Poročilo za 4. Domačo nalogo pri predmetu NSU

Matija Kerkoč (27192017)

25. junij, 2021

1 Opis gramatik

Ker imamo štiri enačbe bomo najprej generirali splošno gramatiko, nato pa jo bomo prilagodili za specifičen problem. Ta ideja je smiselna, saj v primeru, ko delamo s preveč splošnimi gramatikami ne dobimo iskanih enačb. V primeru, ko za vsako enačbo spišemo svojo funkcijo za generiranje gramatike, pa imamo preveč kode, ki se ponavlja. Splošna gramatika, ki jo uporabljamo ima torej tri možne člene katerim dodamo pravilne enote: `E_` ("equation"), `F_` ("formula") in `T_` ("term"). Uporabljali bomo osnovne računske operacije $+, -, *, /$, kjer upoštevamo pravila za računanje z enotami. Na koncu dodamo še posebne funkcije, ki jih lahko uporabljamo. V našem primeru smo se odločili za funkcije $\sin, \cos, \sqrt{\ }$. Glede na ujemanje enot lahko sedaj generiramo splošna prepisovalna pravila za gramatiko.

Opomba. Goljufanje s prirejanjem gramatik načeloma ni zaželjeno, je pa vsekakor možno posredno goljufanje preko verjetnosti, ki jih predpišemo v gramatiki (te so namreč odvisne zgolj od naše odločitve). Ne glede na to kako postavimo verjetnosti pa se izkaže, da je smiselno generirati preprostejše enačbe (torej dati večjo verjetnost zaključku enačbe ter manjšo verjetnost rekurziji). V nasprotnem primeru zelo hitro naletimo na pregloboko rekurzijo oziroma "recursion error" v Pythonu.

1.1 Energija prožnostne vzmeti $W = \frac{1}{2}kx^2$

Pri prvi enačbi obstaja možnost, da bi jo lahko odkrili celo brez vpeljave enot. Ta ideja se izkaže za pravilno, mi pa vseeno vpeljimo gramatiko z enotami.

Če pogledamo najprej splošno gramatiko opazimo, da imamo tri prepisovalna pravila oblike

```
F_m2s-2kg1 -> F_m2s-2kg1 '/' T_m0s0kg0 [0.2],
F_m0s-2kg1 -> F_m0s-2kg1 '/' T_m0s0kg0 [0.2] in
F_m1s0kg0 -> F_m1s0kg0 '/' T_m0s0kg0 [0.2].
```

Torej bomo vedno neko količino delili s količino, ki ima enoto [1]. Torej so ta pravila nepotrebna (lahko jih skrijemo v pravila za množenje s konstanto) in jih lahko odstranimo. Podobno lahko odstranimo tudi pravila za deljenje, ko delimo dve količini z enakimi enotami ter dobimo konstanto. Ta pravila so odveč saj lahko v takem primeru preprosto pokrajšamo spremenljivke..

1.2 Prepotovana pot pri enakomernem pospešenem gibanju $s = \frac{1}{2}at^2 + v_0t$

V tem primeru splošno gramatiko generiramo na podoben način kot prej, le da gramatike tukaj ne moremo kaj bolj omejiti. V splošnem lahko iz vhodnih enot dobimo poljubno kombinacijo enot sestavljeno iz [m], [s] in [kg]. Torej lahko v tem primeru uporabljamo tudi funkcije, ki primejo na količini z enoto [1]. Gramatika je precej splošna zato pričakujemo, da bomo morali povečati število modelov.

V tem primeru se včasih že pojavi primer, ko nam Python generira pregloboko rekurzijo. Zato moramo postaviti precej veliko verjetnost ($p = 0,9$) za končanje.

1.3 Kosinusni izrek $c = \sqrt{a^2 + b^2 - 2ab\cos\gamma}$

V tem primeru imamo le eno enoto [m], ki jo lahko dobimo na različne načine s potenciranjem. Tudi tukaj lahko delamo s 3-dimenzionalnimi vektorji enot, le drugi dve komponenti bosta vedno enaki 0. Kljub na prvi pogled lahki nalogi, se izkaže da je le ena enota pravzaprav težava za generiranje enačb, saj imamo na voljo ogromno različnih možnosti kako pravilno skombiniramo enote. Podana gramatika kljub 1000 generiranim enačbam namreč ni uspela najti prave.

Programu lahko v tem primeru pomagamo ter gramatiko priredimo. Izkaže se, da moramo gramatiko krepko oklestiti (odstraniti deljenje, povečati verjetnost za korenjenje, odstraniti funkcijo sinus, ...), da program najde pravilno enačbo.

1.4 Sinusno nihanje $x = x_0 \sin(\omega t + \alpha_0)$

Gramatika je zopet podobna. Ker se sama enačba imenuje sinusno nihanje, je logično predpostaviti, da bo v enačbi nastopal sinus, ne pa tudi koren. Vseeno pa je smiselno ohraniti funkcijo kosinus, saj sta ti funkciji tesno povezani.

2 Izračun verjetnosti

Označimo s p_{G_i,e_i}, $i \in \{1,2,3,4\}$ verjetnost, da z gramatiko G_i izpeljemo iskano enačbo e_i. V našem primeru imamo na voljo štiri različne načine na katere lahko pristopimo k problemu

1. natančen razpis drevesa ter izračun verjetnosti vseh možnih poti po katerih lahko generiramo iskano enačbo,
2. generiranje "veliko" primerov enačb ter preverjanje kolikokrat se pojavi pravilna enačba,
3. izračun najbolj verjetne (nekaj najbolj verjetnih) poti ter dodaten izračun, na koliko različnih načinov se lahko inačica take poti pojavi,
4. izračun verjetnosti le nekaj najbolj verjetnih poti.

Načini so urejeni po padajoči natančnosti s katero ocenimo verjetnost.

Verjetnost uspeha lahko izračunamo kot v [2]. Verjetnost da bomo našli pravilno enačbo z gramatiko G_i za enačbo e_i, kjer je število generiranih modelov enako N, je enaka

$$p_{G_i,e_i}(N) = 1 - (1 - p_{G_i,e_i})^N.$$

Zgornja enačba ni nič posebnega, izpeljemo jo lahko tudi sami. Glavni izziv je poiskati verjetnost p_{G_i,e_i}.

Za oceno verjetnosti p_{G_i,e_i} navzdol imamo na voljo načina številka 3 in 4. Oba primera sta podobna, le da pri načinu 3 malo bolj kompliciramo. Najnatančnejši način je prvi. Vendar pa že v naprej vemo, da bo ta način najtežji in za zapletenejše gramatiko skoraj nemogoč.

Za konec opišimo še način številka 2. Generiramo veliko število enačb ($N = 100000$) in nato preiščemo generirane enačbe kolikokrat se je v njih pojavila pravilna enačba. Seveda moramo biti pri tem pazljivi, saj se lahko enačba pojavi v več različnih oblikah. Za potrebe naše domače naloge bomo malo verjetne primere kar izpustili. To so tisti primeri, pri katerih dobimo iskano enačbo s kombinacijo več operacij. Na primer spremenljivko s z enoto [m] lahko dobimo na več različnih načinov. Od teh so smiselni na primer: $v_0 t$ ali at^2, za nas nesmiselni pa recimo: a^2t^3/v_0 Seveda so taki primeri smiselni, če gledamo le njihove enote, vendar je njihova verjetnost tako majhna, da jih lahko izpustimo. Seveda pa moramo paziti, da upoštevamo vse možne oblike enačb. V našem primeru to pomeni, da sta za enačbo $s = \frac{1}{2}at^2 + v_0 t$ dobra kandidata tako $s = C_0 at^2 + v_0 t$, kot $s = C_0 at^2 + C_1 v_0 t$.

Opomba. V splošnem je verjetnost najdbe enačbe nemogoče izračunati. Zakaj? V primeru neke splošne (ne niti nujno zapletene) gramatike se lahko vedno "zaciklamo" ter generiramo neko poljubno enačbo, ki še vedno ustreza prepisovalnim pravilom. Verjetnost take enačbe je seveda majhna, vendar to pomeni, da v primerih b), c) in d) ne moremo izračunati natančne verjetnosti, da bomo našli iskano enačbo. Celo natančen izpis drevesa prepisovalnih pravil ter mukotrpno računanje ne bo dalo željenega rezultata. Po tem razmisleku je pravzaprav najnatančneši način za oceno način številka 2.

2.1 Energija prožnostne vzmeti

Kot lahko vidimo iz oblike enačb, ki jih zgeneriramo oziroma kot bo sledilo iz razmisleka nižje je v tem primeru verjetnost, da bomo našli enačbo enaka 1. Z drugimi besedami, prepisovalna pravila (oz. enote količin v prepisovalnih pravilih) nas bodo vedno silila v konstrukcijo enačbe, ki ima ciljno spremenljivko z enotami [m^2kg/s^2]. Ker so vektorji vhodnih spremenljivk neodvisni je možna enačba ena sama.

Dokažimo sedaj zgornji premislek. Naj bo (x, y, z) vektor enot. V našem primeru prva komponenta predstavlja enoto meter [m], druga komponenta enoto sekunda [s], tretja pa enoto kilogram [kg]. Ker ostale enote ne nastopajo, lahko preostale komponente izpustimo. Torej ima energija W vektor enot $(2, -2, 1)$, koeficient vzmeti k vektor enot $(0, -2, 1)$ in raztezek x vektor enot $(1, 0, 0)$.

Za potrebe naše domače naloge bo dovolj, da računamo z vektorji dimenzije 3, analogni rezultati pa veljajo tudi v splošnem – sedemdimenzionalnem – primeru.

Kot že omenjeno v navodilih za funkcijo $x \mapsto f(x)$ imamo spremembe opisane kot

- množenje: $(\alpha_1, \beta_1, \gamma_1) \cdot (\alpha_2, \beta_2, \gamma_2) \mapsto (\alpha_1 + \alpha_2, \beta_1 + \beta_2, \gamma_1 + \gamma_2)$,
- deljenje: $(\alpha_1, \beta_1, \gamma_1)/(\alpha_2, \beta_2, \gamma_2) \mapsto (\alpha_1 - \alpha_2, \beta_1 - \beta_2, \gamma_1 - \gamma_2)$,
- potenciranje: $(\alpha, \beta, \gamma)^n \mapsto (n\alpha, n\beta, n\gamma)$,
- korenjenje: $\sqrt[n]{(\alpha, \beta, \gamma)} \mapsto (\frac{1}{n}\alpha, \frac{1}{n}\beta, \frac{1}{n}\gamma)$,
- sinus/kosinus: funkciji primeta le na brezdimenzijski spremenljivki in ne spremenita enot.

Preverimo lahko, da je prostor vseh vektorjev enot $\mathcal{E}$ z operacijama množenjem in potenciranjem, vektorski prostor. Definirajmo kako se operacije med spremenljivkami izražajo v prostoru vektrojev enot.

Naj bosta e_1 in e_2 dve enačbi z enotama $(\alpha_1, \beta_1\gamma_1)$ in $(\alpha_2, \beta_2, \gamma_2)$. Množenje ter deljenje spremenljivk generirata natanko seštevanje oziroma odštevanje v prostoru vektorjev enot, potenciranje in korenjenje spremenljivk pa natanko množenje s skalarjem v prostoru vektorjev enot. Torej se operacije ujemajo s tistimi, ki bi jih imeli v $\mathbb{R}^3$. Od sedaj naprej lahko gledamo na prostor $\mathcal{E}$ kot na vektroski prostor.

Vektorja enot za vhodne spremenljivke sta linearno neodvisna saj imata 0 na različnih komponentah vektorjev.

Po znanem izreku iz linearne algebre lahko vektor $(2, -2, 1)$ na enoličen način zapišemo kot linearno kombinacijo linearno neodvisnih vektorjev. V našem primeru je to

$$(2, -2, 1) = (0, -2, 1) + 2(1, 0, 0),$$

kar nam določa enoličen zapis enačbe

$$W = Ckx^2,$$

kjer je C brezdimenzijska konstanta, ki jo moramo še določiti.

Opomba. Iz zgornjega primera je lepo razvidno kdaj rešitev ne bo enolična. To se bo zgodilo v primeru, ko vektorji enot za vhodne spremenljivke niso neodvisni ali pa, ko se nam bodo enote vhodnih spremenljivk pokrajšale (pri deljenju ali korenjenju).

Opomba. Vektorja $(1, 0, 0)$ in $(0, -2, 1)$ ne razpenjata polne baze prostora $\mathbb{R}^3$, zato poljubnega vektorja iz tega prostora ne moremo zapisati kot njuno linearno kombinacijo. Ker pa smo v gramatiko vpeljali enote to niti ni potrebno. Dovolj je, da znamo konstruirati vektor $(2, -2, 1)$.

2.2 Prepotovana pot pri enakomernem pospešenem gibanju

Iskana enačba je $\frac{1}{2}at^2 + v_0t$. Najprej lahko verjetnost odkritja enačbe ocenimo preko postopka iz točke 2 v uvodu poglavja kot

$$\widehat{p_2} = 3/100000,$$

kar nam podaja približno vrednost za iskano verjetnost. Torej je verjetnost uspeha enaka

$$\widehat{p_{G_2,e_2}}(100) = 1 - (1 - \widehat{p_2})^{100} = 0,950215172044.$$

Drug način, na katerega lahko dobimo spodnjo mejo je izračun enega sprehoda skozi drevo in dejstva, da lahko iskano enačbo dobimo na kvečjemu več načinov. Torej bo v našem primeru verjetnost, da dobimo enačbo, ko se enkrat spustimo skozi drevo

$$\widetilde{p_2} = 0,3 \cdot 0,4 \cdot 0,2 \cdot (0,6 \cdot 0,9 \cdot 0,9) \cdot 0,4 \cdot (0,9 \cdot 0,9 \cdot 0,9 \cdot 0,7 \cdot 0,6 \cdot 0,3) = 0,002142770111.$$

Ko to vstavimo v enačbo dobimo

$$\widetilde{p_{G_2,e_2}}(100) = 1 - (1 - \widetilde{p_2})^{100} = 0,1928367245533,$$

torej imamo precej dobre možnosti, da enačbo odkrijemo.

Če želimo verjetnost izračunati še natančneje, postopamo takole. V našem primeru naslednje operacije

- množenje konstante z količino z enotami [m],
- kvadriranje količine z enoto [s],
- množenje količin z enotami [m/s^2] in [s^2],
- množenje količin z enotami [m/s] in [s],
- seštevanje količin z enotami [m].

Mislimo si, da so bile vse enačbe generirane z upoštevanjem komutativnosti $\cdot$ in $+$. Torej bo pravilna enačba prišla iz $3!2!2 = 24$ poti skozi drevo. Iskana verjetnost je torej

$$\widetilde{p_{G_2,e_2}}(100) = 1 - (1 - 24\widetilde{p_2})^{100} = 0,9948696936401,$$

kar se sklada s poganjanjem programa, saj z našo gramatiko enačbo odkrijemo skoraj vedno (v resnici je dovolj že 50 poskusov za odkritje enačbe (verjetnost 92%)).

Opomba. Ta verjetnost še vedno ni čisto natančna saj se na poti skozi drevo lahko zaciklamo v poljubno dolge enačbe, ki tudi generirajo pravilno enačbo. Ker pa so te verjetnosti res izredno majhne se zadovoljimo z dobljeno verjetnostjo kot z "zelo dobrim" približkom.

2.3 Kosinusni izrek

V primeru kosinusnega izreka je računanje vseh možnih enačb skozi prepisovalna pravila ideja obsojena na propad. Zato verjetnost raje ocenimo po načinu številka 2. Dobimo

$$\widehat{p_3} = \frac{1}{100000}$$

in verjetnost, da najdemo pravo enačbo v 100000 poskusih

$$\widehat{p_{G_3,e_3}}(100000) = 1 - (1 - \widehat{p_3})^{100000} = 0,6339676587267.$$

V primeru, da generiramo le 100 ali 1000 modelov bo enačbo skoraj nemogoče odkriti.

2.4 Sinusno nihanje

V zadnjem primeru dobimo

$$\widetilde{p_3} = 0.6 \cdot 0.4 \cdot 0.6 \cdot 0.2 \cdot 0.5 \cdot 0.2 \cdot 2 \cdot 0.7 \cdot 0.5 \cdot 0.6 \cdot 0.5 \cdot 0.5 = 0,0003024.$$

V našem primeru imamo $2!2!2!$ možnosti za orientacijo enačbe, torej dobimo

$$\widetilde{p_{G_4,e_4}}(100) = 1 - (1 - 8\widetilde{p_4})^{100} = 0,21215111099497,$$

v primeru 100 generiranih modelov, kar je približno 21% in imamo realne možnosti, da bomo enačbo našli.

Poskusimo še nekoliko večji $N = 1000$. V tem primeru dobimo

$$\widetilde{p_{G_4,e_4}}(1000) = 1 - (1 - 8\widetilde{p_4})^{1000} = 0,9112676720708,$$

kar nam skoraj že zagotavlja, da bomo enačbo našli.

Za konec si poglejmo še oceno verjetnosti po načinu številka 2. Izračunamo $\widehat{p_4} = 2/100000$ in dobimo

$$\widehat{p_{G_4,e_4}}(1000) = 1 - (1 - \widehat{p_4})^{1000} = 0,8673804441052.$$

Izgleda, da smo tukaj verjetno izpustili kakšen primer, ko smo primerjali dobljene enačbe. Verjetnost je višja in in verjetno bolj v območju nad 90%, saj že z zgeneriranimi 1000 primeri enačbo tudi konkretno generiramo.

3 Dobljene enačbe

3.1 Energija prožnostne vzmeti

Dobljena enačba je enaka

```
model: 0.5*k*x**2,
```

z verjetnostjo $1.3207 * 10^{-21}$ in napako $2.8044 * 10^{-33}$. Napaka je majhna, torej lahko z veliko verjetnostjo trdimo, da smo dobili pravilno enačbo.

Kot že omenjeno verjetnost, ki jo vrne `ProGED` ni pravilna. Je le verjetnost, da smo dobili tale specifičen model.

Pripomniti velja še, da z definirano gramatiko dobimo več "različnih" kandidatov. Še ena možnost je na primer `C0*k*x**2 + k*x**2`, ki pa je v resnici le različica iskane enačbe. V končnem ocenjevanju modelov namreč dobilmo le enačbe oblike `C*k*x**2` (lahko si mislimo, da smo člen `k*x**2` izpostavili ter nato konstanto `C0+1` skrili v novo konstanto `C`).

3.2 Prepotovana pot pri enakomernem pospešenem gibanju

Dobljena enačba je enaka

```
model: 0.5*a*t**2 + t*v0,
```

in napaka $4,0104 \cdot 10^{-33}$. Napaka je tudi tukaj majhna, torej lahko z veliko verjetnostjo trdimo, da smo dobili pravilno enačbo

3.3 Kosinusni izrek

V tem primeru z gramatiko G_3 sinusnega izreka ne dobimo. Tudi generiranje modelov ter predvsem učenje traja ogromno časa. Skoraj desetkrat toliko kot pri ostalih enačbah, zato tudi povečevanje števila poskusov (v našem primeru $N = 100$) ni smiselno. Na nek način je ta rezultat pričakovan, saj imamo lahko ogromno število kombinacij spremenljivk, ki bodo dale pravilne enote.

Najboljši model, ki ga dobimo je

```
model: 2,53934381141167*sqrt(0,155080427214726*a**2*g - a*b*g*cos(g)**2*cos(cos(g + cos(cos(g)
   - 17,8027437314645))) + 0,155080427214726*b**2*g**2 + 0,155080427214726*b**2*g*cos(g)
              + 0,0280331800251519*(1,53893503865512*a**2 + b**2)*cos(g)),
```

z napako $1,1818 \cdot 10^{-2}$. Kot zanimivost naj omenim, da bi enačbo z verjetnostjo 86% verjetnostjo odkrili v primeru $N = 100000$, vendar sem svojemu računalniku želel prihraniti nekaj ur mletja. Že v primeru $N = 100$ smo za generiranje in ocenjevanje modelov porabili okoli 15 minut.

Opomba. V kolikor bi želeli modelu nekoliko pomagati lahko gramatiko nekoliko omejimo. Ideja je odstraniti deljenje iz pravil in tako prepovedati programu, da bi generiral dolge enačbe. Vendar pa tudi v tem primeru verjetnosti ne izboljšamo kaj veliko.

3.4 Sinusno nihanje

Dobimo enačbo

```
model: x0*sin(alpha0 + omega*t),
```

z napako $2,3618 \cdot 10^{-33}$.

Zaključek

Izkaže se, da je vpeljava enačb smiselna ideja pri generiranju enačb. Na ta način program ne generira enotsko nesmiselnih kombinacij spremenljivk. Izkaže se, da ta način najbolje deluje, ko imamo na desni strani enačbe različne vektorje enot (idealno celo neodvisne), saj bo v tem primeru program zelo omejen in nam bo vrnil le malo enačb, ki bodo z veliko verjetnostjo pravilne. Nekoliko slabše vpeljava enot deluje v primeru, ko imamo na desni strani zelo podobne enote. V tem primeru lahko s kombinacijami množenj in deljenj (lahko tudi korenjenja) dobimo veliko smiselnih (a različnih) modelov. Primer take enačbe je kosinusni izrek v primeru c).

Seveda je vpeljava enot omejena na fizikalne enačbe, kjer imajo količine enote, pri izpeljavi matematičnih enačb pa nam ta način ne pomaga.

References

[1] Todorovski, L.: *Kontekstno-neodvisne gramatike za odkrivanje enačb* (Magistrsko delo). Ljubljana, 1998.

[2] Brence, J., Todorovski, L. Džerovski, S.: *Probabilistic Grammars for Equation Discovery* (Preprint). Ljubljana, 2021. (str.: 31–32)

[3] Knjižnjica za odkrivanje enačb ProGED. Dostopno na: `https://github.com/brencej/ProGED`.