NSU 2022/23 – četrta domača naloga

17.3.2023

Na voljo sta dve nalogi, osnovna in napredna. Vsaka je vredna 10 točk. Rešuješ lahko samo osnovno nalogo, ali pa obe.

1 Odkrivanje enačb in uporaba predznanja

V tej nalogi je cilj najti (izmišljeno) enačbo ki opisuje prenos toplote v hladilnem stolpu. Podatki so v datoteki **DN4_1_podatki.csv**. Hladilni stolp je naprava, ki se uporablja za za vrnitev toplotnih izgub v ozračje s hlajenjem vodnega pretoka na nižjo temperaturo. Za reševanje tega problema lahko uporabiš poljubni algoritem za odkrivanje enačb. Predznanja ni nujno uporabljati, se pa toplo priporoča, saj bo to močno zmanjšalo računsko zahtevnost naloge. Zanima nas povezava med toplotnim tokom ter štirimi neodvisnimi spremenljivkami:

- spremenljivka na levi strani enačbe je Q, toplotni tok v vatih (W),
- T_w je temperatura vode v stopinjah Celzija,
- T_a je temperatura zraka v stopinjah Celzija,
- θ je kot med smerjo vetra in osjo stolpa v radianih, od 0 do π (0 pomeni, da sta veter in os vzporedna),
- η je empirični izolacijski indeks, brez dimenzij.

Domensko predznanje

- Kot med smerjo vetra in osjo stolpa vpliva na toplotni tok, pri čemer se največ prenosa toplote doseže, ko veter piha pravokotno na os stolpa.
- Razlika med temperaturo vode in zraka vpliva na toplotni tok, pri čemer višje razlike v temperaturi prinašajo višji toplotni tok. To razmerje verjetno ni linearno.
- Toplotni tok verjetno ni odvisen od absolutnih vrednosti temperature vode ali zraka, ampak samo od njune razlike.
- Empirični izolacijski indeks izhaja iz izmerjenih lastnosti tovarniškega procesa izdelave in povzema termične lastnosti materiala vodnega stolpa. Nihče ga zares ne razume, vendar pa vsak ve, da toplotni tok monotono pada z višanjem izolacijskega indeksa.

Poročilo naj vsebuje:

1. katero orodje si izbral_a in zakaj,
2. natančen opis postopka reševanja, vključno z vsemi uporabljenimi nastavitvami orodja,
3. katero predznanje si uporabil_a in kako,
4. končno enačbo,
5. utemeljitev izbire enačbe ter komentar, koliko najdeni enačbi zaupaš in zakaj.

2 Verjetnostne gramatike in posodabljanje verjetnosti

Verjetnostna gramatika definira verjetnostno porazdelitev nad prostorom enačb. Porazdelitev parametrizirajo verjetnosti posameznih produkcijskih pravil (pri čemer se morajo verjetnosti pravil z istim nekončnim simbolom na levi strani sešteti v 1).

Knjižnica ProGED enačbe odkriva z Monte-Carlo vzorčenjem verjetnostne porazdelitve, ki jo definira gramatika. Z drugimi besedami, metoda generira naključne enačbe iz verjetnostne gramatike, na koncu pa vzame tisto z najnižjo napako. Ena od pomanjkljivosti tega pristopa je, da vsako enačbo generiramo neodvisno – nikoli ne upoštevamo infomacij o uspešnosti (napaki) že preizkušenih enačb. Pristop se da nadgraditi tako, da:

1. generiramo eno kandidatno enačbo (ali pa manjši nabor enačb),
2. trenutni kandidatni enačbi optimiziramo parametere in izračunamo napako,
3. informacijo o napaki enačbe uporabimo, da posodobimo verjetnosti produkcijskih pravil gramatike (na primer, če je napaka nizka, povišamo verjetnosti pravil, s katerimi je bila enačba generirana),
4. če je napaka dovolj nizka, prekinemo zanko in vrnemo najdeno enačbo ter posodobljeno gramatiko, v nasprotnem primeru pa gremo nazaj na 1. korak.

Ideja je, da nadgrajen algoritem najde pravo enačbo hitreje – z manj preizkušenimi kandidatnimi enačbami. Razvij algoritem za iterativno posodabljanje verjetnosti gramatike, ki sledi zgornjemu okvirju in ga implementiraj. Preizkusi ga na danih podatkih ter njegovo delovanje primerjaj z neodvisnim vzorčenjem. Uporabljaj gramatiko:

```
E -> E + F | E - F | F
F -> F * T | F / T | T
T -> V | (E) | sin(E)
V -> x1 | ... | xn
```

Algoritem preizkusi pri iskanju naslednjih treh enačb:

$y = x_1 - 3x_2 - x_3 - x_5,$

$y = x_1^5 x_2^3,$

$y = \sin x_1 + \sin\left(x_2 / x_1^2\right)$

Generiranje podatkov je implementirano v **DN4_2_podatki.py**. Ker navedene prave enačbe ne potrebujejo numeričnih konstant, bo preverjanje kandidatnih enačb zelo hitro. Pri evalviranju algoritma zato generiraj in testiraj vsaj nekaj tisoč kandidatnih enačb.

Poročilo naj vsebuje:

1. Opis razvitega algoritma s psevdokodo ali v besednem opisu.
2. Python skripto, ki vsebuje samo implentiran algoritem v obliki funkcije.
3. En graf za vsako od testnih enačb, ki primerja delovanje novega algoritma z neodvisnim vzorčenjem. Na x-osi naj bo število preizkušenih enačb, na y-osi pa najmanjša napaka. Vsak algoritem naj ima svojo krivuljo na istem grafu. Ker so algoritmi verjetnostni, se zanesljivost grafov lahko izboljša s ponavljanjem eksperimenta in povprečenjem rezultatov (če imamo dovolj časa in procesorske moči).
4. Diskusijo delovanja razvitega algoritma ter možnih izboljšav.

Tehnični nasveti

- Praktično je napisati funkcijo, ki sprejme numerične vrednosti verjetnosti in zgenerira gramatiko.
- Do posameznih modelov v ProGED-u lahko dostopamo z "ED.models". Na primer, napako prve enačbe dobimo z "ED.models[0].get_error()".
- Vsaka generirana enačba v ProGED-u vsebuje tudi oceno svoje verjetnosti. Na primer, "ED.models[0].p".
- Za uporabno se utegne izkazati tudi t.i. parsanje – rekonstrukcija drevesa izpeljave danega izraza z gramatiko. To nam omogoča npr. InsideChartParser iz knjižnice NLTK. Primer uporabe za linearno gramatiko z vaj:

```
from nltk.parse import InsideChartParser
g = pg.GeneratorGrammar(g)
parser = InsideChartParser(g.grammar)

parse_trees = parser.parse("x+y")
for tree in parse_trees:
    print(tree, tree.productions())
```