Evolutionäre Algorithmen

Vorlesung 8

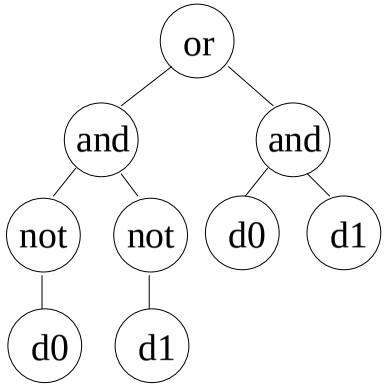
Genetisches Programmieren
Classifier-Systeme

Genetisches Programmieren

- Hauptoperator: Rekombination
- Hintergrundoperator: Mutation
- Repräsentation: hat keine feste Größe
- Populationsgröße: meist sehr groß (bis zu 10000)

Syntaxbäume

- ursprünglich: S-Expressions (LISP), Syntaxbäume



Repräsentation

- Auswertung auf einer virtuellen Maschine
- Abbruch bei zu langer Laufzeit
- unterschiedliche Typen?
- beliebig große Individuen?
- Operatoren auf unterschiedlich großen Individuen?

Implementation

- Speicherung durch Zeiger und dynamisch zugewiesenen Speicherplatz
  $\Rightarrow$ aufwändige Verwaltung und Zugriff
- Speicherung in Präfix-Notation

  or and not d0 d1 if not d1 d2 $<$ 5 d0

  $\Rightarrow$ einfache Auswertung durch Stack-Maschine

Atomare Operationen

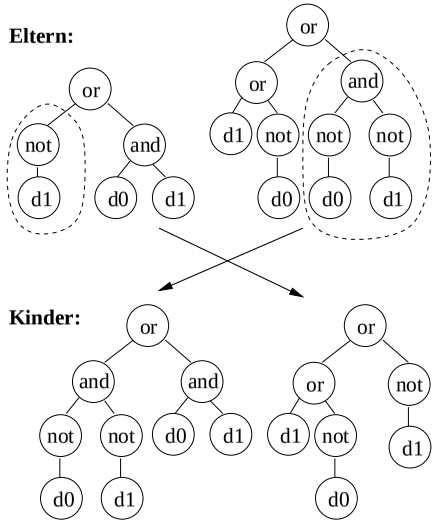
- Entferne(T,i): entferne aus Baum T den Teilstring der linearen Darstellung, der dem Unterbaum mit der Wurzel an Position i entspricht – an der Position i verbleibt ein Platzhalter
- Teilbaum(T,i): liefert den Teilstring des Unterbaums, der in T an Position i beginnt
- Erzeugebaum(): erzeugt einen beliebigen zufälligen, aber konsistenten Teilbaum

Atomare Operationen

- Einfügen(T,i,U): fügt in T den Unterbaum U anstelle des Knotens an Position i ein
- Enthalten(T,i,j): prüft in T ob Knoten j im Unterbaum beginnend mit i enthalten ist.

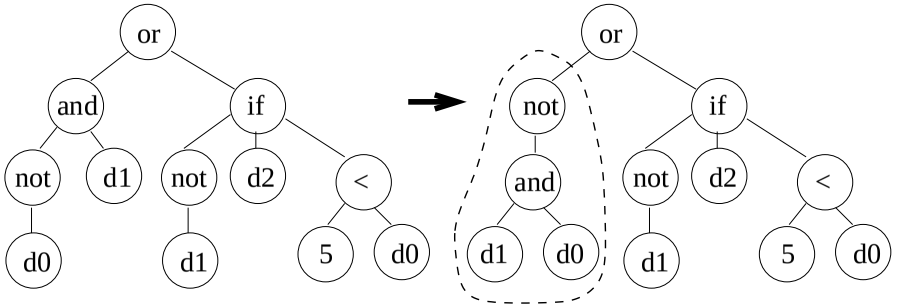
Rekombination

- vertauschen von Unterbäumen
- Typkonsistenz?
- Baumgröße?



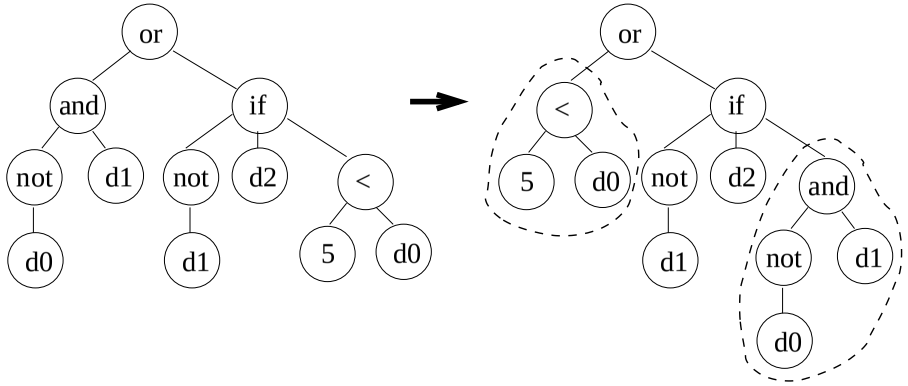
Mutation

- Ersetzen von Teilbäumen durch zufällige Bäume



Mutation

- interne Rekombination



Initialisierung

- Vielfalt nicht nur bezüglich einzelner Gene, sondern auch bezüglich Struktur
- mögliche Strategie: $2(d-1)$ gleich große Fraktionen
  - Tiefe: $d' \in \{2, \ldots,$ maximale Tiefe$\}$
  - vollständige Bäume vs. Bäume wachsen lassen

Gesamtablauf

- sowohl GA- als auch ES-Ablauf möglich, auch: q-fache Turnierselektion
- Populationsgröße: meist wenigsten 500 – 5000
- Anwendung der Operatoren:
  - zunächst Rekombination, dann Mutation
  - getrennt: z.B. 80% der Nachkommen mit Rekombination, 10% mit Mutation, 10% unverändert kopiert

Fortgeschrittene Operatoren

- Encapsulation
  - $\Rightarrow$ einzelne Unterbäume in einem Terminal zusammenfassen
  - $\Rightarrow$ ist umstritten
- Unterprogramme (ADFs)
  - vorgegebene Anzahl von Unterprogrammen mit vorgegebener Stelligkeit
  - Unterprogramme werden im Hauptprogramm benutzt
  - Rekombination nur zwischen Unterprogrammen bzw. Hauptprogrammen

„Junk DNA“ bzw. Introns

- Individuen besitzen irrelevante Teile („$a + (1 - 1)$“ oder „$if\ 2 < 1\ then\ \ldots$“)
- Grund hierfür: Operatoren sind eher destruktiv, daher haben güteneutrale Veränderungen Vorteile bei der Selektion
- Introns wachsen oft exponentiell (gegen Ende der Optimierung)
  - $\Rightarrow$ Operatoren haben nur noch wenig Effekt
  - $\Rightarrow$ Crossover tauscht komplette aktive Module

Vermeidung von Introns

- mögliche Probleme: Stagnation der Optimierung, schlechte Resultate, größere Laufzeit
- Vermeidung destruktiver Operatoren, z.B.
  - Brood Recombination
  - intelligenter Crossover, der gezielt Crossoverpunkte auswählt
- Bestrafung von großen Individuen
- fortwährendes, leichtes Verändern der Bewertungsfunktion

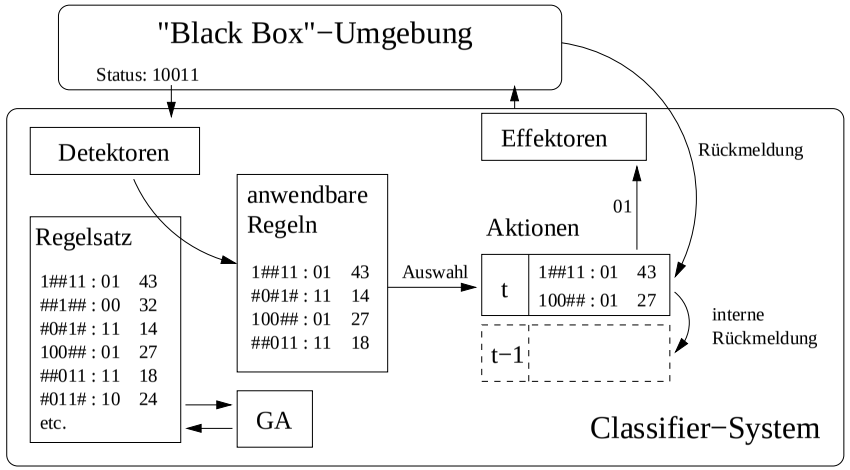
Andere Repräsentationen

- linear: z.B. Maschinencode
- Graphen
- Grammatikevolution: aus einem linearen String werden gemäß kontextfreien Regeln Syntaxbäume abgeleitet

Anwendungen von GP

- ▷ symbolic regression
- ▷ artificial ant: Santa Fe Trail
- ▷ Kontrolle von Industrieprozessen oder Robotern
- ▷ Bildverarbeitung
- ▷ Mustererkennung

Classifier–Systeme



Nachrichten und Regeln

- ▷ Nachrichten: $\nu \in \mathcal{N} = \{0,1\}^l$
- ▷ Bedingungen: $\beta \in \mathcal{B} = \{0,1,\#\}^l$ mit Platzhalter $\#$
- ▷ Regeln β/α mit $\beta \in \mathcal{B}$ und Aktion α
- ▷ Regel β/α heißt aktiviert falls

$$\bigwedge_{i=1}^{l} (\beta_i \neq * \;\Rightarrow\; \beta_i = \nu_i)$$

Auswahl der Aktion

- ▷ jede Regel besitzt eine Stärke
- ▷ Für die Auswahl: Wahrscheinlichkeiten für die möglichen Aktionen α der anwendbaren Regeln $\mathcal{A}$

$$\mathrm{Pr}[\alpha] \;=\; \frac{\sum_{\beta/\alpha \in \mathcal{A}} Stärke(\beta/\alpha)}{\sum_{\nu \in \mathcal{A}} Stärke(\nu)}$$

Modifikation der Stärke

- angewandte Regeln zur Zeit t: $Z(t)$
- Rückkopplung r vom System ($r = 0$ bei keiner Rückkopplung)
- Modifikation von $z \in Z(t)$:

$$St\ddot{a}rke(z) \leftarrow (1-\beta)St\ddot{a}rke(z) + \beta\frac{r}{|Z(t)|}$$

- Modifikation von $z \in Z(t-1)$:

$$St\ddot{a}rke(z) \leftarrow St\ddot{a}rke(z) + \beta\gamma\frac{\sum_{z' \in Z(t)} St\ddot{a}rke(z')}{|Z(t-1)|}$$

Generation neuer Regeln

- steady-state genetischer Algorithmus
- Erzeugung von zwei Kindern
- Auswahl proportional zu den Stärkewerten
- Crossover und Mutation
- Ersetzung von zwei zufälligen Regeln (proportional zum Kehrwert der Stärke)

Moderne Variationen

- zusätzliche Betrachtung einer Vorhersagegenauigkeit
- andere Darstellungen: z.B. Bäume aus GP
- Modifikationen des EA für breitere Streuung von Regeln
  z.B. nur gleichzeitig aktivierte Regeln werden rekombiniert

Michigan vs. Pittsburgh

- Michigan-CS: Population entspricht Regelsystem
  - Vorteil: Anpassung während der Regelung (sowohl Stärke als auch Regeln)
  - Nachteil: Konkurrenz zwischen Individuen und Kooperation als Regelwerk sind schwer vereinbar
- Pittsburgh-CS: Individuum entspricht Regelsystem
  - Vorteil: Regelsystem wird mehr als Einheit betrachtet
  - Nachteil: Anpassung während der Regelung ist kaum möglich

Anwendungen der CS

- Data-Mining

  $\Rightarrow$ können kompakt Zusammenhänge in Daten beschreiben

- Robotik

- Zeitreihenprognose