Computer codieren Computercode

Genetische Programmierung - verständlich erklärt

Genetische Programmierung ist ein Gebiet einer noch sehr jungen Wissenschaft im Bereich der Informatik. Anhand des Begriffes wird leicht Gentechnik, Biologie oder Bioinformatik assoziiert. In gewisser Weise ist GP mit diesen Begriffen verbunden, denn ein wesentlicher Bestandteil der genetischen Programmierung ist die künstliche Evolution. Ziel der genetischen Programmierung ist es, Algorithmen, Programme, oder allgemein gesagt, Funktionalitäten zu züchten! Was sich zu Beginn wie eine wenig sinnvolle Computerspielerei anhört, birgt ein enormes Potential für neue Lösungsansätze in vielen Fachgebieten. Wir wollen uns GP dennoch zu Beginn spielerisch nähern.

30.04.2002, Christoph Bültemann (cb@innoworx.de)

Das Spiel

Unser Spiel besteht darin, eine künstliche Welt aufzubauen, die den von uns definierten Regeln gehorcht. Wir setzen in diese eine Population primitiver Individuen aus, deren Wohl und Wehe davon abhängt, wie sie mit unseren Regeln zurechtkommen. Überlebt und gestorben wird nach Darwinschen Prinzipien - "survival of the fittest". Damit unsere Population nicht ausstirbt, können die ausgesetzten Individuen Nachwuchs bekommen. Besonderer Kindersegen ist jenen beschert, die sich gut an die Regeln anpassen konnten. Für die Artenvielfalt sorgt Mutation der Neugeburten.

Die Spielfiguren

Unsere Spiele-Welt ist keine Nachbildung der echten Welt - sie ist höchst primitiv. Unsere Individuen haben nur zwei Organe:

- das Input-Organ - Dieses ist ausschließlich in der Lage, Zahlen entgegenzunehmen und diese weiterzuverarbeiten.
- das Output-Organ - Nach Sammeln der Sinneseindrücke des Input-Organs reagiert das Output-Organ in Form der Ausgabe einer einzigen Zahl.

Der Zahlenwert des Output-Organs wird durch die verbleibenden Bestandteile des Individuums bestimmt - einer Kombination aus vordefinierten kleinsten Funktionseinheiten. Diese "Gene" führen beispielsweise arithmetische und logische Rechenoperationen aus oder treffen Ablaufentscheidungen wie Verzweigungen oder Wiederholungen. Sie bestimmen daher den Wert des Output-Organs. Bei der Geburt von neuen Individuen erhalten diese die Gene ihrer Erzeuger. Durch Mutation werden diese zusätzlich meist in geringem Maße zufällig verändert. So entstehen Individuen mit neuen Eigenschaften.

Die Spielregeln

Unsere Regeln werden von den Individuen nur durch ihre spärlichen Organe wahrgenommen. Die "Fitness" eines Individuums ist um so höher, je genauer es die Regeln einhält. Eine Regel besteht aus Zahlen für das Input-Organ und dem erwarteten Ergebnis des Output-Organs. Je näher das Ergebnis des Output-Organs eines Individuums dem durch die Regeln vorgegebenen Werten kommt, desto besser fällt die Bewertung für dieses Individuum aus. Um ein komplexes Verhalten unserer Individuen zu erzielen, reicht eine Regel bei weitem nicht aus. In diesem Fall muss ein Individuum einem großen Regelwerk genügen, den sogenannten Trainingsdaten - unzählige Datensätze, von denen jeder einzelne aus meist mehreren Zahlenwerten für die Input-Organe und dem erwarteten Ergebnis des Output-Organs besteht.

Ziel des Spiels

Die Population von Individuen ist selbständig - unsere Aufgabe ist es Regeln zu schaffen, die den Individuen begreiflich machen, wie sie sich zu verhalten haben. Meist bestehen diese Regeln aus Trainings-Datensätzen (Input-Wert(e) und gewünschter Output-Wert). Wenden wir diese an, bemerken wir, dass sich eine Population im Laufe einiger hundert, tausend oder zehntausend Generationen immer besser an die gegebenen Regeln anpasst.

In unserer künstlichen Welt findet Evolution statt!

Wir können uns nun entspannt zurücklehnen, einige Generationen von Individuen heranwachsen lassen und - sobald diese die erwünschte Fitness erreicht haben - das Individuum mit der größten Fitness heraussuchen. Dieses hat bei sorgsam gewählten Spielbedingungen nun die Fähigkeit, unsere beigebrachten Regeln zu generalisieren. Unsere Daten wurden nicht "auswendig gelernt", sondern das Individuum kann diese nun abstrahieren und auch für noch nie gelernte Input-Daten anwenden!

Von der Kunst, Problemlöser zu züchten

"Unlösbare" Aufgaben gibt es viele - in der Mustererkennung, Klassifizierung, Regelungstechnik, Prozesstechnik, für Vorhersagen jeglicher Art, im Bereich der künstlichen Intelligenz, für Data Mining, Expertensysteme, in der Mathematik, Physik, Chemie.... Sobald eine Aufgabe nicht ausreichend oder nicht mit einem vertretbaren Aufwand konventionell gelöst werden kann, ist das Züchten eines Individuums mit entsprechenden Fähigkeiten eine vielversprechende Alternative. Kann man Trainingsdaten für eine Aufgabe beschaffen oder bietet sich die Möglichkeit, die Evolution "onboard" - also in die Aufgabe integriert - ablaufen zu lassen, kann die Zucht beginnen. Grundsätzlich benötigen wir zur künstlichen Evolution eine Möglichkeit des "Feedbacks". Fällt dieses positiv aus, so erhöht sich die Fitness des überprüften Individuums.

Achtung, Faulpelze!

Das Wesen unserer Individuen ist ausschließlich davon geprägt, den vorgegebenen Regeln zu entsprechen - sie favorisieren dafür meist die einfachste Lösung. Schnell fällt diese nicht im Sinne des Regelerfinders aus. Dazu ein Beispiel: Mittels GP sollte ein einfacher Spracherkenner entstehen, welcher auf die Worte "ja" und "nein" reagiert - ein binärer Klassifikator. Jeweils 20 Audiosamples mit den Äußerungen "Ja" und "Nein" wurden als Trainingsdaten verwendet, der Vergleichswert wurde auf 1 für "Ja" und 0 für "Nein" bestimmt. Die Population lernte erstaunlich schnell, die Individuen hatten nur wenige Gene. Anfängliche Begeisterung wich schnell ernüchternder Erkenntnis: Die kleinen Faulpelze zählten die Anzahl der übergebenen Audiosamples. Da Aufnahme des Wortes "Nein" etwas länger ist, hatten sie damit Erfolg. Dieser systematische Fehler im Regelwerk wurde durch die Variation in der Sprechgeschwindigkeit und durch eine konstante Anzahl an Input-Parametern schnell behoben. Nach einem darauffolgenden etwas längeren Evolutionsprozess entstanden wiederum Individuen mit hoher Fitness. Eine Gegenprobe mit neu erstellten "Ja"- und "Nein"-Aufnahmen brachte erneute Ernüchterung: Die Ergebnisse des besten Individuums waren reiner Zufall. Nach Analyse der Gene stellte sich heraus, dass dieses Individuum durch "geschickte" Verknüpfungen einzelner Input-Werte zum Ergebnis kam - es hatte "auswendig" gelernt.

Erziehungsmaßnahmen

Um diesen unangenehmen Charaktereigenschaften entgegenzuwirken bieten sich einige Maßnahmen:

- eine große Anzahl an Trainingsdaten
- Verwendung einer zufällig ausgewählten, für jede Generation unterschiedlichen Untermenge der Trainingsdaten
- Erzeugen zusätzlicher künstlicher Trainingsdaten, beispielsweise durch Interpolation von realen Daten
- geringfügige, zufällige Veränderung der Trainingsdaten

Es ist leicht ersichtlich, dass die zwei letztgenannten Maßnahmen nicht für alle Aufgaben einsetzbar sind. Die Morphologie der künstlichen Daten sollte möglichst gut der der realen Daten entsprechen. Im genannten Beispiel wurden die Trainingsdaten durch einen nachgeschalteten "Rauschgenerator" verändert, was der Natur eines Audiosignals wohl am besten entspricht. So erblickte nach einigen zig-tausend Generationen ein Mini-Spracherkenner das Licht unserer simulierten Welt. Dieser konnte sehr zur Freude des Autors sein "Herrchen" recht gut verstehen.

Ein derart gut erzogenes Individuum kann sehr positive Eigenschaften aufweisen:

- Robustheit gegenüber Fehlern in den Input-Daten
- Sehr gute Generalisierung eines Problems
- Effizient durch wenige verwendete Funktionseinheiten (geringer Speicherbedarf, hohe Ausführungsgeschwindigkeit)

Des Pudels Kern

Ein durch Evolution in einer künstlichen Welt erzeugtes Individuum kann auf unterschiedlichste Art aufgebaut sein. Hier wird eine dieser Arten betrachtet, die des GP-Systems des Autors. Die "Gene" eines Individuums werden hier direkt durch ein ausführbares Kommando repräsentiert. Verfügbare Kommandos sind logische und arithmetische Grundrechenarten, Sprünge und Bedingungen. Es steht dem Anwender außerdem frei, eigene Funktionalitäten in Form von Kommandos hinzuzufügen. Ein Individuum besteht daher aus einer Sequenz solcher Kommandos, welche wie ein Programm sequentiell ausgeführt werden. Für Zwischenergebnisse steht ein "Gedächtnis" in Form von Registern zur Verfügung, welches eine gewisse Anzahl von Werten speichern kann. Das GP-System errechnet die Fitness der Individuen durch das Ausführen deren Kommandos. Erreicht eines dieser Programm-Individuen die gewünschte Fitness, wird es automatisch in Programmcode der Programmiersprache "C" konvertiert und kann sofort in Softwareprojekte eingebunden werden. Andernfalls erzeugt das GP-System die nächste Generation von Individuen auf Basis der erwähnten evolutionären Prinzipien.

```
GET          2     0     0              0 255
JUMP_N       2    10     5     1492523112   1
SET         10     4     4      -31079858   1
SET          2     4    15       55147604   1
NAND        14     1     7      584919344   1
MUL          0     6    10    -1069338880   1
OR           5     3     2    -1247620864   1
MUL          8     9    10     2121433944   1
SET         11     9     4    -1276720336   1
SKIP_NZ     15    13     0     1076383462   1

...
```

Repräsentation eines Individuums im GP-System...

```
long gp00(long nItems, long *data)
{
      register long     regCond = 0;
      register long     regLoop = nItems * 2.000000;
      register long     currentItem=0;
      long              reg[16];

      memset(reg, 0, 16*sizeof(long));

l000: regCond=reg[1]=0;
l001: if(currentItem<nItems) reg[2]=data[currentItem++]; else goto
      byebye;
l002: if(regCond>0 && (--regLoop>0)) goto l000;
l003: if(regCond<0) goto l005;
l004: regCond=reg[0]= reg[14]? reg[10]%reg[14] : 0;
l005: regCond=reg[13]=1094079744;
l006: if(regCond) goto l008;
      ...
```

...und automatisch generierter C-Quellcode

Geduld, bitte!

Abhängig von der Problemstellung, der Populationsgröße und der Trainingsdaten ist ein enormer Rechenaufwand notwendig, um die Fitness der Individuen einer Generation zu bestimmen. Neben einer hochgradig optimierten Fitness-Berechnung der Individuen ist das GP-System des Autors für den Parallelbetrieb auf beliebig vielen Rechnern ausgelegt. Ausgehend von dem Mangel an eigens für die GP-Evolution bereitstehenden Rechnern, bietet es eine sogenannte "Büro-Cluster"-Lösung, welche nur überschüssige Rechenleistung verwendet. Ein Benutzer eines derartigen Rechners kann diesen ohne Verlust an Rechengeschwindigkeit weiter für eigene Anwendungen nutzen. Auch entfernte Rechner können, eine gelegentliche Verbindung zum Internet vorausgesetzt, überschüssige Rechenleistung beisteuern. So ergeben sich beste Voraussetzungen für schnelle Ergebnisse. Dennoch müssen nicht selten Problemstellungen vor Beginn der Evolvierung einer Lösung stark partitioniert werden, um in einer zumutbaren Zeit Ergebnisse zu erzielen. Beispielsweise entwickelte sich der bereits beschriebene Mini-Spracherkenner innerhalb eines Tages auf zwei Rechnern.

Universalgenies züchten...

...können evolutionäre Systeme sicher nicht! Dennoch lässt sich mit ihnen Erstaunliches erreichen. Beispielsweise wurden für ein biometrisches Personenverifizierungssystem genetische Programme zur letztlichen Entscheidung herangezogen. Aus 11 Indikatoren sollte eine eindeutige Entscheidung gewonnen werden. Im Vergleich zu einer konventionell entwickelten Lösung konnte ein Programm gezüchtet werden, welches die Anzahl fehlerhaft zurückgewiesener Personen halbierte.

Aus unzähligen weiteren Einsatzmöglichkeiten dieser Methode seien einige kuriose genannt:

- Aufrechter Gang von Robotern
- Verfeinerung des Geschmacks von Kaffee
- Computerschach
- Ausbildung von Fortbewegungsorganen einfachster, effektiver Laufmaschinen
- aerodynamische Optimierungen
- Design elektronischer Schaltungen

Alltägliche Probleme werden wohl auch in Zukunft konventionell gelöst werden. Doch findet sich keine ausreichende Lösung, können evolutionäre Ansätze wie die genetische Programmierung diese auf effektive Weise hervorbringen■