Optimierung

Vorlesung 1

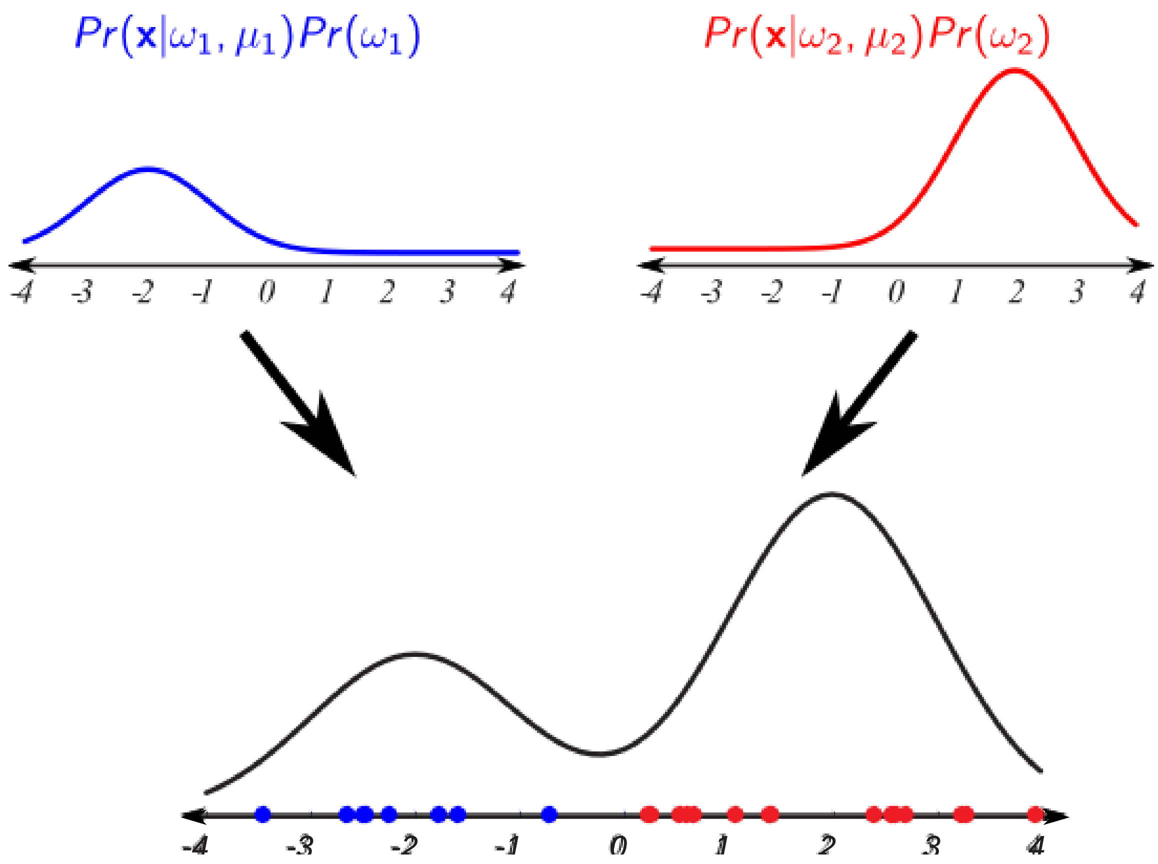
Einführung und Überblick

Wofür ist Optimierung gut?

- 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**)

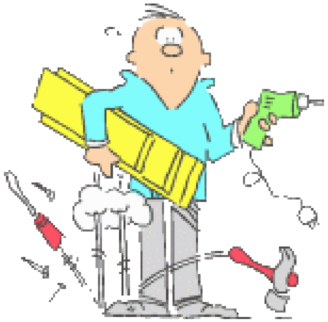
Beispiel Hauspreise

- 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

Vom Bastler zum Akademiker

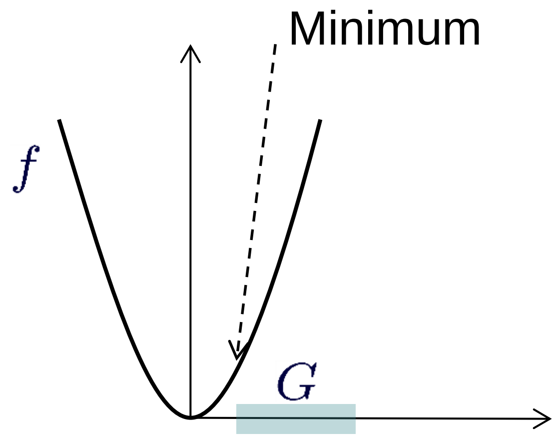


- Basteln und Ausprobieren ist die kreative Seite der Wissenschaft
- Für komplexe Aufgaben reicht Kreativität alleine nicht aus

- 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** $f : G \to \mathbb{R}$



- Beispiel:

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

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

- **Minimum**: $\min_x f(x) = 4$

- **Minimierer**: $\text{argmin}_x f(x) = 2$

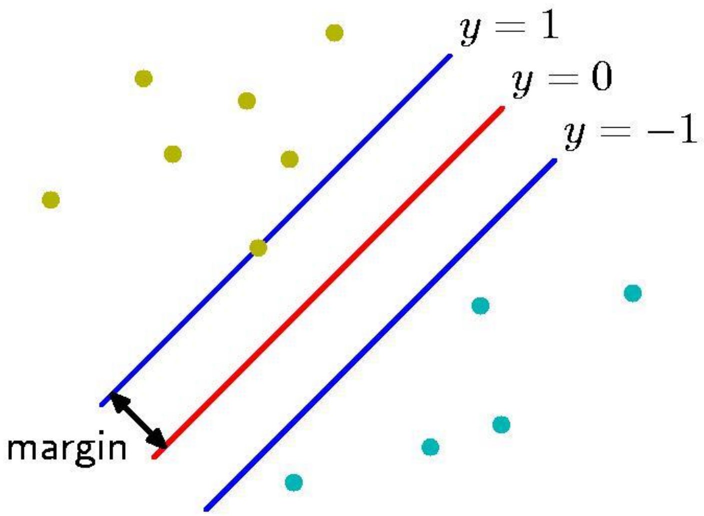
- Wir haben also ein Modell in Form einer Zielfunktion und möchten aus einer großen Anzahl möglicher Lösungen (zulässige Menge) die beste finden (Minimierer).

Anwendungen für Optimierung

- 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

Beispiel aus dem maschinellen Lernen: Support Vector Machine

- Gegeben: Samples $\mathbf{x}_n$ 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

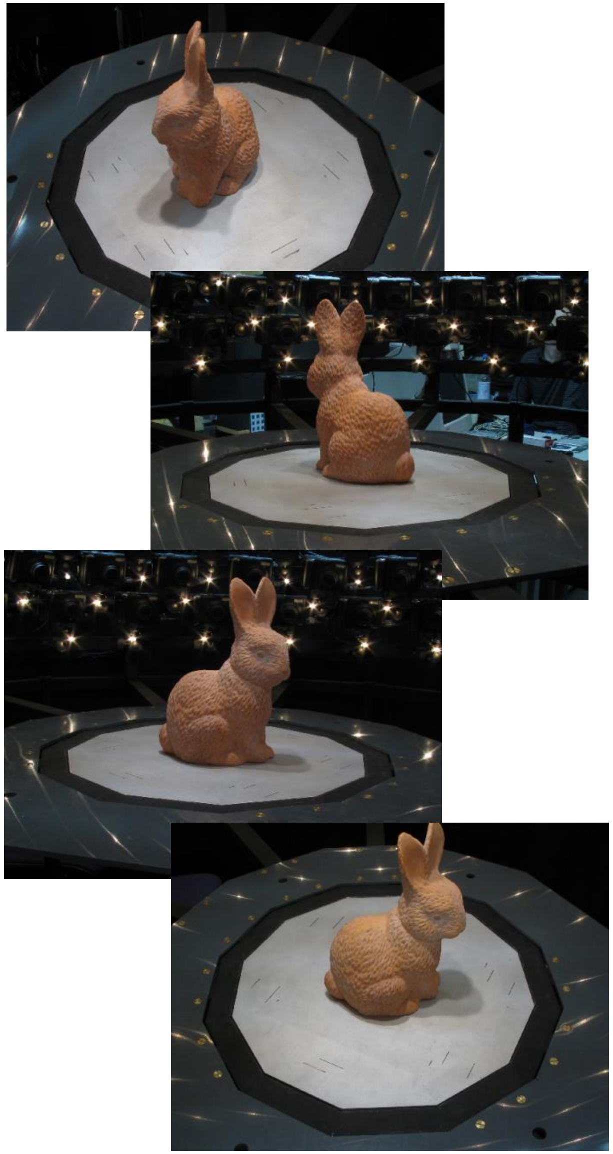
   $\mathrm{argmin}_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$ (kleine Gewichte sind günstiger)

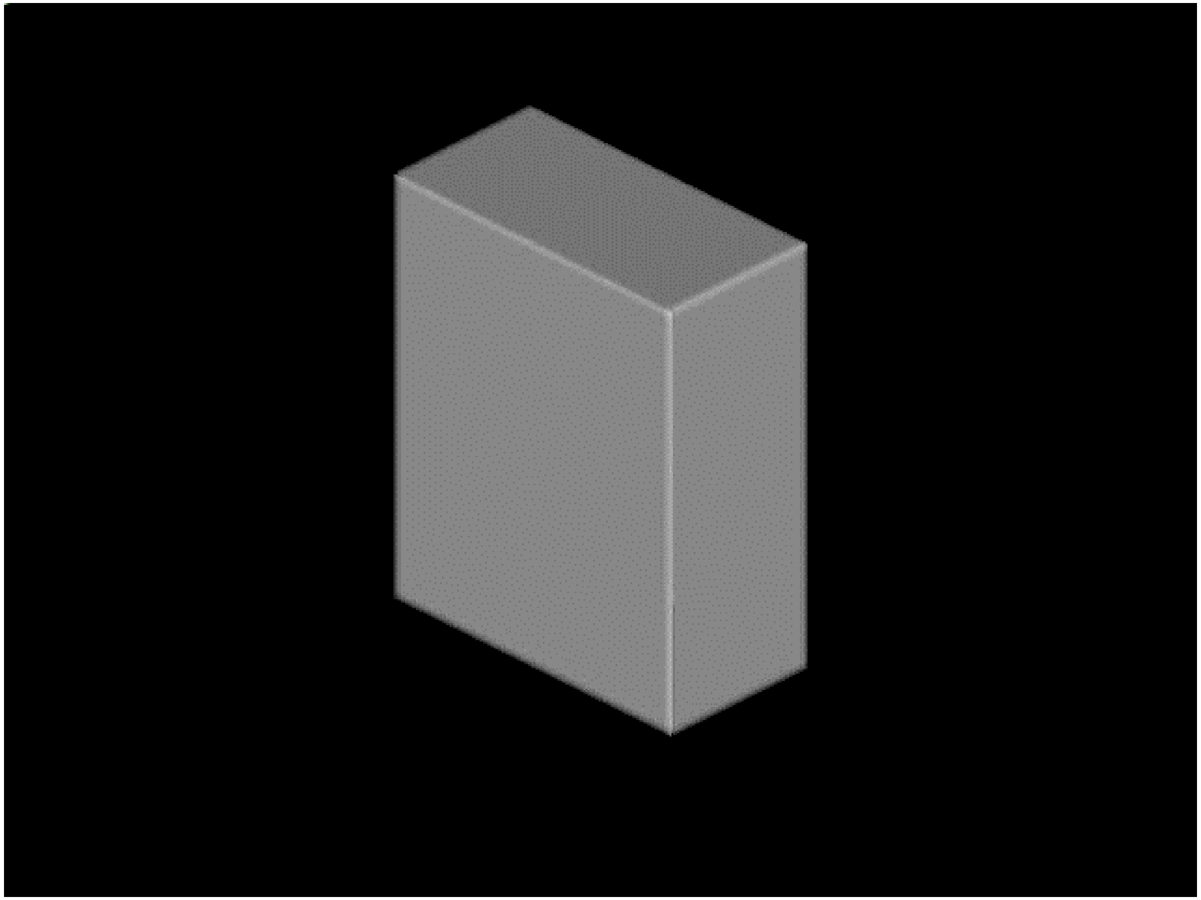
   mit Nebenbedingungen

   $t_n(\mathbf{w}^\top \mathbf{x}_n + b) \geq 1$ (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

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

Beispiel aus Computer Vision: Bildsegmentierung

- Ziel: Finde eine Kontur C, die das Bild In Vordergrund Ω_1 und Hintergrund Ω_2 aufteilt

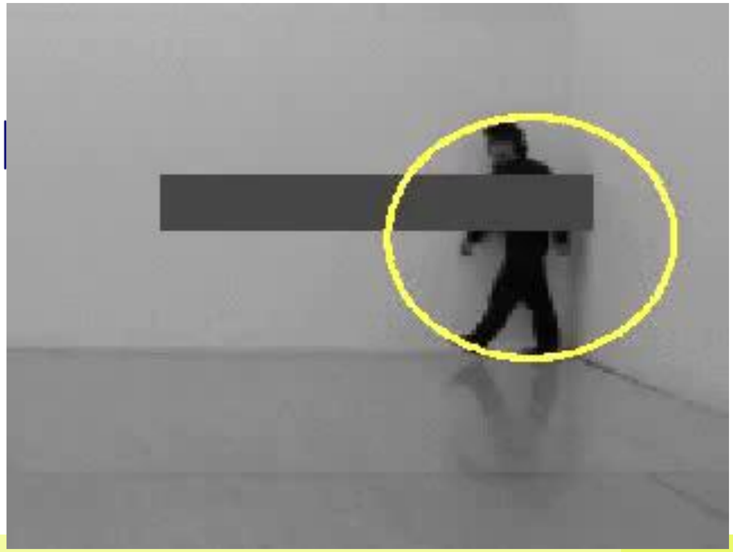
- Beide Regionen werden durch die mittlere Helligkeit μ_1, μ_2 modelliert

- Kürzere Konturen sind besser

- Kostenfunktion:

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

Kann mittels **Gradientenabstieg** minimiert werden



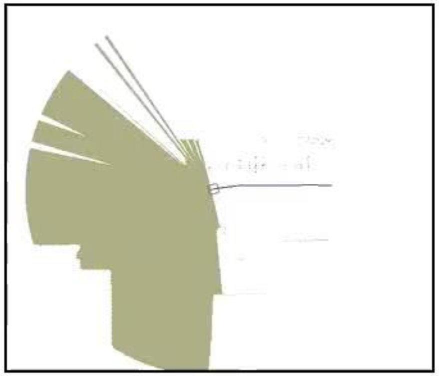
Autor: Daniel Cremers

Beispiel aus der Robotik: SLAM

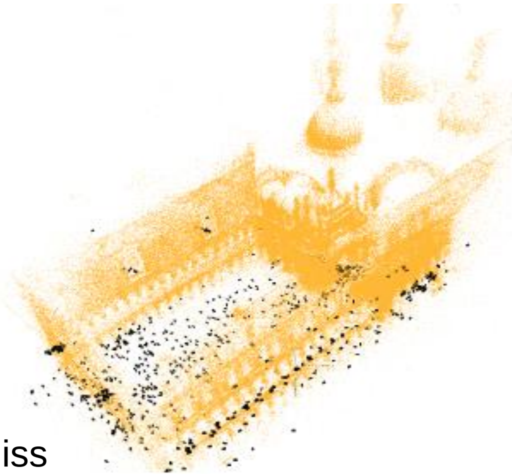
- Ziel: Erstellen einer Umgebungskarte und gleichzeitige Lokalisierung in der Karte (SLAM = simultaneous localization and mapping)

- Sensormessungen sind jeweils relativ zur Roboterposition
  → Berücksichtigung aller Messungen in einem Optimierungsproblem

- Lösung mit
  Gauss-Newton-Verfahren



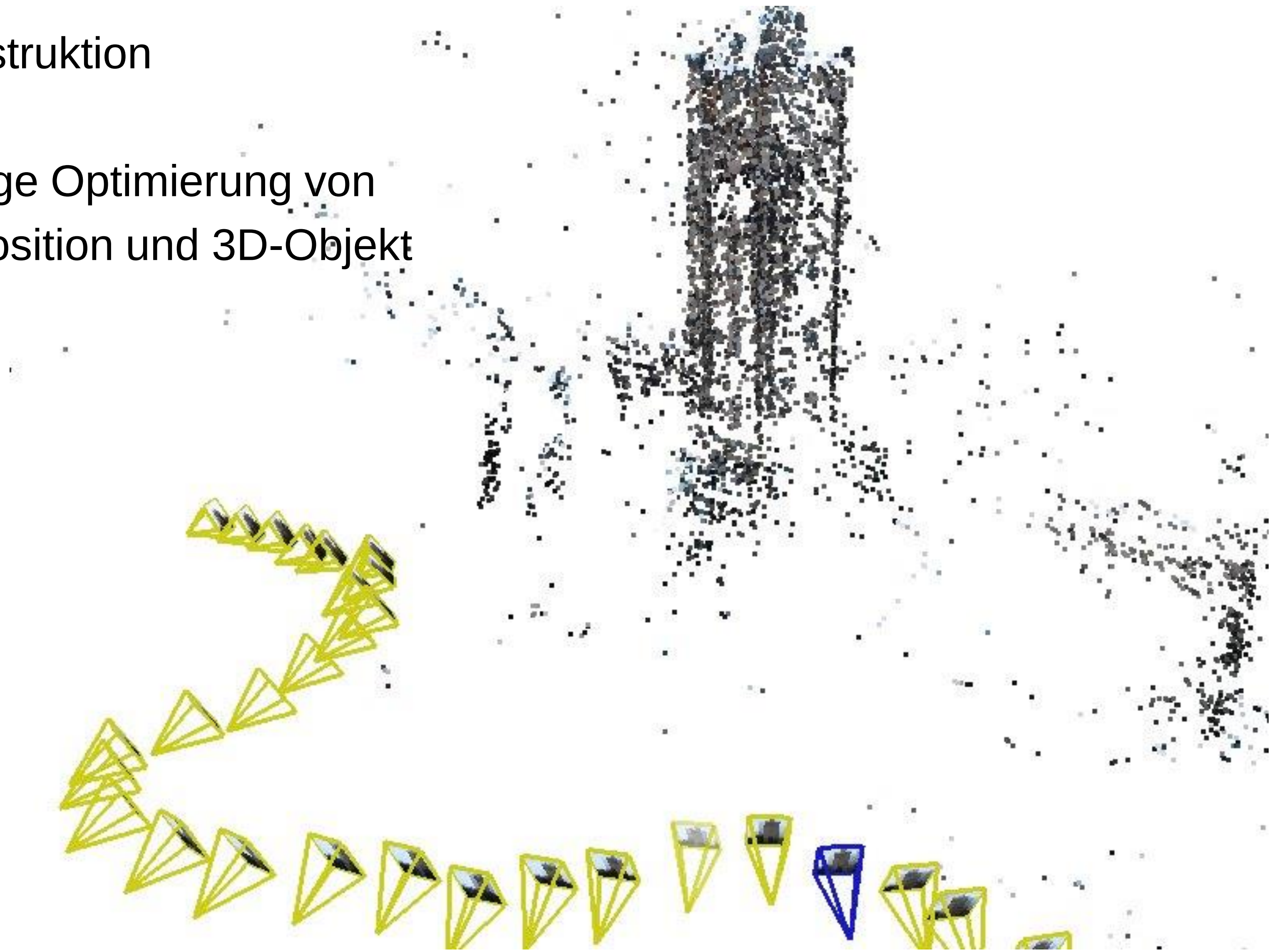
- Äquivalentes Problem (im Sinne der Optimierung) in Computer Vision: Bundle Adjustment



Autor: Cyrill Stachniss

Bundle Adjustment

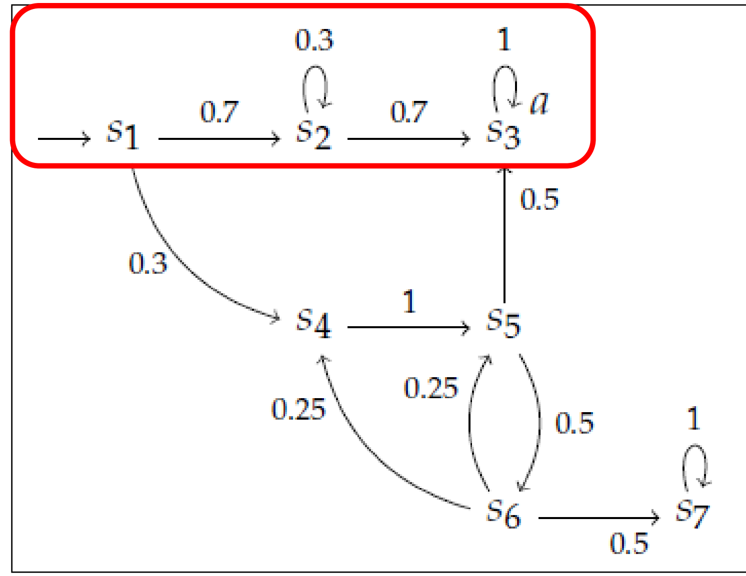
- 3D-Rekonstruktion
- Gleichzeitige Optimierung von Kamera-Position und 3D-Objekt



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

Beispiel aus dem Hard-/Softwaredesign: Sicherheitsgarantien

- 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

$$
\begin{aligned}
\text{minimize} \quad & -\frac{1}{2} p_{s_{\text{init}}} + \sum_{s \in S} x_s \\
\text{such that} \quad & \\
\forall s \in T_a : \quad & p_s = x_s \\
\forall s \in S \setminus T_a : \quad & p_s \leq x_s \\
\forall s \in S \setminus T_a : \quad & p_s \leq \sum_{s' \in \text{succ}(s)} P(s, s') \cdot p_{s'} \\
& p_{s_{\text{init}}} > \lambda \, .
\end{aligned}
$$

Mixed integer linear program

Quelle: Wimmer et al. 2012

Beispiele aus der Bioinformatik

- 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

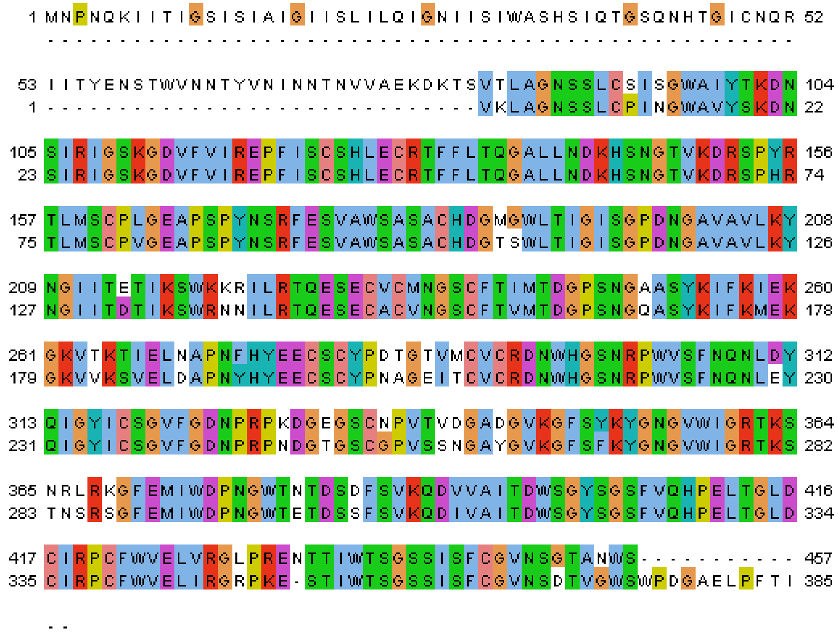
  Linear-Space mit Divide-and-Conquer

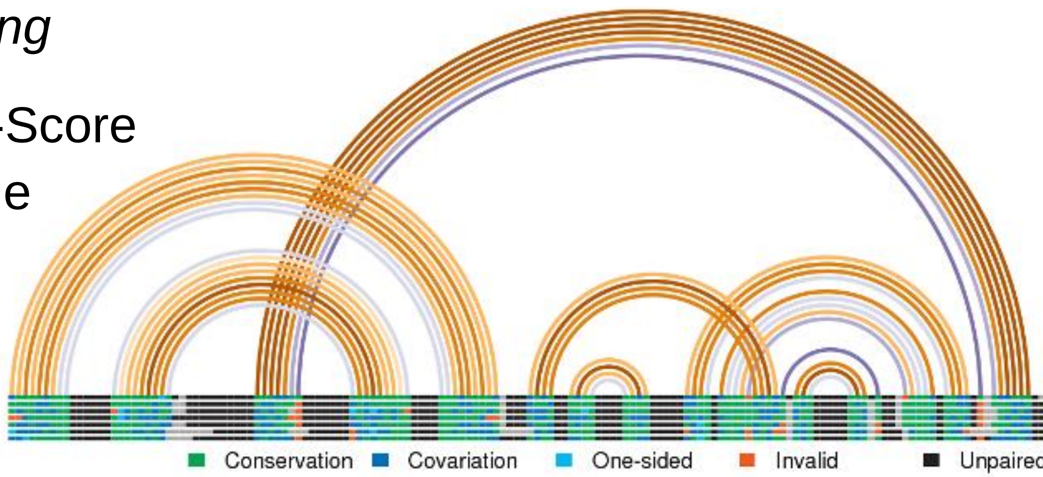
- Erweiterung: Strukturelles Alignment ist NP-schwer

  Integer Linear Programming

- Lokales Alignment: Alignment-Score wird durch die Alignment-Länge geteilt

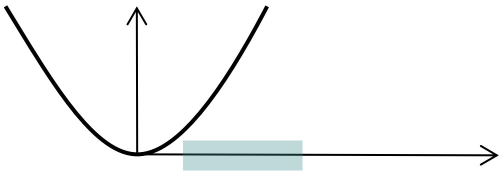
  Fractional Programming



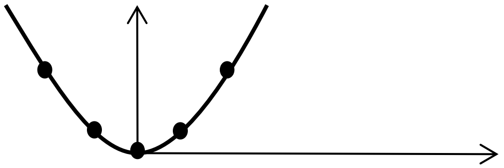


Typische Problemklassen in der Optimierung

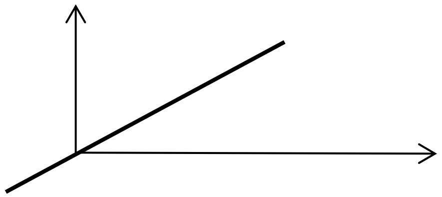
- Probleme mit Nebenbedingungen vs.
  Probleme ohne Nebenbedingungen



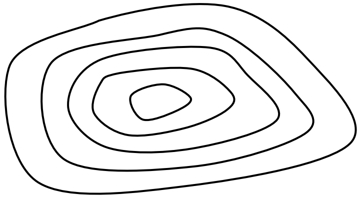
- Optimierung mit kontinuierlichen Variablen vs.
  diskreten Variablen



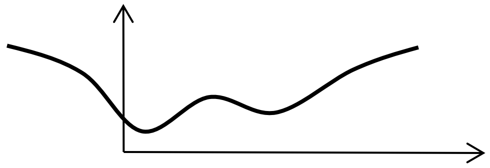
- Lineare vs. nichtlineare Funktionen



- Eindimensionale vs. mehrdimensionale
  Funktionen

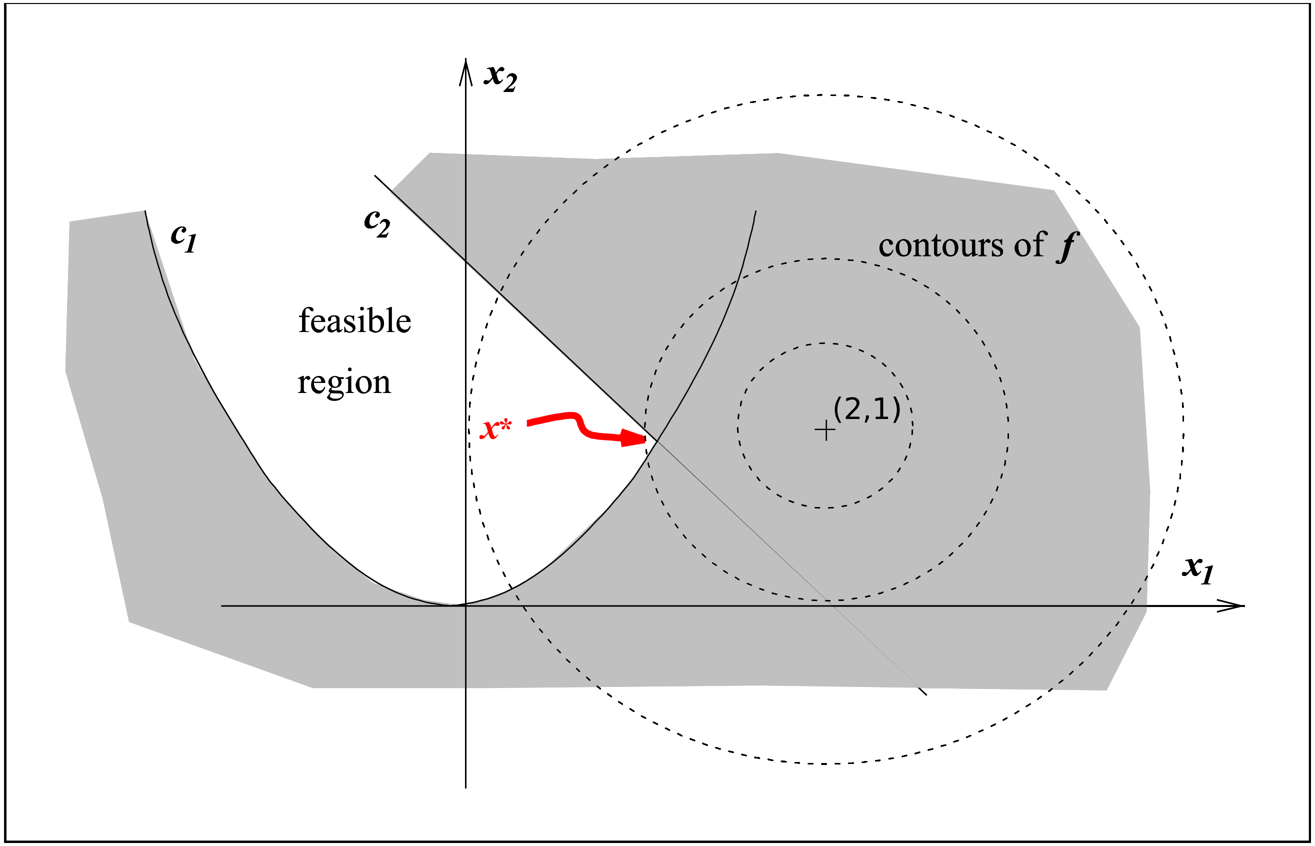


- Konvexe vs. nicht-konvexe Funktionen und
  Mengen (mehr dazu gleich)



Beispiel Kreisfunktion mit NB

- Problem: $\min(x_1-2)^2+(x_2-1)^2$ subject to $\begin{array}{l} x_1+x_2 \le 2 \\ x_1^2-x_2 \le 0 \end{array}$
- Lösungsbereich:



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

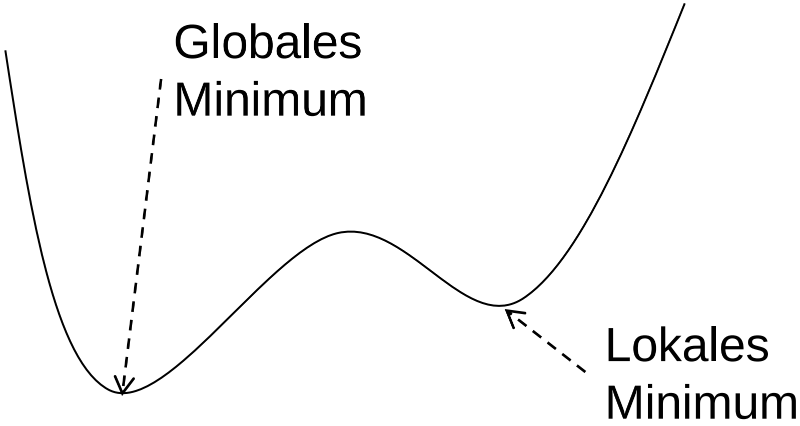
Optimalitätsgarantien

- 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**)?

Lokale und globale Minima



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

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

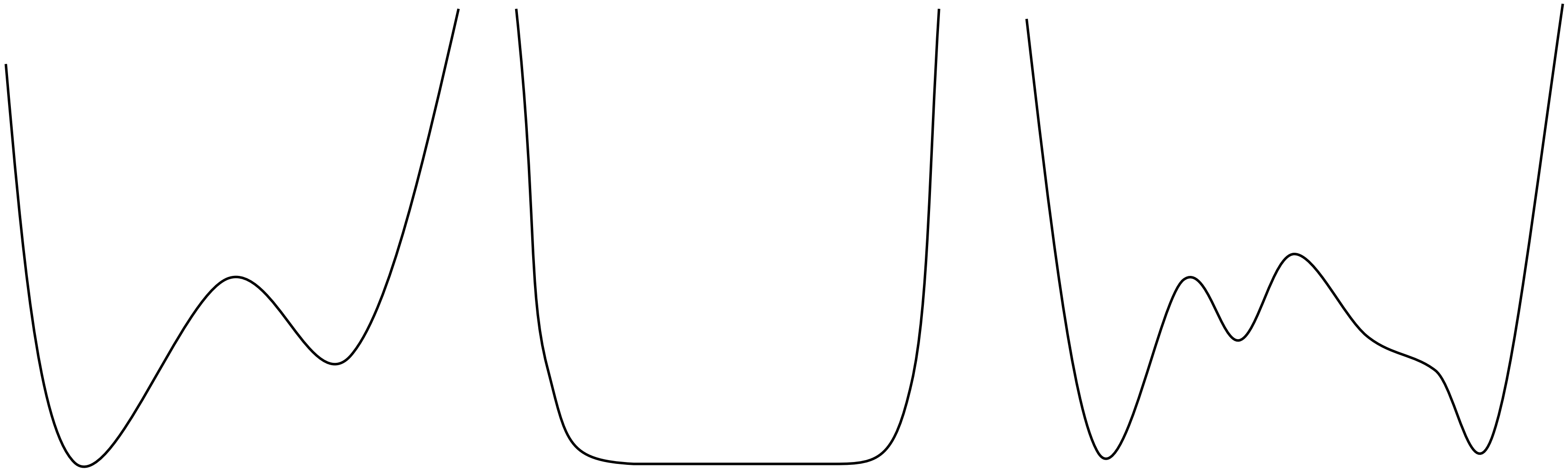
  Dabei ist $U(x^L)$ eine Normkugel mit einem hinreichend kleinen Radius $\epsilon > 0$:

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

- 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



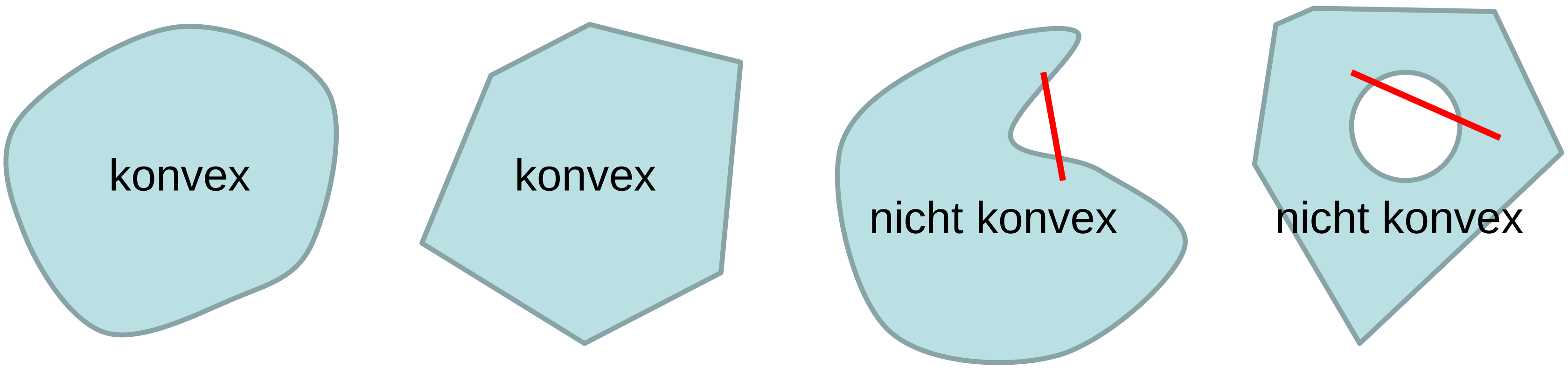
Eindeutiges globales Optimum

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?

Konvexität, konvexe Mengen



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

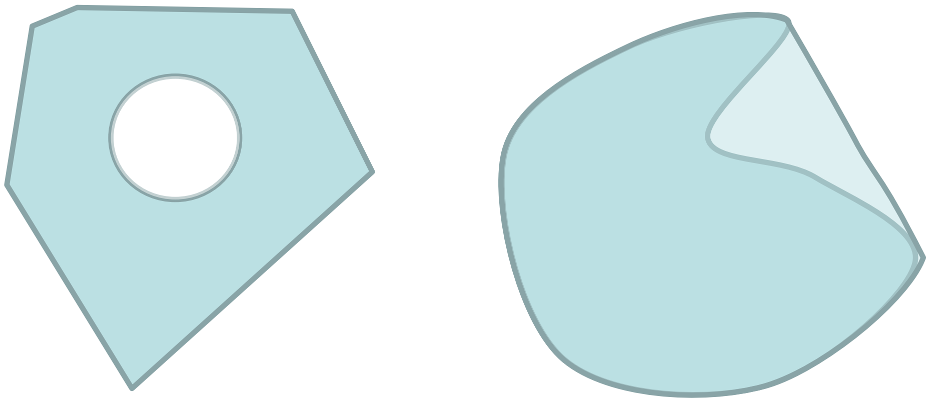
  $$[x, y] := \{z := (1-\lambda)x + \lambda y | \lambda \in [0, 1]\}$$

  in G enthalten ist:

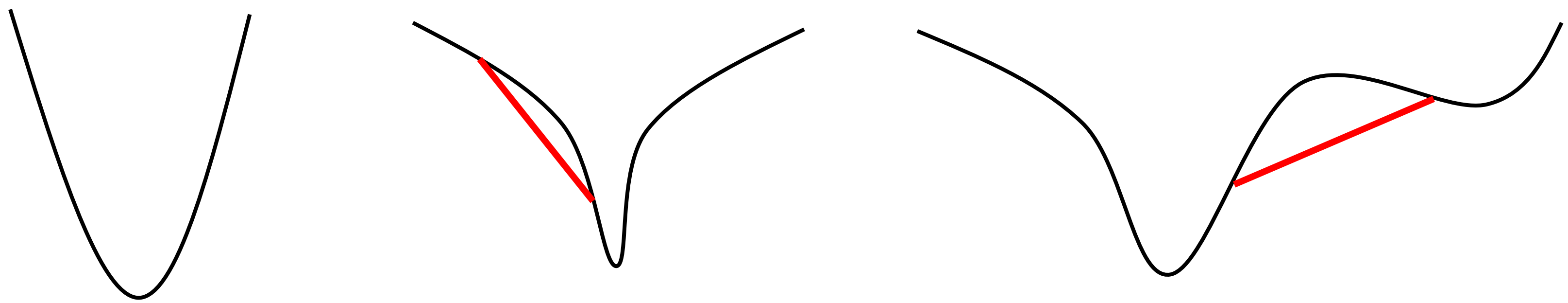
  $$x, y \in G \;\Rightarrow\; [x, y] \subset G$$

Konvexe Hülle

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



Konvexe Funktionen



konvex nicht konvex nicht konvex

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

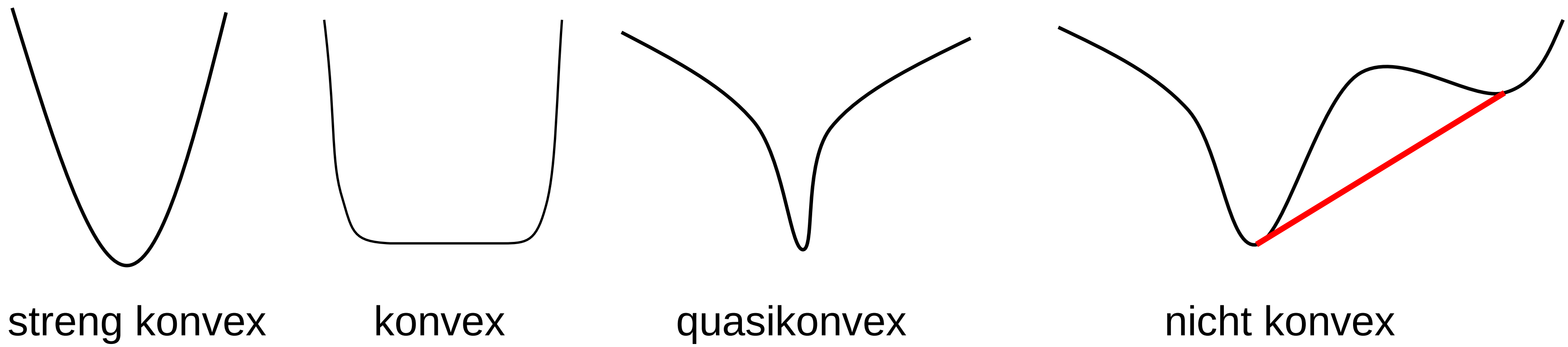
$$x, y \in G; x \neq y \;\Rightarrow\; f((1-\lambda)x+\lambda y) \leq (1-\lambda)f(x)+\lambda f(y) \quad \forall \lambda \in [0,1]$$

- Sie heißt **streng konvex**, falls

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

Konvexität und globale Minima

- 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 unter-halb des Graphen liegen!

Kursüberblick

- 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

Organisatorisches

- 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

Eigenverantwortung

- Sie sind erwachsene Menschen und Studenten einer **Universität**

- Fachwissen ist <u>nicht</u> 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)

Literatur

- 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.