen wie es Unterschiede in der Größe der initialen Arbeitsmenge und der optimalen Arbeitsmenge gibt.
- Um keine Zyklen zu bekommen, muss wie beim Simplex-Verfahren sichergestellt sein, dass die gleiche Arbeitsmenge nicht mehrfach besucht wird.
- Bei nicht degenerierten Problemen ist dies der Fall, da dann einer von drei Fällen eintreten muss:
  1. Nicht-trivialer Schritt, d.h. die Funktion wird weiter minimiert und wir entfernen uns vom Rand ehemaliger Bedingungen der Arbeitsmenge. Diese werden daher nie wieder aufgenommen.
  2. Eine neue Bedingung ist verletzt. Nach spätestens  $n$  Iterationen kann keine Bedingung mehr verletzt sein.
  3. Eine Bedingung muss entfernt werden, um weiter zu minimieren. Danach machen wir wieder einen nicht-trivialen Schritt.

- Wie bei linearen Programmen gibt es die Möglichkeit, den Rand der gültigen Menge zu meiden statt ihn zu suchen.
- Diese **Innere-Punkt-Methoden** sind bei großen Problemen meist effizienter.
- Falls die Nebenbedingungen eine einfache Form haben, sind **Projektionsmethoden** oft eine effiziente Wahl.
- Beide Ansätze werden in der nächsten Vorlesung kurz vorgestellt.

- Quadratische Programme können konvex oder nicht konvex sein, abhängig von  $Q$
- Konvexe quadratische Programme können immer global optimiert werden.
- Es kann wie bei der linearen Programmierung ein Aktive-Set-Verfahren verwendet werden.
- Das Optimum liegt nicht zwangsläufig auf dem Rand der gültigen Menge.
- Das Verfahren verwendet eine dynamische Arbeitsmenge von Nebenbedingungen.

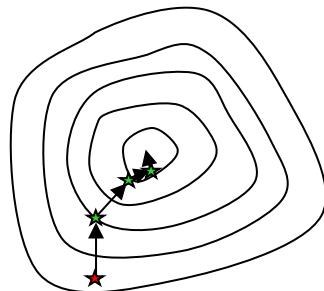




# Optimierung

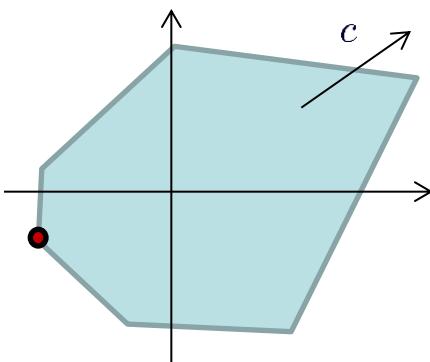
---

Vorlesung 7  
Nichtlineare Programmierung



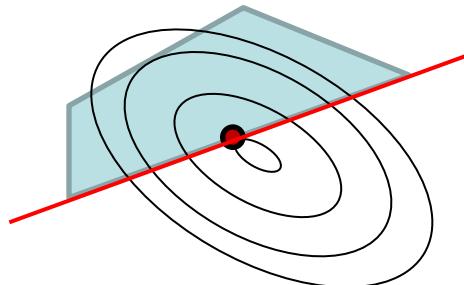
### Optimierung ohne Nebenbedingungen

- Gradientenabstieg, Quasi-Newton, Newton
- Beliebige glatte, nichtlineare Funktionen



### Lineare Programme (Simplex-Verfahren)

- Lineare Funktion, lineare Nebenbedingungen



### Quadratische Programme (Active-Set-Verfahren)

- Konvexe quadratische Funktion,  
lineare Nebenbedingungen

- Lineare und quadratische Programme sind sehr eingeschränkte Problemklassen
- Die Active-Set-Verfahren waren sehr speziell auf diese Problemklassen ausgelegt.
- Wie lösen wir Probleme mit allgemeineren Funktionen und allgemeineren Nebenbedingungen?
- Es gibt zwei Arten von Ansätzen:
  1. Sequentielle quadratische Programmierung (SQP)  
(Lösen einer Reihe quadratischer Programme)
  2. Sequentielle Approximation durch Formulierungen ohne Nebenbedingungen  
(Projektionsverfahren, Strafverfahren, Barriereverfahren)

- Hauptidee: lokale quadratische Approximation der Lagrange-Funktion und lokale lineare Approximation der aktiven Nebenbedingungen

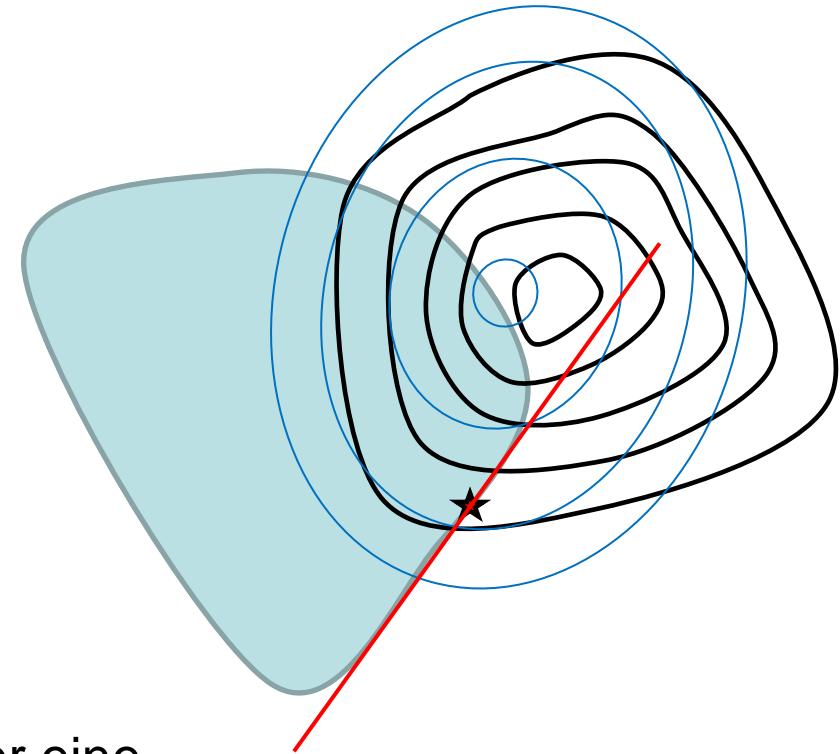
- Motivation ähnlich wie bei der Optimierung ohne Nebenbedingungen:

Quadratische Approximation entspricht Newton-Verfahren

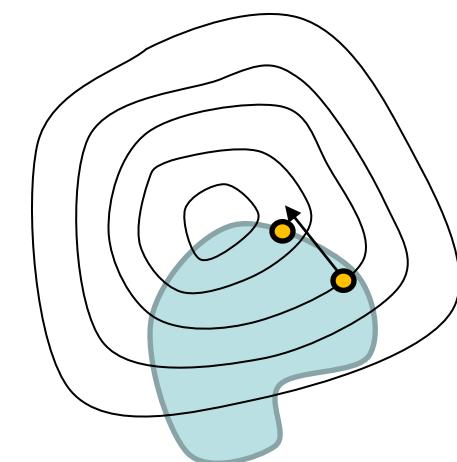
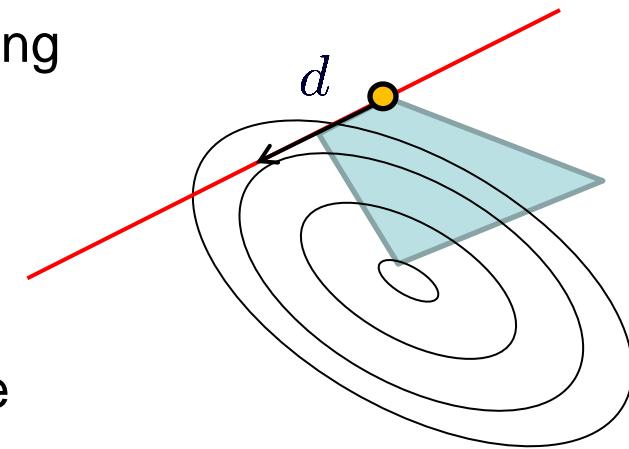
Ausführen eines Schritts, danach erneute Approximation

- Die Nebenbedingungen werden über eine Arbeitsmenge (wie bei der quadratischen Programmierung) berücksichtigt

Die Arbeitsmenge wird immer wieder aktualisiert.



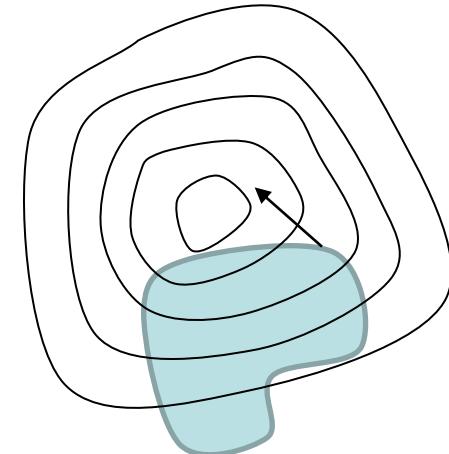
- Das Verfahren zur quadratischen Programmierung in der letzten Vorlesung verletzte zunächst Nebenbedingungen, die nicht in der Arbeitsmenge enthalten waren.
- Wir verhinderten dies, indem wir die Schrittänge passend verkürzten.
- Könnten wir die Nebenbedingungen zunächst nicht einfach ignorieren, einen Gradientenschritt berechnen und den Schritt dann so anpassen, dass alle Bedingungen eingehalten werden?
- Dies ist die allgemeine Idee von Projektionsmethoden mit verschiedenen Vor- und Nachteilen.



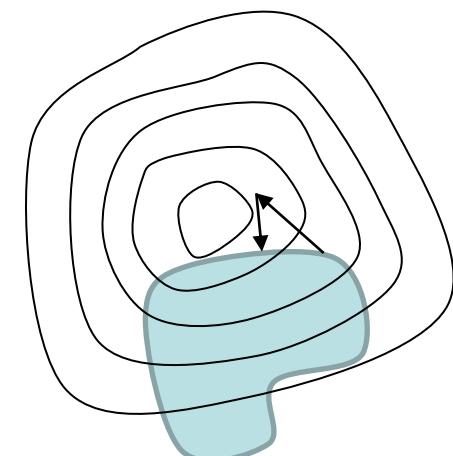
## Schrittverkürzung reicht nicht

- Die Idee, den Schritt einfach so zu verkürzen, dass alle Bedingungen eingehalten werden, führt irgendwann zu Schritten der Länge 0.

→ Das Verfahren stoppt in einem Punkt, der die KKT-Bedingungen normalerweise nicht erfüllt.



- Um dies zu verhindern, müssten wir den Gradienten bereits unter Berücksichtigung der aktiven Bedingungen berechnen (Active-Set-Methode).
- Wenn wir zunächst alle Nebenbedingungen ignorieren möchten, müssen wir stattdessen die neue Lösung auf die gültige Menge zurückprojizieren.
- Aber wie projizieren?



- Nicht jede Projektion ist geeignet.
- Beispiel: Funktion mit zwei Variablen  $x_1, x_2$  und den Bedingungen

$$x_1 \geq 0$$

$$x_2 \geq 0$$

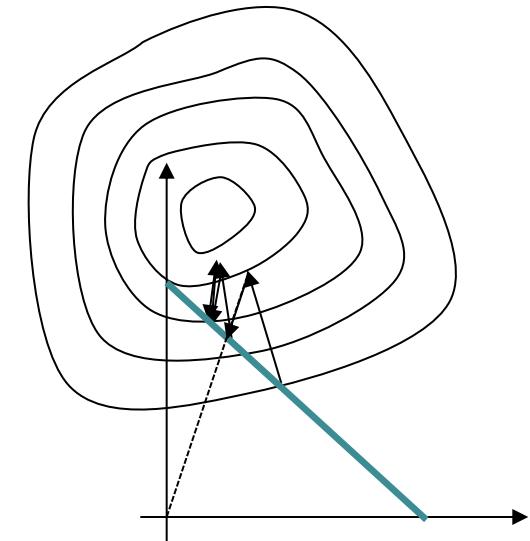
$$x_1 + x_2 = 1$$

- Projizieren wir nach jedem Schritt mittels

$$x_1 \leftarrow \frac{x_1}{x_1 + x_2} \quad x_2 \leftarrow \frac{x_2}{x_1 + x_2}$$

konvergiert das Verfahren nicht zum Optimum.

- Der Schritt entlang der gültigen Menge wird irgendwann komplett von der schlechten Projektion kompensiert.



# Beispiel kommt in einem Interpolationsproblem vor

- Gegeben: Einzelne Bildpunkte mit Labels  
Gesucht: Labels auf allen Bildpunkten
- Optimierungsproblem:

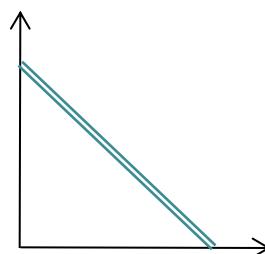
$$f(x) = \sum_{i=1}^n \left( c_i^\top x_i + \mathcal{R}(x_i) \right)$$

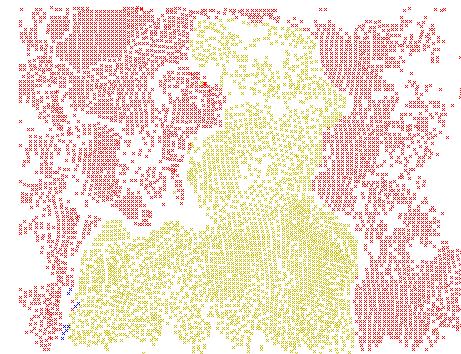
Kosten  $c_i > 0$  wenn ein Bildpunkt nicht das vorgegebene Label annimmt plus ein Strafterm  $\mathcal{R}$ , für benachbarte Punkte mit unterschiedlichen Labels

Nebenbedingungen:

$$\sum x_{il} = 1 \quad \forall l$$

$$x_{il} \in [0, 1] \quad \forall i, l$$





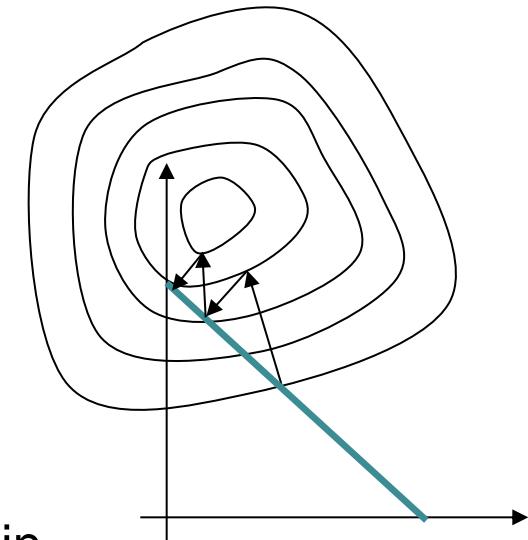
Eingabe

Iterationen



Indikatorvektor der roten Region (weiß=1, schwarz=0)

- Um Konvergenz zu einem Optimum sicherzustellen, müssen wir orthogonal zur Oberfläche der gültigen Menge projizieren.
- Der Anteil entlang der Oberfläche wird durch diese Projektion nicht mehr beeinflusst.
- Problem: Allgemein ist es sehr schwierig eine orthogonale Projektion auf die gültige Menge zu bestimmen.
- Dazu müssten die aktiven Bedingungen bekannt sein, was wieder zu einem Active-Set-Verfahren führt.
- Projektionsmethoden eignen sich daher nur für Probleme mit einfachen Nebenbedingungen, bei denen die orthogonale Projektion einfach ist.



- Bei einigen quadratischen Programmen kann eine einfachere Form der Nebenbedingungen durch Betrachtung des dualen Problems erreicht werden.

Quadratisches Programm:

$$\min_x \frac{1}{2} x^\top Q x + c^\top x, \quad Ax \geq b$$

Duales quadratisches Programm:

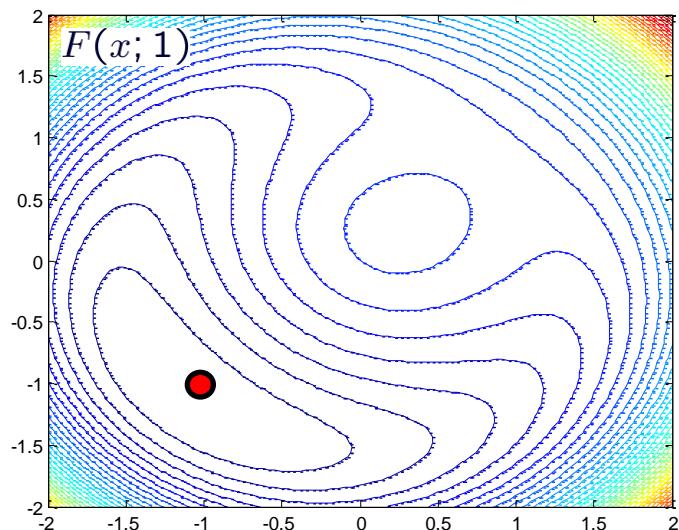
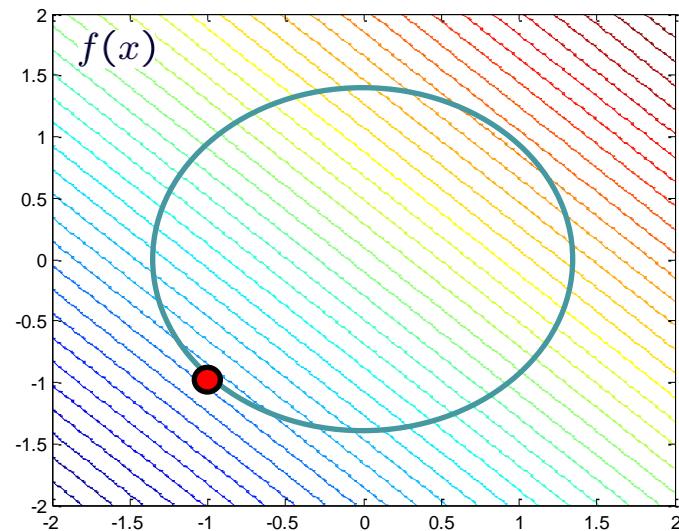
$$\max_\lambda -\frac{1}{2} (A^\top \lambda - c)^\top Q^{-1} (A^\top \lambda - c) + b^\top \lambda, \quad \lambda \geq 0$$

- Das duale Problem hat sehr einfache Nebenbedingungen, auf die man leicht projizieren kann.
- Sehr effiziente Lösungsstrategie, wenn  $Q$  einfach zu invertieren ist (siehe Beispiel der Support Vector Machine wo  $Q$  die Einheitsmatrix ist)

- Idee: Modifiziere die Zielfunktion  $f$  so, dass Abweichungen von den Nebenbedingungen bestraft werden:

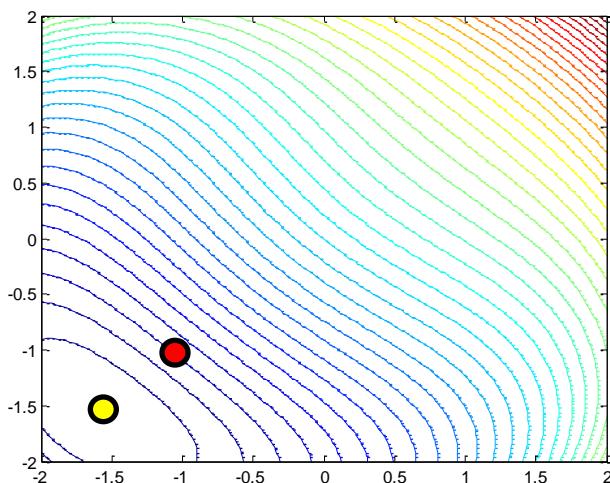
$$F(x; \mu) = f(x) + \frac{\mu}{2} \sum_i c_i^2(x)$$

- Dies ist nicht zu verwechseln mit der Lagrange-Funktion, die notwendige Bedingungen für ein Optimum definiert, aber selbst nicht minimiert wird.
- Im Optimum von  $F$  werden alle Nebenbedingungen eingehalten sobald  $\mu$  groß genug gewählt wird.

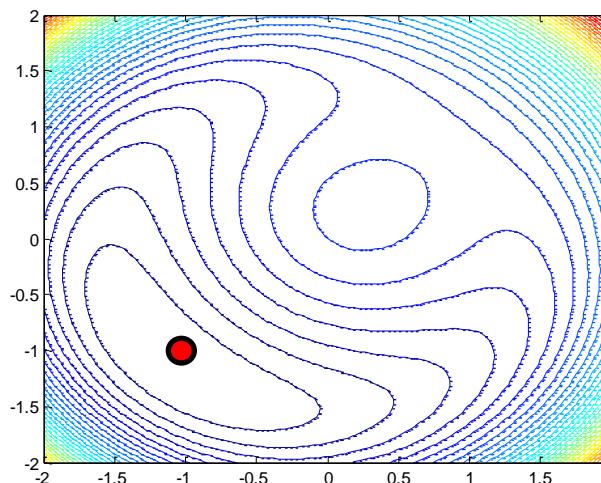


## Wahl des Strafparameters

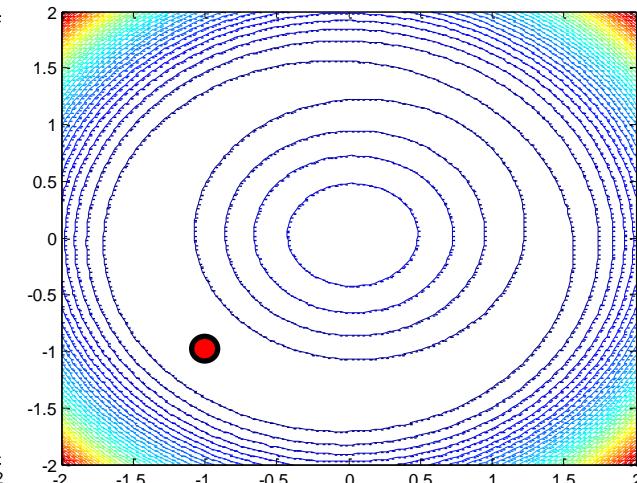
- Ist der Parameter  $\mu$  zu klein, erfüllt das Optimum von  $F$  nicht die Nebenbedingungen
- Mit größerem Einfluss des Strafterms steigt jedoch die Konditionszahl der Hesse-Matrix → numerische Probleme



$$\mu = 0.1$$



$$\mu = 1$$



$$\mu = 10$$

- Man löst daher eine Sequenz von Minimierungsproblemen mit größer werdenden  $\mu$

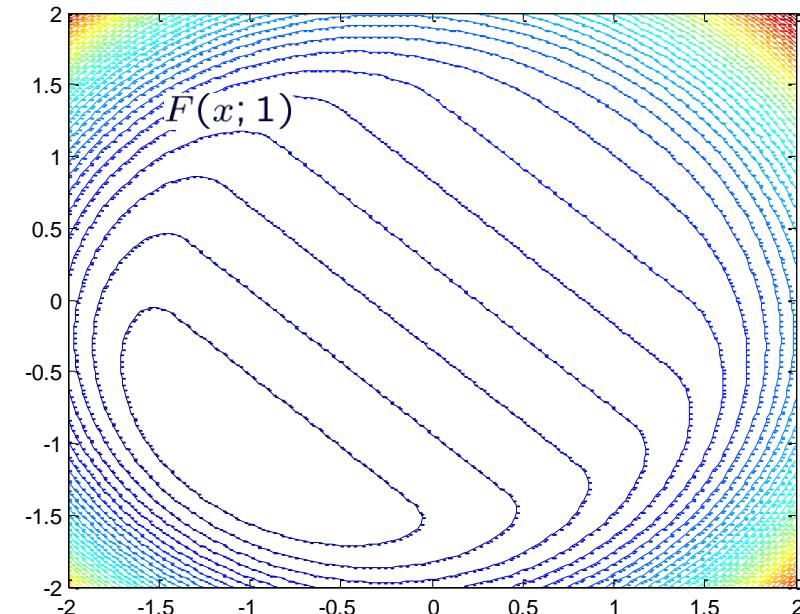
- Auch bei Ungleichheitsbedingungen

$$c_i \geq 0$$

können Strafterme eingesetzt werden:

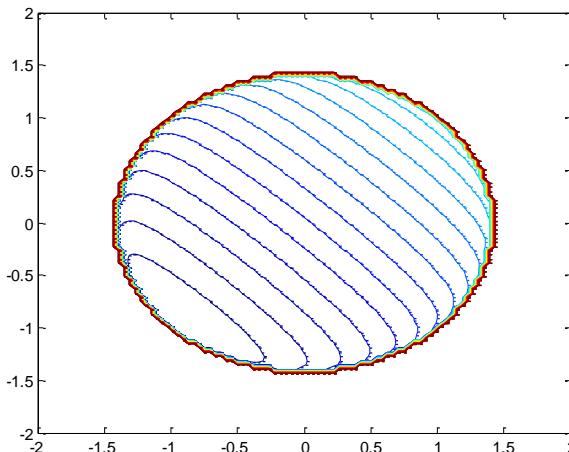
$$F(x; \mu) = f(x) + \frac{\mu}{2} \sum_i (\max(-c_i(x), 0))^2$$

- Hier tritt ein weiteres Problem auf:  
Die Funktion ist nur noch einmal differenzierbar.
- Problem für das Newton-Verfahren, das wiederum mit der ungleichmäßigen Skalierung am besten umgehen könnte

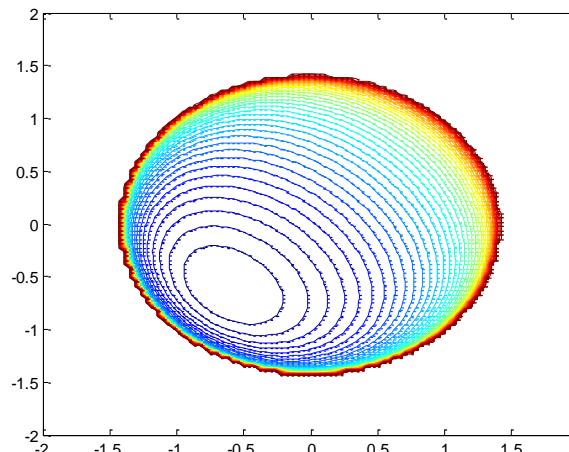


- Eine sehr ähnliche Motivation liegt den Log-Barrier-Verfahren zugrunde

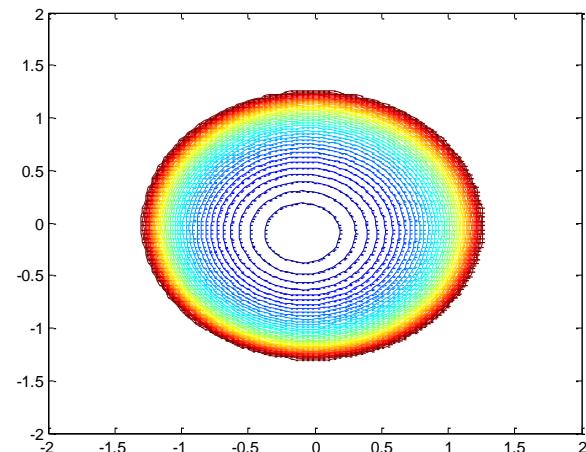
$$F(x; \mu) = f(x) - \mu \sum_i \log c_i(x)$$



$$\mu = 0.1$$



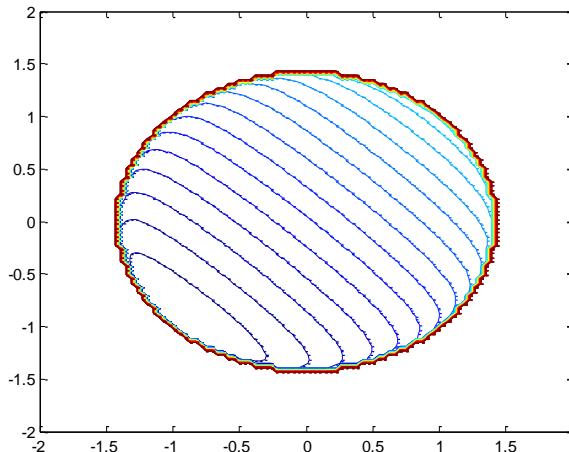
$$\mu = 1$$



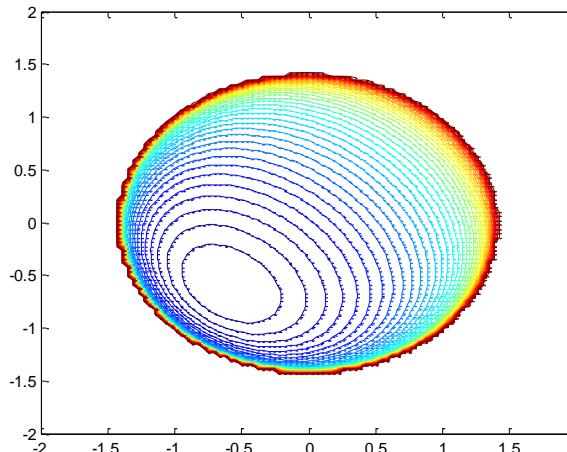
$$\mu = 10$$

- Da  $\log(c) \rightarrow -\infty$  für  $c \rightarrow 0$ , ist für alle Minima von  $F$  mit  $\mu > 0$  sichergestellt, dass die Nebenbedingungen eingehalten werden.
- Man verwendet diesmal eine Sequenz von kleiner werdenden  $\mu$

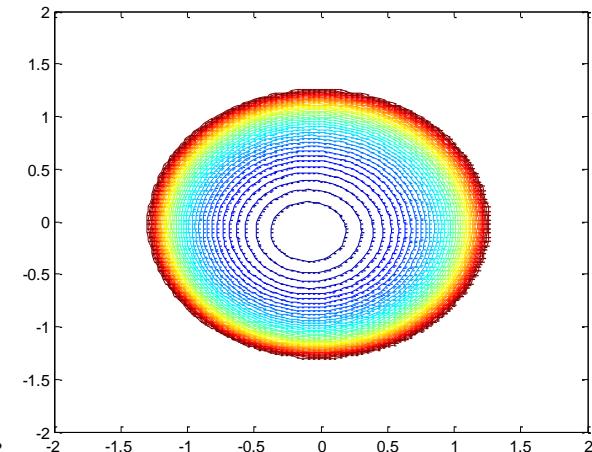
# Innere-Punkt-Methode



$$\mu = 0.1$$



$$\mu = 1$$



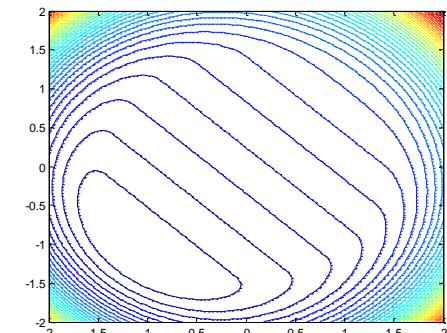
$$\mu = 10$$

- Die durch den Logarithmus aufgebaute Barriere verhindert, dass wir uns beim Optimieren dem Rand der gültigen Menge zu sehr nähern.

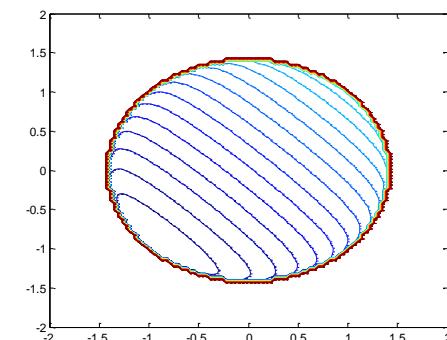
## → Innere-Punkt-Methode

- Dies steht im Gegensatz zur Strategie der Active-Set-Verfahren, die vermehrt am Rand der gültigen Menge entlanglaufen (z.B. Simplex-Verfahren, SQP)

- Wie der quadratische Strafterm, führt auch dieser Strafterm zu schlecht konditionierten Hesse-Matrizen  
→ Newton-Verfahren empfohlen
- Die Funktion  $F$  lässt sich schlecht quadratisch approximieren. Darauf basiert aber das Newton-Verfahren.  
→ Meist niedrige Konvergenzraten
- Gleichheitsbedingungen bedürfen eines quadratischen Strafterms, mit den entsprechenden Nachteilen.
- Innere-Punkt-Methoden waren daher lange Zeit unbeliebt bis in den 80/90er Jahren Primal-Dual-Verfahren (zunächst für lineare Programme) entwickelt wurden.



Quadratische Strafe



Log-Barrier

- Primal-Dual-Verfahren lösen das (leicht modifizierte) KKT-System, welches die primalen und dualen Variablen enthält.

- Betrachten wir zunächst lineare Programme

$$\min_x c^\top x, \quad Ax = b, x \geq 0$$

- Die KKT-Bedingungen:

$$A^\top \lambda + s = c$$

$$Ax = b$$

$$x_i s_i = 0, \quad \forall i$$

$$x, s \geq 0$$

- Wir möchten das Gleichungssystem unter Einhaltung von  $x, s \geq 0$  lösen.
- Hierfür können wir die Gleichungen auch als Residuum eines Least-Squares-Problems formulieren (siehe Vorlesung 3)

- Wir haben das Residuum

$$r(x, \lambda, s) = \begin{pmatrix} A^\top \lambda + s - c \\ Ax - b \\ XSe \end{pmatrix}$$

mit Diagonalmatrizen  $X$  und  $S$  mit Einträgen  $x_i$  bzw.  $s_i$  Und  $e = (1, \dots, 1)^\top$

- Linearisierung für den Gauss-Newton-Schritt

$$\underbrace{r(x^{k+1}, \lambda^{k+1}, s^{k+1})}_{=0} = r(x^k, \lambda^k, s^k) + J(x^k, \lambda^k, s^k)(\Delta x, \Delta \lambda, \Delta s)^\top$$

- Wir erhalten den Schritt durch Lösen von

$$\begin{matrix} \nabla x & \nabla \lambda & \nabla s \\ \nabla(A^\top \lambda + s + c) & \left( \begin{array}{ccc} 0 & A^\top & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{array} \right) & \begin{pmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{pmatrix} \end{matrix} = \begin{pmatrix} c - A^\top \lambda^k - s^k \\ b - Ax^k \\ -X^k S^k e \end{pmatrix}$$

Jacobi-Matrix

Schritt

Jacobi-Matrix

Schritt

- Der Newton-Schritt kann die Bedingungen  $x, s \geq 0$  verletzen. Um diese einzuhalten, müssten wir den Schritt entsprechend verkürzen.

Führt meist zu sehr kleinen Schritten entlang des Rands.

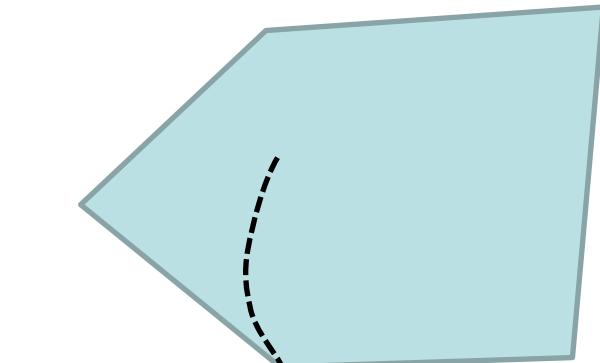
- Idee der Inneren-Punkt-Methode: der Rand der gültigen Menge ist tabu!
- Statt  $XSe = 0$  zu fordern (damit landen wir direkt auf dem Rand), fordern wir  $XSe = \mu e$

Dabei entspricht  $\mu$  dem **Dualitätsmaß**

$$\mu = \frac{1}{n} x^\top s$$

also der aktuellen durchschnittlichen Abweichung von der Komplementarität.

Verfahren erreicht den Rand erst im Optimum



Primale Darstellung des Pfads  
(ohne duale Variablen  $\lambda, s$ )

- Dieses Verfahren lässt sich für nichtlineare Programme verallgemeinern

$$\min_{x,z} f(x) \quad g_E(x) = 0 \quad g_I(x) - z = 0, z \geq 0$$

(Schlupfvariablen  $z$  zur Darstellung der Ungleichheitsbedingungen)

- KKT-Bedingungen

$$\nabla f(x) - A_E^\top(x)\lambda - A_I^\top(x)s = 0 \quad \text{Gradient der Lagrange-Funktion}$$

$$ZSe = 0 \quad \text{Komplementarität}$$

$$g_E(x) = 0$$

$$g_I(x) - z = 0$$

$$z, s \geq 0$$

- Auch hier können wir  $ZSe = 0$  zu  $ZSe = \mu e$  ändern und die entsprechenden Newton- oder Gauss-Newton-Schritte berechnen.

- Numerische Instabilität  
hohe Konditionszahlen, (fast) singuläre Matrizen
- Größe der Probleme  
viele Variablen, viele Nebenbedingungen  
→ Speicher- und Rechenzeitbeschränkungen
- Lineare/quadratische Approximation ist unpassend  
→ ineffiziente Schritte
- Fehlende Glattheit  
Gradient/Hesse-Matrix kann nicht berechnet werden  
→ Subgradienten-Verfahren
- Nicht-konvexe Funktionen  
→ lokale Minima

- Die Active-Set-Verfahren können auf allgemeine nichtlineare Programme erweitert werden, indem eine Sequenz quadratischer Programme gelöst wird.
- Projektionsmethoden eignen sich, wenn die orthogonale Projektion einfach vorzunehmen ist.
- Strafmethoden sind intuitiv einleuchtend aber schwer zu kontrollieren und numerisch instabil.
- Primal-Dual Innere-Punkt-Methoden lösen sequentiell das KKT-System, das in jedem Schritt so modifiziert wird, dass der Rand der gültigen Menge erst im Optimum erreicht wird.

1. Diesmal möchten wir den kleinen Hund in `puppy.png` vom Hintergrund segmentieren. Das Optimierungsproblem ist sehr ähnlich zu dem, das wir bei der Bildverbesserung bekommen haben:

$$f(x) = \sum_{i,j} \left( ((y_{ij} - 128)^2 - (y_{ij} - 255)^2) x_{ij} + \frac{1}{2} \sqrt{(x_{ij} - x_{i+1j})^2 + (x_{ij} - x_{ij+1})^2 + 1} \right)$$

allerdings haben wir noch Nebenbedingungen

$$x_{i,j} \in [0, 1], \forall i, j$$

Berechnen Sie den Gradienten.

2. Implementieren Sie wieder einen Gradientenabstieg. Dazu müssen Sie Ihre Implementierung aus der zweiten Übung nur geringfügig anpassen. Nach jedem Iterationsschritt müssen Sie diesmal jedoch noch auf die gültige Menge zurückprojizieren.
3. Wenden Sie das Verfahren auf `puppy.png` an. Verwenden Sie 0,5 als Startwerte. Sie sollten am Ende für jeden Bildpunkte Werte nahe 0 oder 1 bekommen. Diese stehen für Hintergrund bzw. Vordergrund. Genaugenommen haben Sie damit ein kombinatorisches Problem gelöst, das wir uns in der nächsten Vorlesung noch einmal genauer ansehen werden.