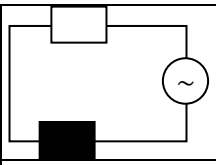
KÜNSTLICHE INTELLIGENZ IN DER PRODUKTEENTWICKLUNG

Einführung

Qualität	 Q* ist meist unbekannt Q* bezeichnet meist die Kosten
Welche Aufgaben sind für Computer geeignet?	**Ja** •repetitiv •formale Gesetze •gut-sehr gut messbar **Nein** •nicht formale Gesetze •nicht/schlecht messbar (nicht vollständig)
Gesucht ist ein	„Optimales-Design“ oder ein „Optimaler-Entwurf“
Nebenbedingungen =Restriktion = Constraints	sind Flächen, die den Hochdimensionalen Raum zerschneiden Die Bedingungen können sich wiedersprechen -> kein Punkt mit Optimum (irregulär)
MATLAB	$fmincon(\ \ )$
Maximumfunktion	maximal-Funktion umwandeln in minimal-Funktion durch Multiplizieren mit -1
Ungleichungen	bei Ungleichung, wenn Multiplikator negativ: Zeichen umkehren

Klassifikation & „Lösung“ von mathematischen Optimierungs-Aufgabe

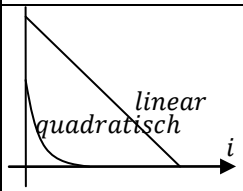
Bisher:

1. Aufgabe	
2. Modell	DGL (Mathematik)
3. Lösung	$x(t) = \cdots$
4. Interpretation	$u_R(t) = \cdots$

Neu:

1. Aufgabe	
2. Optimierungsaufgabe	$Min\ f(L_i, C_i)$ $g(L_i, C_i) \geq 0$ ab hier Mathematik
3. Lösung	$L_i' = \cdots$ $L_2' = \cdots$ $C_i = \cdots$
4. Interpretation	

Masse um Effizienz zu messen (anhand Dichotome Suche)

Anzahl Auswertungen	Anzahl Funktionsaufrufe	$N_f(i)$
Länge anhand Iteration	$L_1 = \frac{L_0}{2} + \varepsilon$ $L_2 = \frac{\frac{L_0}{2} + \varepsilon}{2} + \varepsilon$	$L_i = \left(\frac{1}{2}\right)^2 L, \qquad \varepsilon \to 0$
Grenze zur Ungenauigkeit	oder Abstand zu x^* $s_i = \lvert x_i - x^* \rvert$ $s_i \leq \frac{L_i}{2}$	$s_{i+1} \approx \frac{1}{2} s_i$
Konvergenzrate	 $s_i \to 0, für\ i \to \infty$ $s_{i+1} = \frac{1}{2} s^{(1)}$	$\to lineare\ Konvergenz$
Anzahl Iterationen i	um Ungenauigkeit L_i zu erreichen	$i = \log_2\left(\frac{L_0}{L_i}\right)$

1Dim unrestringierte Optimierungsaufgabe

Problemstellung (1DOP)	$x \in [a,b] \subset \mathbb{R}$ $f: \mathbb{R} \to \mathbb{R}$ $f \in C^2$ (ohne Restriktion/Nebenbedingungen)	
Gesucht	Dann ist das (globale) Minimum $f^* = Min\, f(x)$ $x \in [a,b]$	
Definition: lokales Minimum	$f_i^* = Min\, f(x), \quad \varepsilon > 0$ $x \in]x_i^* - \varepsilon, x_i^* + \varepsilon[$	
Beispiele	$Min(-x^2)$ $x \in [-1,2]$ $X^* = \{-1, +2\}$ $F^* = \{-1, -4\}$ $f^* = Min\, F^* = -4$	$Min\left(\frac{1}{x}\right)$ $x \in \mathbb{R}\{0\}$ $X^* = \emptyset$ $F^* = \emptyset$ 1DOP hat keine Lösung
	$Min(-\cos 2\pi x)$ $X^* \in \mathbb{Z}$ $F^* = \{-1\}$ $f^* = -1$ 1DOP hat ∞ viele Lösungen	$Min\, f(x), f \in C^2$ $x \in [a,b]$ f nicht explizit darstellbar!
Einfache **Lösung** von 1DOP	1) Bestimme X^*, F^* $X^* = \{x_1^*, x_2^* \dots, x_L^*\}$ $F^* = \{f_1^*, f_2^* \dots, f_L^*\}$ 2) Min $F^* = f^*$	

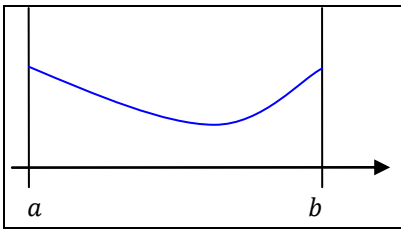
Spezialfälle

2.2.3 Polynome endlichen Grades

$Min\, P_n(x)$
$x \in [a,b]$

->Lösbar, da Polynomdivision im komplexen der ersten Ableitung muss n-1 Nullstellen haben (=Abbruchbedingung)

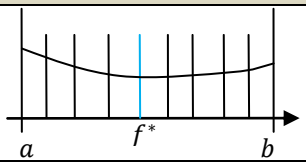
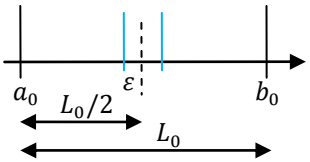
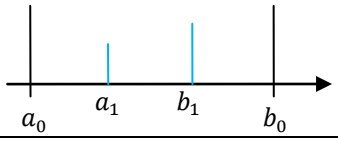
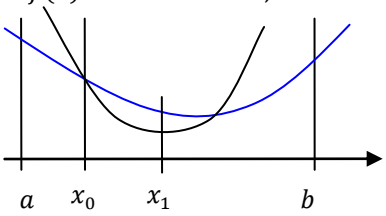
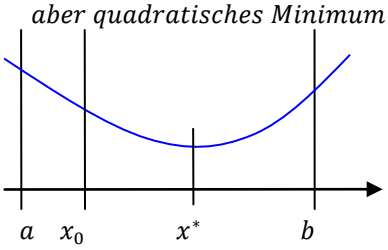
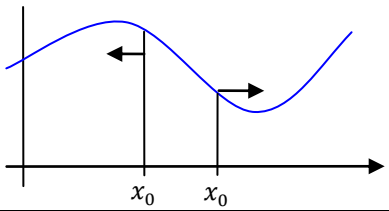
2.2.4 Konvexes Problem



$f \in C^2$
$f''(x) > 0$
$Min\, f(x)$
$x \in [a,b]$

lösbar -> genau eine Lösung
-> Existenz + Eindeutigkeit sichergestellt

2.2.5 Numerische Verfahren zur Lösung des 1DOP

Verfahren	Idee	Skizze	Eigenschaften	$f \in C^0, konvex$ $x \in [0,1]$
2.2.5.1 **Abdeckungs-verfahren**	Gitter rüber legen, min bestimmen $x^* \pm \delta$		ineffizient	
2.2.5.2 **Bijektion**	**Mit zwei Stützstellen, entscheiden in welchem Intervall die Lösung liegt**	**Dichotome Suche** 	Die Hälfte fällt pro Iteration weg	$N_f(i) = 2 + 2\,i$ $x^* \; mit \; L_i = 10^{-12}$ $i = 10^{12}$ $i = \log_2 10^{12} \approx 40$
	Nur eine Stützstelle legen letzte Auswertung wiederverwenden	**Goldener Schnitt** 	Ein Drittel fällt pro Iteration weg effizienter	$x^* \; mit \; L_i = 10^{-12}$ $L_{i+1} = L_i * 0.618$ $L_i = 0.618^i * L_0$ $i = 57$ $N_f(i) = 3 + i = 60$
2.2.5.3 **Das Newton Verfahren**	Startpunkt x_0 bestimmen Funktionswert Erste Ableitung Zweite Ableitung Minimum schätzen (x_1) Beginne wieder von oben	**Für Parabeln** $f(x) = ax^2 + bx + c, \quad a > 0$ 	über Taylor-Entwicklung in der Ersten Ableitung $f(x^*) = 0 = f'(x_0) + (x^* - x_0)\, f''(x_0) + \cdots$ $-f'(x_0) = (x^* - x_0)\, f''(x_0)$ $x^* = x_0 - \frac{f'(x_0)}{f''(x_0)}$	
	man benötigt: $f \in C^2$ Abbruchbedingungen a. $\lvert x_{i+1} - x_i \rvert \le \varepsilon_x$ b. $\lvert f_{i+1} - f_i \rvert \le \varepsilon_f$ c. $\lvert f_i' \rvert \le \varepsilon_{f'}$ d. $i \le i_{max}$	**Verallgemeinerung** $f(x) = keine \; Parabell$ $aber \; quadratisches \; Minimum$ 	$f(x^* + \Delta x) = f(x^*) + \frac{\Delta}{2} x^2 f''(x^*)$ Abstände schrumpfen quadratisch $x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}, \qquad f \in C^2, konvex$ $s_{i+1} \approx \lvert \alpha \rvert \; s_i^2$ 1) in einem Konvexen Gebiet konvergiert gegen Minimum 2) sehr schnell, da quadratische Konvergenz 3) konvergiert in gemischt konvex/konkaven Gebieten gegen Minima/Maxima/ Sattelpunkt	
2.2.5.4 **Modifizierter Newton**		**Verhalten bei Suche Min:** 	**Reparaturmechanismus** da Newton bei konkavem Gebiet gegen Maximum geht: $x_{i+1} = x_i - \frac{f_i''}{\lvert f_i'' \rvert}$	
2.2.5.5 **Quasi-Newton**	In der Praxis: f_i' ist schwierig f_i'' ist sehr schwierig Deshalb: Taylor	Vorteile: keine zweite, sondern erste Ableitung vom vorhergehender Schritt Nachteile: ungenauer, zwei Schritte in die Vergangenheit (ist aber gratis)	$f_i'' \approx \frac{f_{i-1}' - f_i'}{x_{i-1} - x_i}$ $x_{i+1} = x_i - \frac{f_i'}{f_{i-1}' - f_i'}(x_{i-1} - x_i)$ **Finite Differenzen (falls f' nicht zugänglich)** $f'(x_i) = \lim_{\Delta x \to 0} \frac{f(x_i + \Delta x) - f(x_i)}{\Delta x}$ Probleme: Δx zu gross ist schlecht (Rauschen), zu klein ist auch schlecht (Fehler)	

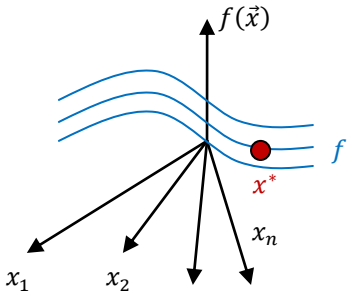
2.3 N-Dimensionalen unrestringierte Optimierungsaufgabe

2.3.1 Definition und Eigenschaften

$$Min\, f(\vec{x}), \qquad \vec{x} \in \mathbb{R}^N, \qquad f: \mathbb{R}^N \to \mathbb{R}, \qquad x \in [\vec{a}, \vec{b}]$$

- ohne Restriktion

Geometrisch:



$n+1$ dim-Raum

im Allgemeinen Algorithmisch nicht lösbar!
(falls ich nichts weiter über f bekannt)

2.3.2 Spezialfall: linear

$$Min(\vec{c}\vec{x}), \qquad x \in [\vec{a}, \vec{b}], \qquad \vec{x} \in \mathbb{R}^N$$

➔ f ist eine Hyper-Ebene (Ebene im höherdimensionalen Raum)
➔ eindeutige Lösung -> diese liegt auf einer Ecke!

2.3.3 Spezialfall: konvex

$$Min\, f(\vec{x}), \qquad \vec{x} \in \mathbb{R}^N, \qquad f \in C^2 \to \mathbb{R}, \qquad f\ konvex, \qquad x \in [\vec{a}, \vec{b}]$$

Hesse-Matrix

$$\underline{\underline{H}}(\vec{x}) = \begin{pmatrix} f_{x_1x_1}(\vec{x}) & f_{x_1x_2}(\vec{x}) & \dots & f_{x_1x_n}(\vec{x}) \\ f_{x_2x_1}(\vec{x}) & f_{x_2x_2}(\vec{x}) & & \\ \dots & & & \\ f_{x_nx_1}(\vec{x}) & & & f_{x_nx_n}(\vec{x}) \end{pmatrix}$$

➔ $n * n - Matrix$

Taylor-Reihe in $\mathbb{R}^N$

$$f(\overrightarrow{x_0} + \Delta\vec{x}) = \underbrace{f(\overrightarrow{x_0})}_{0.\,Ordnung} + \underbrace{\vec{\nabla} f(\overrightarrow{x_0}) * \overrightarrow{\Delta x}}_{1.\,Ordnung} + \underbrace{\frac{1}{2}\,\overrightarrow{\Delta x}\,\underline{\underline{H}}(\overrightarrow{x_0})\,\overrightarrow{\Delta x}}_{2.\,Ordnung} + \dots$$

➔ f ist konvex falls die Hessematrix positiv definit ist $f\ddot{u}r\ alle\ \vec{x} \in [\vec{a}, \vec{b}]$
➔ für f konvex: Lösung x^* existiert eindeutig!

2.3.4 Numerische Verfahren zur Lösung des n-Dim-OP

Verfahren	Idee	Skizze	Eigenschaften
2.3.4.1 Nelder-Mead Verfahren / Downhill Simplex	Simplex: 2D -> Dreieck 3D -> Tetraeder	(siehe Zusatzblätter)	nur lokale Lösungen clever
2.3.4.2 **Penalty-Funktionen** (Straffunktion)			einfach & anschaulich unstetige 1.Ableitungen nur Lösungsverfahren, die keine stetige 1.Ableitung fordern -> Downhill simplex

2.3.4.2 Penalty-Funktionen (Straffunktion)

Frage: Wie bekomme ich eine Optimierungsaufgabe in der Produkteentwicklung in eine gewünschte Form:

$$\min f(\vec{x}), x \in \mathbb{R}^N, x \in [a,b]$$

Werkzeug: Penalty-Funktion

am Beispiel Übung 1:

NLP

$$f(A_1, A_2) = 2.4\left(\sqrt{2}A_1 + A_2\right)$$

1) $A_1 \geq 14.1$
2) $A_2 \geq 10$
3) $A_2 \geq \frac{A_1}{0.2A_1 - 2\sqrt{2}}$

-> 3 Penalty Funktionen einführen (geom. eine Wand/Schüssel)

1) $$p_1(A_1, A_2) = \begin{cases} 0 & , A_1 \geq 14.1 \\ \alpha_1 (A_1 - 14.1)^2 & , A_1 < 14.1 \end{cases}$$

$$\alpha = Steilheit\ der\ Schüssel$$

2) $$p_2(A_1, A_2) = \begin{cases} 0 & , A_2 \geq 10 \\ \alpha_2 (A_2 - 10)^2 & , A_2 < 10 \end{cases}$$

3) $$p_3(A_1, A_2) = \begin{cases} 0 & , A_2 \geq \frac{A_1}{0.2A_1 - 2\sqrt{2}} \\ \left(A_2 - \frac{A_1}{0.2A_1 - 2\sqrt{2}}\right)^2 & , A_2 < \frac{A_1}{0.2A_1 - 2\sqrt{2}} \end{cases}$$

Neue Zielfunktion

$$\min \overline{f}(A_1, A_2) = \min\left(f(A_1, A_2) + p_1(A_1, A_2) + p_2(A_1, A_2) + p_3(A_1, A_2)\right)$$

-> Downhill Simplex

am Beispiel Übung 6:

DGL

$$x(t, v_1, v_2, v_3)$$
$$y(t, v_1, v_2, v_3)$$
$$z(t, v_1, v_2, v_3)$$

Optimaldesign

$$x(10, v_1, v_2, v_3) = 5$$
$$y(10, v_1, v_2, v_3) = 5$$
$$z(10, v_1, v_2, v_3) = 0$$

-> 3 Penalty Funktionen einführen

1) $p_1(v_1, v_2, v_3) = (x(10, v_1, v_2, v_3) - 5)^2$
2) $p_2(v_1, v_2, v_3) = (y(10, v_1, v_2, v_3) - 5)^2$
3) $p_3(v_1, v_2, v_3) = (z(10, v_1, v_2, v_3) - 0)^2$

$$\min \overline{f}(v_1, v_2, v_3) = \min\left(x(v_1, v_2, v_3) + y(v_1, v_2, v_3) + z(v_1, v_2, v_3)\right)$$

-> Downhill Simplex

2.3.4.3 Das Newton Verfahren in N-Dimensionen

Ges:

$$\min f(\vec{x}), x \in \mathbb{R}^N$$

$$\vec{\nabla} f(\vec{x}^*) = 0 = \vec{\nabla} f(\vec{x}_0) + \underline{\underline{H}}(\vec{x}_0) * (\vec{x}^* - \vec{x}_0)$$
$$0 = \underline{\underline{H}}^{-1}(\vec{x}_0) * \vec{\nabla} f(\vec{x}_0) + \vec{x}^* - \vec{x}_0$$
$$\vec{x}^* = \vec{x}_0 - \underline{\underline{H}}^{-1}(\vec{x}_0)\, \vec{\nabla} f(\vec{x}_0)$$
$$\vec{x}_{i+1} = \vec{x}_i - \underline{\underline{H}}^{-1}(\vec{x}_i) * \vec{\nabla} f(\vec{x}_i)$$

Eigenschaften

- iterativ
- falls $f(x)$ konvex ($H\ positiv\ definit$)
  dann konvergiert das Newton verfahren für alle $x_0\ gegen\ x^*$
- falls $f(x)$ nicht konvex, dann hängt das Ergebnis von x_0 ab
- Das Newton-Verfahren konvergiert gegen ALLE kritische Punkte $\vec{\nabla} f = 0$
- $f(x)$ muss 2-mal stetig differenzierbar sein!
- konvergiert gegen Minima und Maxima
- falls konvex -> konvergiert gegen Minima
- Newtonschritt = Richtung + Schrittweite = $\underline{\underline{H}}^{-1}(\vec{x}_i) * \vec{\nabla} f(\vec{x}_i)$
- Einzugsgebiete der Minima sind eher klein

2.3.5.3 Modifiziertes Newton-Verfahren

A) Newton Verfahren mit Schrittweitensteuerung

$$x_{i+1} = x_i - \lambda_{ij} * \underline{\underline{H}}^{-1}(\vec{x}_i) * \vec{\nabla} f(\vec{x}_i)$$
$$0 < \lambda_{ij} \leq 1, \qquad \lambda_{ij} \in \mathbb{R}$$

A1) backtracking (kleinere Schrittweite)

$$\lambda_{i1} = 1, \qquad falls\ f(x_i + 1) < f(x_i)$$
$$\lambda_{i2} = k^1 * (0.7^1)$$
$$\lambda_{ij} = k^{j-1} (0 < k < 1)$$

A2) Liniensuche (Golden Section Search)

B) Methode des steilsten Abstiegs

$$\underline{\underline{H}}^{-1}(\vec{x}_i) \approx \lambda * I$$
$$x_{i+1} = x_i - \lambda_{ij} * \vec{\nabla} f(\vec{x}_i)$$

Probleme:

- liefert keine Schrittweite, sondern nur Richtung
- ineffizient

ZF:

$$\min f(\vec{x}), x \in \mathbb{R}^N$$

Newton-Verfahren

Quasi-Newton

- BFGS oder DFP
- Golden Section Line Search
- (globale) Suche durch Variation des Startwerts

zugänglich: Bibliotheken SW z.B. Numerical Reics
Mathe-Programme: (octare, MATLAB, Matematica

Downhill-Simplex

2.4 Suche im n-dimensionalen Raum mit Restriktionen (Nebenbedingungen)

2.4.1 Definition NLP

Optimierungsaufgabe mit Nebenbedingungen (engl. Nonlinear Programming(NLP))

Zielfunktion	$\min f(\vec{x}), x \in \mathbb{R}^N$
n_1 Ungleichheitsnebenbedingungen (Ungleichheitsrestriktion)	$g_1(x) \geq 0$ … $g_{n_1}(x) \geq 0$
n_2 Gleichheitsnebenbedingungen (Gleichheitsrestriktion)	$h_1(x) = 0$ $h_{n_2}(x) = 0$

Bemerkung

Typischerweise fordert man: $f, g_i, h_i \in C^2(zweimal\ stetig\ differentierbar)$

Gleichheits-NB können in 2 Ungleichheits-NB gewandelt werden.

wenn n und m klein, können die NB über Penaltyfunktionen in die Zielfunktion kodiert werden

Beispiel 1

$$\min f(\vec{x}) = x_1^2 + x_2^2, \qquad unter\ x_1 + 2\,x_2 + 1 = 0$$

-> keine Ungleichheits-NB

-> 1 Gleichheits-NB

Lösung über:

- einsetzen
- Lagranche

Beispiel 2 (Two-Bar)

$$NLP$$
$$\min f(A_1, A_2) = \sqrt{2}A_1 + A_2$$
$$A_1 \geq 14.1$$
$$A_2 \geq 10$$
$$A_2 \geq \frac{A_1}{(\sqrt{2}A_1 - 10)}$$

-> 3 Ungleichheits-NB

2.4.2 Lösungsideen

- SQP-Verfahren (Sequential Quadratic Programming)

S1: Langrage-Funktion

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{n_1} \lambda_i g_i(x) + \sum_{i=1}^{n_2} \mu_i h_i(x)$$
$$L: \mathbb{R}^{N+n_1+n_2} \to \mathbb{R}$$

S2: suche kritische Punkte von L

$$\vec{\nabla} L = \vec{0}$$

Quasi-Newton-Verfahren mit Anfangswert x_0

-> Lösungskandidat (x^*, λ^*, μ^*)

anschaulich:

$n_2 = 0\ (nur\ Ungleichheitsbedingungen)$

Fläche im Hochdimensionalen Raum

Wird durch Ungleichheitsbedingungen abgeschnitten -> ergibt „feasible region“ (erlaubtes Gebiet)

Fall 1: $x^*\ am\ Rand\ von\ F$

$$\to \lambda_i \neq 0\ (aktiv)$$
$$\to alle\ anderen\ \lambda = 0\ (nicht\ aktiv)$$

Fall 2: $x^*\ liegt\ im\ Inneren\ von\ F$

$$\to \lambda_i = 0$$

Eigenschaften

- ➔ eventuell gibt es lokale Minima → testen durch Variation der Anfangswerte x_0
- ➔ x_0 kann auch ausserhalb von F liegen

MATLAB

	`options = optimset('Display', 'iter', 'PlotFcns', @optimplotx);`
Downhill Simplex	`[x1,fval1] = fminsearch(@funcA1, [-29,-72,57], options);`
Quasi-Newton Verfahren	`[x2,fval2] = fminunc(@funcA1,[-29,-72,57], options);`
N-Dimensionaler Newton	`[x3,i,x] = nNewton([-29;-72;57],10e-6);` (Selbstprogrammierter Algorithmus)