

ADS: Algorithmen und Datenstrukturen 2

11. Teil

Peter F. Stadler & Konstantin Klemm

Bioinformatics Group, Dept. of Computer Science & Interdisciplinary Center for
Bioinformatics, **University of Leipzig**

15. Juni 2011

Stochastische Optimierungsalgorithmen

Problemstellung

Lösungsmenge X , Bewertungsfunktion $f : X \rightarrow \mathbb{R}$.

Finde globales Minimum von f

Grundprinzip

- Start: endliche Menge A von erlaubten Lösungen (“Population”) mit $|A| = n$.
- Schleife
 - (1) Erzeuge erweiterte Lösungsmenge B , $|B| = m > n$.
 - (2) Selektiere eine neue Population $A' \subset B$ mit $|A'| = n$ in der die besseren Lösungen angereichert sind.
- Abbruchbedingung

Stochastische Optimierungsalgorithmen

Motivation durch Prozesse in der Natur:

Evolution als Wechselspiel von Mutation (allergemeiner: Erzeugung von Variabilität und Selektion (aus den vorhandenen Varianten)

Schrittweise Energie-Minimierung in physikalischen Systemen (z.B. Gefrieren von Flüssigkeiten ...)

Random Generate and Test

- $A = \{x\}$, eine einzelne Lösung
- $B = \{x, y\}$, wobei $y \in X$ eine **zufällig** gezogene Lösung ist.
- $A' = \{y\}$ if $f(y) < f(x)$, sonst $A' = \{x\}$.

Keine Garantie, dass ein globales Minimum gefunden wird.

FRAGE: wie kann man die *Struktur* von f ausnutzen?

IDEE: Gute Lösungen werden "irgendwie" ähnlich zu anderen guten Lösungen sein.

“Moves”

Lasse nur bestimmte “Schritte” zu.

Einfachster Fall: $A = \{x\}$

Formal: Definiere eine (erlaubte) Nachbarschaft $N(x)$ für jedes $x \in X$.

Nun wähle ein $y \in N(x)$ mit Wahrscheinlichkeit $1/|N(x)|$.

Impliziert eine Graph-Struktur Γ auf X : $(x, y) \in E$ iff $y \in N(x)$.

Notwendige Bedingung: Γ muss stark zusammenhängend sein
sonst kann der Prozess auf einer Zusammenhangskomponente
eingesperrt bleiben, die kein globales Minimum enthält.

Fitness-Landschaften

Formale Beschreibung des Optimierungsproblems als Funktion über der Knotenmenge des Graphen Γ

Die Frage nach geometrischen Eigenschaften wie

- Zahl von lokalen Minima
- Tiefe und Breite von Tälern
- Sattelpunkte zwischen Tälern
- Rauheits-Maße

und deren Einfluss auf die Optimierung ist Gegenstand der Forschung

Veränderung des Move-Sets verändert die Landschaft und damit das Optimierungsverhalten.

Suche problemangepasste Movesets. Idee: einzelne Moves sollten die Qualität der Lösung (meistens) nicht zu drastisch verändern.

Adaptive Walk

- **Start:** wähle $x \in X$ zufällig
- **Schleife:**
 - (1) wähle $y \in N(x)$ gleichverteilt
 - (2) falls $f(y) < f(x)$, setze $x \rightarrow y$.
- **Abbruchbedingung**

Folge der Lösungen entspricht einem Pfad in Γ , entlang dem f strikt abnimmt. (“Adaptive Walk”)

Lokales Minimum: $f(y) \geq f(x)$ für alle $y \in N(x)$.

Pfade bleiben in lokalen Minima stecken. Bei geeigneter Abbruchbedingung erreichen alle Pfade ein lokales Minimum.

PROBLEM: Lösungen können sehr schlechter Qualität sein, d.h. $f(x) \gg f(\min)$.

Gradient Descent

- **Start:** wähle $x \in X$ zufällig
- **Schleife:**
 - (1) bestimme die Menge $Z(x)$ aller $z \in N(x)$ für die $f(z)$ minimal über $N(x)$ ist.

wähle $y \in Z(x)$ gleichverteilt
 - (2) falls $f(y) < f(x)$, setze $x \rightarrow y$.
- **Abbruchbedingung:** stop falls $f(y) \geq f(x)$.

Folge der Lösungen entspricht einem Pfad in Γ , entlang dem f so steil wie möglich abnimmt. (Weg des steilsten Abstieg, "Gradient Descent").

Endet immer in einem lokalen Minimum.

Teurer als adaptive walk, weil $N(x)$ durchsucht werden muss.

Metropolis-Walks

Um lokalen Minima zu entkommen, lasse auch “aufwärts” Schritte zu:

- **Start:** wähle $x \in X$ zufällig
- **Schleife:**
 - (1) wähle $y \in N(x)$ gleichverteilt
 - (2) falls $f(y) < f(x)$, setze $x \rightarrow y$

sonst setze $x \rightarrow y$ mit Wahrscheinlichkeit

$$\exp(-(f(y) - f(x))/T)$$

- **Abbruchbedingung**

Steil bergauf = exponentiell selten

Eigenschaft: Wenn man **lange** genug wartet, besucht man zu einem bestimmten Zeitpunkt t die Lösung x mit Wahrscheinlichkeit

$$p(x) = \exp(-f(x)/T)/Z$$

Für kleine T : Lösungen mit kleinen Funktionswerten werden angereichert.

Simulated Annealing

IDEE: Reduziere die “Temperatur” T im Laufe der Simulation
Kirkpatrick, Gelatt and Vecchi (1983) and V. Černý (1985)

WICHTIGE ROLLE:

“cooling schedule” T_k , typischerweise eine monoton fallende
Funktion der Schrittzahl k

Der Algorithmus konvergiert falls $T_k \geq d/\log(k)$ für große k .

Das ist in der Praxis aber zu langsam, man braucht exponentiell
viele Schritte k um eine gegebene Genauigkeit $|f(x) - f(\min)|$ zu
erreichen.

Anwendung auf harte Probleme, für die keine effizienten
(polynomialen) Algorithmen bekannt sind.

Genetische Algorithmen

Cross-Over: “mischen” von 2 Eltern.

$\mathcal{R}(x, y)$ Menge der möglichen Kinder der Eltern x und y

Konstruktion der erweiterten Population:

- 1 Wähle eine Menge von Eltern-Paaren
- 2 Für jedes Elternpaar (x, y) ziehe einen, zwei oder mehrere Nachkommen $z \in \mathcal{R}(x, y)$.
- 3 Wähle eine Teilmenge von A und ziehe für jedes $x \in A$ einen Mutanten $y \in N(x)$.
- 4 Bilde die Vereinigung von A , den Cross-over Nachkommen, und den Mutanten

Genetische Algorithmen

Selektion:

viele Varianten:

- fitness-proportional selection.
Ziehe aus B mit Wahrscheinlichkeiten proportional zu $-f(z)$ oder $\exp(-f(z))$ bis genau n Lösungen gezogen sind
- ziehe zwei Lösungen z und z' . Falls $z > z'$, akzeptiere z .
Wiederhole bis n Lösungen akzeptiert sind.
- Wähle die besten n Lösungen.
- ...

Die Eingeweide: Kodierung

Explizite Representation der Lösungen

- Strings fixer Länge.
Moves: Ersetzen eines Zeichens
 $N(x) = \{y \mid d_H(x, y) = 1\}$ $d_H \dots$ Hamming Distanz
- Permutationen.
Moves: Transpositionen
- Reell-wertige Vektoren.
Hier werden oft normalverteilte “Mutanten” gewählt:

$$p_{yx} \sim \exp(-\|x - y\|^2 / \sigma^2)$$

- auch andere kombinatorische Objekte können benutzt werden:
Spannbäume, Partitionen, Matchings, ...

Cross-Over

Für Strings:

1-Punkt Crossover

$x_1 = 0100001010100010101010101101$

$x_2 = 1010101110101010011100110101$

Bestimme einen zufälligen *Bruchpunkt* k und vertausche die Suffixe:

$x'_1 = x_1[1..k]x_2[k + 1..n]$ und $x'_2 = x_2[1..k]x_1[k + 1..n]$

Für $k = 7$

$x'_1 = 0100001.110101010011100110101$

$x'_2 = 1010101.010100010101010101101$

Cross-Over

Uniformer Crossover

übernimm jede Position zufällig entweder unverändert oder vertausche die beiden Zeichen zwischen Eltern

x1 = 0110001110101010101100100101

x2 = 1000101010100010011010111101

tausch ---**--***---*****---**---***--

Beobachtung: $d(x_1, x'_1) \leq d(x_1, d_2)$ und $d(x_1, x'_2) \leq d(x_1, d_2)$

Übliche Forderung an Rekombination:

- $\mathcal{R}(x, y) = \mathcal{R}(y, x)$
- Für $z \in \mathcal{R}(x, y)$ gilt $d(x, z), d(y, z) \leq d(x, y)$

Crossover für Vektoren

Mittelwertbildung:

$$z = \frac{1}{2}(x + y)$$

Konvexe Kombination:

$$z = px + (1 - p)y \quad p \in [0, 1]$$

Die Zufallszahl p wird typischerweise aus einer Gleichverteilung gezogen

Spezielle Verfahren für andere Typen von Objekten, wie z.B.,
Permutationen, Bäume ...

Crossover im Allgemeinen

Gegeben: Suchraum X eine Graphenstruktur auf X , d.h.,
Nachbarschaften $N(x)$ für jede Konfiguration so dass
 $y \in N(x) \implies x \in N(y)$.

Die graphische Distanz $d : X \times X \rightarrow \mathbb{R}_0^+$ auf (X, N) ist definiert
durch

$d(x, y)$ ist die Länge des kürzesten Pfades von x nach y .
Segment $[x, y]$ zwischen x und y ist definiert durch

$$[x, y] = \{z \in X \mid d(x, z) + d(z, y) = d(x, y)\}$$

(=Menge aller Konfigurationen auf kürzesten Pfaden von x nach y)
Sogenannte *topologische Crossover-Operator* erfüllen

$$\mathcal{R}(x, y) \subseteq [x, y]$$

D.h., jede Rekombinanten liegt auf einem kürzesten Pfad zwischen
den Eltern

Abbruchbedingungen

- Normalerweise ist nicht bekannt, ob das Ziel bereits erreicht ist, d.h., ob ein globales Minimum bereits gefunden ist.
- Fixe Anzahl von Iterationen
- Keine Verbesserung seit K Iterationen
- Verbesserung in den letzten K Iterationen weniger als ϵ (im Fall von reell-wertigen Problemstellungen)

Schwarmoptimierung

PSO = “particle swarm optimization”

funktioniert auf Vektorräumen, i.e., $f : \mathbb{R}^n \rightarrow \mathbb{R}$.

Partikel i durch Ort \vec{x}_i und Geschwindigkeit \vec{v}_i bestimmt.

Ausgezeichnetes Positionen \vec{p}_i der besten bisherigen Position von Partikel i

Beste bisherige Lösung \vec{g} .

- 1 update $\vec{v}_i \leftarrow \omega \vec{v}_i + \xi_1 r_p (\vec{p}_i - \vec{x}_i) + \xi_2 r_g (\vec{g} - \vec{x}_i)$ mit Zufallszahlen $\xi_1, \xi_2 \in U[0, 1]$ und Parametern ω, r_p, r_g .
Trägheit $\omega \vec{v}_i$, Zug zu den besten Positionen
- 2 update $\vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i$
- 3 update $\vec{p}_i \leftarrow \vec{x}_i$ if $f(\vec{x}_i) < f(\vec{p}_i)$.
- 4 update \vec{g} als bestes der \vec{p}_i .