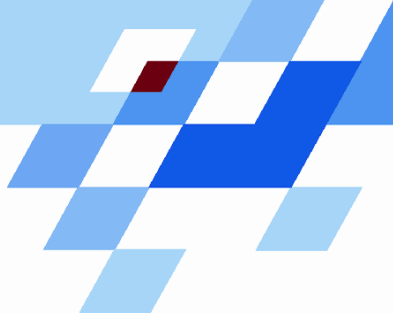


Wintersemester 2006/07

Fundamente der Computational Intelligence

(Vorlesung)

Prof. Dr. Günter Rudolph

Fachbereich Informatik

Lehrstuhl für Algorithm Engineering

Inhalt

- EA im $\mathbb{R}^n$
- Schrittweitensteuerungen

Schrittweitensteuerung nach Rechenberg:

Individuum (x, σ) $\qquad \gamma \in (0,1) \subset \mathbb{R}$

$$\sigma^{(k)} = \begin{cases} \sigma^{(k - \Delta k)} / \gamma \text{ , falls } \dfrac{\text{\# Verbesserungen}}{\text{\# Mutationen}} > 1/5 \text{ während } \Delta k \text{ Mutationen} \\ \sigma^{(k - \Delta k)} \cdot \gamma \text{ , sonst} \end{cases}$$

Problem: keine Konvergenz mit W'keit 1

aber: schnelle Konvergenz zum lokalen Optimum + W'keit > 0 dieses zu verlassen!

$\Rightarrow$ kein globales Verfahren, aber gutes nicht-lokales Verhalten!

Beobachtung: Anpassung σ sprunghaft $\Rightarrow$ Anpassung kontinuisieren!

Schrittweitensteuerung nach Schwefel:

Individuum (x, σ) : auch Strategieparameter wie σ werden mutiert

Mutation:

1. $\sigma_{k+1} = \sigma_k \cdot \exp(N(0, \tau))$ $\qquad \tau = 1 / n^{1/2}$
2. $X_{k+1} = X_k + \underbrace{\sigma_{k+1}} \cdot N(0, I)$

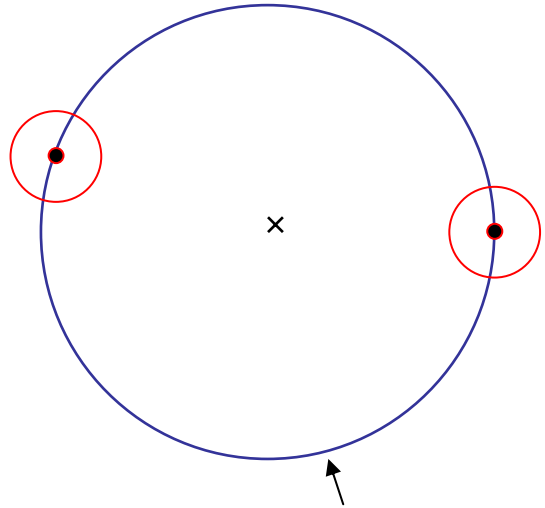
Wichtig: die bereits mutierte Schrittweite wird verwendet!

„Schrittweite“ σ wird multiplikativ verändert (logarithmisch normalverteilt),

neue Schrittweite wird verwendet bei additiver Veränderung der Position

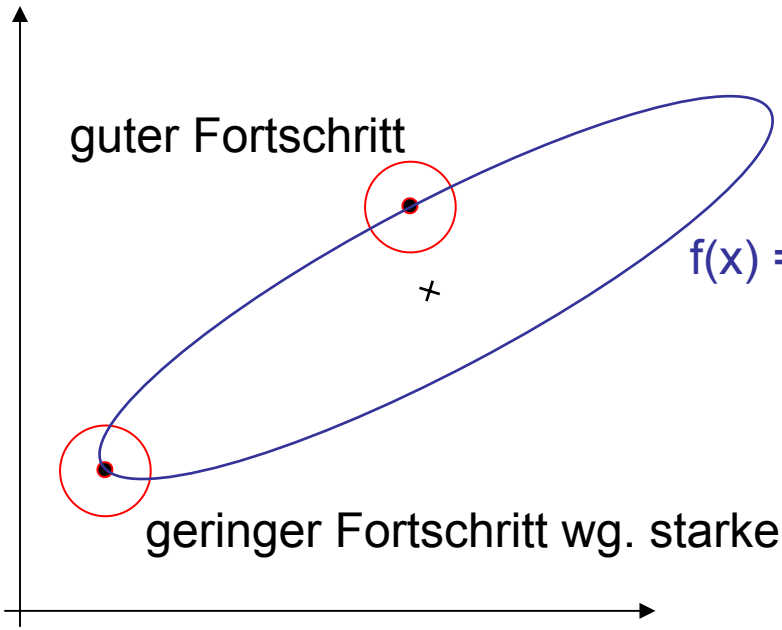
Bisher: rotationssymmetrische Mutationen

Isolinie gleicher W‘keitsdichte

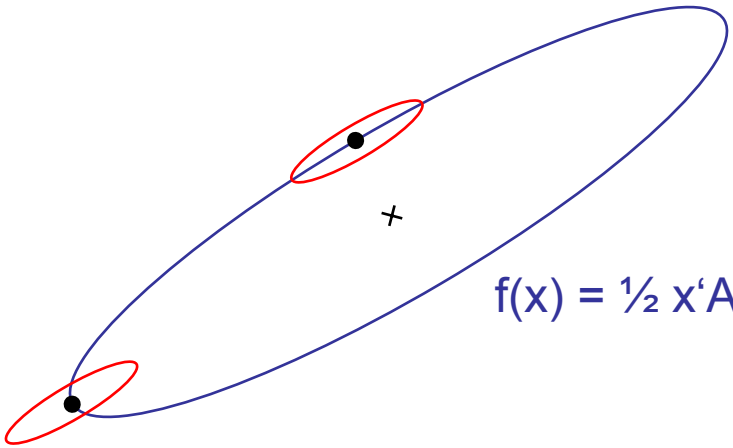


Isolinie { x : f(x) = a }
falls f(x) = || x ||^2

(Höhenlinie)



Idee:
Mutationsverteilung wie Höhenlinien ausrichten



Wie erzeugt man solche Mutationsverteilungen?

$Z \sim N(0, \sigma^2 I_n)$ $\Rightarrow$ rotationssymmetrisch (I_n = Einheitsmatrix mit Rang n)

$Z \sim N(0, \mathbf{D}^2)$ $\Rightarrow$ ellipsoid, achsenparallel ($\mathbf{D}$ = diag(σ_1, ..., σ_n), Diagonalmatrix)

$Z \sim N(0, \mathbf{C})$ $\Rightarrow$ ellipsoid, frei beweglich ($\mathbf{C}$ = Kovarianzmatrix)

$\mathbf{C} = \mathbf{C}^t$ (symmetrisch) und $\forall x: x^t\mathbf{C}x > 0$ (positiv definit)

Wie muss Kovarianzmatrix C gewählt werden?

Ansatz: Taylor-Reihenentwicklung

$$f(x + h) = f(x) + \underbrace{h' \nabla f(x)}_{\text{linear}} + \tfrac{1}{2} \underbrace{h' \nabla^2 f(x)\, h}_{\text{quadratisch}} + R(x, h)$$

Restterme (ignorierbar, da h klein)

$\nabla^2 f(x) = H(x)$ **Hessematrix**

→ enthält Informationen über Skalierung und Orientierung der Höhenlinien

→ Es wird sich zeigen: Wähle $\mathbf{C} = H^{-1}$!

Approximation: $f(x) \approx \frac{1}{2}\, x'Ax + b'x + c$ $\Rightarrow$ Hessematrix $H = A$

Koordinatentransformation: $y = Q\,x$ Q: (n x n) - Matrix

$$\Rightarrow f(Qx) = \frac{1}{2}\,(Qx)'\,A\,(Qx) + b'\,(Qx) + c$$

$$= \frac{1}{2}\, x'Q'AQx + b'Qx + c$$

$$= \frac{1}{2}\, x'Q'B'BQx + b'Qx + c$$

mit Cholesky-Zerlegung $A = B'B$

$$= \frac{1}{2}\, x'(Q'B')(BQ)x + b'Qx + c$$

sei jetzt $Q = B^{-1}$

$$= \frac{1}{2}\, \underbrace{x'x} + b'\, B^{-1}x + c$$

rotationssymmetrische Höhenlinien!

also: wir benötigen Dreiecksmatrix Q bzw. B^{-1}

Satz:

Sei $y \sim N(0, I_n)$ und Q‘Q eine positiv definite Matrix mit Rang n.

Dann $x = Q‘y \sim N(0, Q‘Q)$.

$\Rightarrow$ mit $Q = B^{-1}$ können wir Mutationsverteilungen wie gewünscht ausrichten!

aber: woher bekommen wir Matrix Q?

$\Rightarrow$ Selbstanpassung der Matrixelemente wie bei Schrittweite nach Schwefel

da $H = A = B‘B$, ist $H^{-1} = (B‘B)^{-1} = B^{-1}(B^{-1})‘ =_{def} C = Q‘Q$

Q entsteht durch Cholesky-Zerlegung von C, ist also Dreiecksmatrix

$\rightarrow$ Skalierungsfaktoren je Zeile herausziehen: in Diagonalmatrix S ablegen

$\rightarrow$ Q zerlegbar in $Q = S \cdot T$ mit $t_{ii} = 1$ (S hat n Parameter, T hat n(n-1)/2 Parameter)

Satz:

Jede sym., pos. definite Matrix A ist zerlegbar via A = T‘DT und umgekehrt,

wobei T orthogonale Matrix (T‘ = T^{-1}) und D Diagonalmatrix mit $d_{ii} > 0$.

$\Rightarrow$ also wählen wir S = $D^{1/2}$, so dass A = (TS)‘(TS)

Satz:

Jede orthogonale Matrix T kann durch das Produkt von n(n-1)/2 elementaren Rotationsmatrizen R_{ij} (ω_k) dargestellt werden:

$$T = \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} R_{ij}(\omega_k)$$

$R_{ij}(\omega)$ = wie Einheitsmatrix, jedoch mit $r_{ii} = r_{jj} = \cos \omega$, $r_{ij} = -r_{ji} = -\sin \omega$

Geometrische Interpretation

durch Q‘y = TSy wird rotationssymmetrischer Zufallsvektor y

1. zunächst achsenparallel skaliert via Sy
2. und dann durch n(n-1)/2 elementare Rotationen in gewünschte Orientierung gebracht via T(Sy)

Mutation der Winkel ω :

$\omega^{(t+1)} = (\omega^{(t)} + W + \pi) \bmod (2\pi) - \pi \qquad \in (-\pi, \pi]$

wobei $W \sim N(0, \kappa^2)$ mit $\kappa = 5° \pi / 180°$

$\rightarrow$ Individuum jetzt: (x, σ, ω) mit n Schrittweiten (Skalierungen) + n(n-1)/2 Winkel

Praxis zeigt:
Idee gut, aber Realisierung nicht gut genug (funktioniert nur für kleines n)

Wie könnte man sonst noch an Matrixelemente von Q kommen? (Rudolph 1992)

Modellannahme: $f(x) \approx \frac{1}{2}\, x'Ax + b'x + c$

Beobachtung: Bei $(\mu \overset{+}{,} \lambda)$ – Selektion werden λ Paare $(x, f(x))$ berechnet.

$\Rightarrow$ Falls $\lambda > n(n-1)/2 + n + 1$, dann **überbestimmtes** lineares Gleichungssystem:

$$\left.\begin{aligned} f(x_1) &= \tfrac{1}{2}\, x_1'Ax_1 + b'x_1 + c \\ &\vdots \\ f(x_\lambda) &= \tfrac{1}{2}\, x_\lambda'Ax_\lambda + b'x_\lambda + c \end{aligned}\right\} \quad \begin{array}{l} v = (A, b, c) \text{ hat } n(n-1)/2 + n + 1 \text{ zu} \\ \text{schätzende Parameter, wobei } A = B'B \end{array}$$

$\Rightarrow$ multiple lineare Regression für $f = Xv \;\rightarrow\; X'f = X'Xv \;\rightarrow\; (X'X)^{-1}X'f = v$

$\Rightarrow$ aus Schätzer $v = (A, b, c)$ bekommen wir Hessematrix $H = A$

$\Rightarrow$ Cholesky-Dekomposition von H und Matrixinversion liefert Q

Praxis zeigt: funktioniert sehr gut, aber zu hoher Aufwand: $(X'X)^{-1}$ kostet $\mathcal{O}(n^6)$

Idee: Matrix C nicht in jeder Generation schätzen, sondern iterativ nähern!

(Hansen, Ostermeier et al. 1996ff.)

→ **C**ovariance **M**atrix **A**daptation **E**volutionary **A**lgorithm (CMA-EA)

Setze initiale Kovarianzmatrix auf $C^{(0)} = I_n$

$$C^{(t+1)} = (1-\eta)\, C^{(t)} + \eta \sum_{i=1}^{\mu} w_i\, d_i\, d_i'$$

η : „Lernrate“ $\in (0,1)$

$$m = \frac{1}{\mu} \sum_{i=1}^{\mu} x_{i:\lambda}$$

Mittelpunkt aller selektierten Eltern

Aufwand: $\mathcal{O}(\mu n^2 + n^3)$

$d_i = (x_{i:\lambda} - m) / \sigma$

Sortierung: $f(x_{1:\lambda}) \le f(x_{2:\lambda}) \le \ldots \le f(x_{\lambda:\lambda})$

dyadisches Produkt: dd‘ =
$$\begin{pmatrix} d_1 d_1 & d_1 d_2 & \cdots & d_1 d_\mu \\ d_2 d_1 & d_2 d_2 & \cdots & d_2 d_\mu \\ \vdots & & & \vdots \\ d_\mu d_1 & d_\mu d_2 & \cdots & d_\mu d_\mu \end{pmatrix}$$

ist positiv semidefinite Streuungsmatrix

Variante:

$m = \frac{1}{\mu} \sum_{i=1}^{\mu} x_{i:\lambda}$ Mittelpunkt aller selektierten Eltern

$p^{(t+1)} = (1 - \chi)\, p^{(t)} + (\chi\,(2 - \chi)\,\mu_{eff})^{1/2}\,(m^{(t)} - m^{(t-1)}) / \sigma^{(t)}$ „Evolutionspfad“

$p^{(0)} = 0$ $\chi \in (0,1)$

$C^{(0)} = I_n$

$C^{(t+1)} = (1 - \eta)\, C^{(t)} + \eta\, p^{(t)}\,(p^{(t)})'$

Aufwand: $\mathcal{O}(n^2)$

→ Cholesky-Zerlegung: $\mathcal{O}(n^3)$ für $C^{(t)}$

State-of-the-art: CMA-EA

→ erfolgreiche Anwendungen in der Praxis

→ insbesondere wenn Zielfunktionsauswertung zeitaufwändig

(z.B. Zielfunktionsauswertung durch Simulationsprogramm)

Implementierungen im WWW verfügbar