

Flash fill en Equation Discovery*

Craps Jeroen

Bachelor Informatica

KU Leuven, België

jeroen.craps@student.kuleuven.be

De Groote Tom

Bachelor Informatica

KU Leuven, België

tom.degroote@student.kuleuven.be

Abstract

Deze paper beschrijft een applicatie voor het vinden van een vergelijking in een set van relatief willekeurige getallen die door een gebruiker zijn gespecificeerd. Het doel van deze applicatie is om gebruikers te helpen in het vinden van een passende vergelijking voor de overige gegevens aan de hand van enkele voorbeelden. De applicatie voorziet dus de gebruiker om meerder voorbeelden te geven met welke de applicatie rekening moet houden. Op basis van enkele experimenten bleek deze applicatie in een beperkte tijd toch satisfieerbare resultaten te voorzien aan de gebruiker.

1 Introductie

Steeds vaker wordt er van een computer verwacht dat deze complexere taken kan uitvoeren. Terwijl het voor de gebruikers meestal duidelijk is wat er gewenst wordt, is dit voor de computer niet altijd het geval. Een recent voorbeeld hiervan is de 'FlashFill' functionaliteit dit momenteel aanwezig is in Excell 2013. [?] Hierbij is het de bedoeling dat de computer het door de gebruiker gewenste patroon op een set van data herkent en kan toepassen op gelijkaardige sets van data om de gebruiker tijd te besparen. Een ontbrekende functionaliteit hierin is het herkennen van wiskundige bewerkingen of functies. Met andere woorden, het 'bepalen' van een passende vergelijking voor de gebruikers data. Verbeteren van het gebruikersgemak bij het bepalen van geschikte wiskundige vergelijkingen is het doel van dit werk.

Het doel van dit werk is om gelijkaardige functionaliteit aan die van Flash Fill te onderzoeken op het vlak van wiskundige vergelijkingen. Het onderzoek heeft zijn plaats in het domein van 'Artificiële Intelligentie', meer bepaald 'Machine Learning By Example'. Slechts een zeer beperkte groep van gebruikers hebben dit probleem. Vandaar dat er nog weinig tot geen onderzoek gedaan is met betrekking tot dit onderwerp. Bijgevolg is er geen dataset die kan gebruikt worden als invoer voor de experimenten van het onderzoek.

Toch is het bepalen van dit soort van vergelijking geen volledig onbekend onderzoeksdomein. In 2002 verscheen

er een paper die een oplossing voor het 'Countdown Problem' gaf. Bij het 'Countdown' is een Brits televisie programma waarbij de deelnemers een wiskunde vergelijking zoeken met de gegeven nummers om zo dicht mogelijk tot bij een bepaald streefdoel te komen. Een belangrijke beperking op dit probleem zijn wel dat elk gegeven cijfer maximaal éénmalig mag voorkomen. Indien een gebruiker bepaalde waarde meermaals wilt gebruiken is dit niet meer volgens de regels van 'Countdown'. Aangezien de gebruiker sommige waarden meermaals wil kunnen gebruiken moet er een alternatieve manier van aanpak gevolgd worden.

2 Probleemstelling

Gebruikers hebben reeds de mogelijkheid om patroonherkenning te gebruiken op hun dataset. Toch kan de gebruiker op zoek zijn naar een wiskundige relatie tussen in zijn dataset in plaats van een patroon.

2.1 Context-vrije grammatica

Om deze wiskundige relaties op te bouwen wordt er gebruikt gemaakt van een contextvrije grammatica. Volgens de 'formele taal theorie' [] is een contextvrije grammatica een formele grammatica waarbij elke productieregel er uit ziet als volgt: $V \rightarrow w$. Hier staat V voor één niet-terminaal symbool is en w een tekenreeks is van niet-terminale en terminale symbolen. 'Contextvrij' betekend dat de productieregels kunnen toegepast worden los van de context waarin het niet-terminale symbool zich bevindt.

$$\begin{array}{l} E \rightarrow E + E \\ E \rightarrow E - E \\ E \rightarrow E \times E \\ E \rightarrow E \div E \\ E \rightarrow 1, 2, \dots, 9, a, b, \dots \end{array}$$

Als een andere wiskundige operatie gewenst is moet deze toegevoegd worden als een productieregel aan bovenstaande grammatica. De gemakkelijke aanpasbaarheid is een voordeel van het werken met een context-vrije grammatica. Omdat de productieregels volledige onafhankelijk zijn van elkaar is het eenvoudig om de gewenste operaties te definiëren.

*Voor verandering vatbaar.

2.2 Boomstructuur

Door herhaaldelijke toepassing van de productieregels ontstaat er een boomstructuur. In deze boom bevinden zich alle mogelijke samenstellingen van productieregels die door de grammatica bepaald worden. Deze operatie zorgt voor een exponentiële groei r^d waarbij d staat voor de diepte van de boom en r voor het aantal mogelijke productieregels van de vorm $E \rightarrow E \text{ operand } E$. De groei van deze boom staat beschreven in volgende figuur en tabel.

Aangezien de groei van de boom exponentieel is bekommt men veel grotere berekeningstijden volgens toename van de diepte in de boom. De grote overhead die het opstellen van de boom is, heeft tot gevolg dat het ten zeerste aangeraden om deze boom slechts éénmalig op voorhand uit te werken.

2.3 Evaluatiecriteria

'Snelheid, correctheid, gebruikersgemak en uitbreidbaarheid': dit zijn de vier evaluatie criteria waarop dit onderzoek gaat beoordeeld worden. Het doel van dit onderzoek is dus een efficiënt, correct en uitbreidbaarheid algoritme te vinden. Ook het gebruikersgemak is van belang bij het definiëren van wat de gebruiker juist van het algoritme verwacht.

3 Aanpak

3.1 Brute Force

Dit is de vertrek basis, we willen dit enorm verbeteren door het toepassen van optimalisaties

3.2 Toevoegen van gewichten

Verhoogt de zoekruimte \rightarrow vertraagt maar hogere oplossingsgraad

3.3 Prunen op basis van vergelijkingen

Verkleint zoekruimte zonder de oplossingsgraad te verlagen

3.4 Prunen op basis van vergelijkingen en gewichten

Verkleint de zoekruimte enorm maar sommige oplossingen kunnen niet gevonden worden, toch de beste keuze zal blijven uit experimenten

4 Dataset

we beginnen met het vermelden dat er geen gebruikersdata is

4.1 Verschillende randomgeneratoren

We vermelden welke mogelijke randomgeneratoren we hebben

4.2 Verantwoording keuze

We vergelijken de randomgeneratoren met de brute force en kiezen de generator die het dichtst bij gebruikersdata zou liggen. We zullen alle experimenten met deze generator uitvoeren

5 Experimenten

Alle experimenten worden enerzijds getest op oplossingsgraad als de volledige boom wordt doorlopen en anderzijds binnen een tijdsmarge van 2s. Aanhaken dat we naarmate de experimenten vorderen we ons baseren op voorgaande experimenten om keuzes te maken, anders zou het runnen van de experimenten enerzijds nutteloos zijn en anderzijds gigantisch veel tijd vragen (denk bv maar aan 10 gewichten enzo)

Vermelden welke processor enzo

5.1 Boom op voorhand opstellen

Enkel tijdswinst aanhalen hier

5.2 Prunen op vergelijkingen

Boom wordt niet op voorhand opgesteld aangezien we dan meer kunnen prunen, met name K1-K1 enzo

5.3 Gewichten

Uitgevoerd op basis van een geprunte boom (experiment hierboven)

5.4 Prunen op vergelijkingen en gewichten

Gebaseerd op prime gewichten, boom wordt niet meer op voorhand berekend

6 Resultaten

Enkel bespreken van experimenten op basis van tijd en prime gewichten. Vergelijken van de brute force aanpak, de op voorhand berekende boom aanpak (met beperkte pruning mogelijkheden) en de uitiem geprunte boom

7 Andere aanpakken

Hier halen we paden aan die ons doorheen het jaar zijn tegengekomen. In onze ogen hebben ze geen meerwaarde en hier verantwoorden we waarom

7.1 Heuristiek

Resultaten kunnen enorm verschillen door $\times \div$ en \wedge .

7.2 Constraint programming

Onze kennis is niet goed genoeg om hier degelijke resultaten in te genereren

7.3 Op basis van onbekende

1 vb, je kan met 1 onbekende werken, enz. Niet nuttig aangezien dan altijd een oplossing gevonden wordt. Enerzijds veroorzaakt dit een enorme overhead waardoor de zoektijd enorm toeneemt anderzijds zullen de gevonden vergelijkingen gebaseerd zijn op constantes eerder dan op kolomwaarden

8 Toekomstig werk

8.1 Toevoegen haakjes

8.2 Berkenen van 1 losstaand gewicht

9 Conclusie

Hier komt onze conclusie

10 Code

Hier moeten we vermelden dat er een GUI bestaat en dat de code via git beschikbaar is. Alles geschreven met java

Acknowledgments

Referenties

References

[Hutton, 2002] Graham Hutton. Functional pearl: The countdown problem. *Journal of Functional Programming*, 6(12):609–616, November 2002.