UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA APLIKOVANEJ INFORMATIKY

Úvod do Umelej Inteligencie
Projekt: CSP alebo viacvrstvová sieť

December 7, 2022

VŠEOBECNÉ INFORMÁCIE:

Ako projekt si vyberáte **jednu** z nasledujúcich úloh.

Povinnou súčasťou riešenia je aj súbor $README.[pdf|doc|txt]$, v ktorom popíšte svoje riešenie - aké metódy ste vyskúšali, aké štandardné/vlastné heuristiky ste použili, aké finty ste vymysleli na zlepšenie/zrýchlenie programu, ako mali jednotlivé časti vplyv na úspešnosť, aké výsledky ste dosiahli, ... Udržte prosím aj formálnu stránku tohto reportu na úrovni.
Projekt môžete odovzdať aj viackrát - pokiaľ odovzdáte predčasne už finálne riešenie, tak nám pošlite mail aby sme ho ohodnotili.
Deadline je **8. január 2023.** Pre tých, ktorí pôjdu na skúšku skôr ako je deadline (záleží na termínoch skúšky), deadline je 3 dni pred skúškou + povinnosť dať vedieť, kedy je projekt hotový, aby sme ho stihli opraviť do skúšky.
Projekt je možné odovzdať aj po deadline, avšak za každý deň meškania sa vám z výsledného počtu odrátajú 2 body.
Potrebujete aspoň 50 % bodov (t. j. 7.5b) z projektu, aby ste boli pripustení ku skúške.

PROJEKT 1: CONSTRAINT-SATISFACTION PROBLEM

Pomocou metód na riešenie CSP budete mať za úlohu vyriešiť krížovky.

Vstupné dáta:

V priloženom súbore *words.txt* máte slovník s cca 30 tis. anglickými slovíčkami, ktoré môžete použiť na vypĺňanie krížoviek.
V súbore *krizovky.txt* sú zadané jednotlivé prázdne plániky krížoviek - plánik obsahuje iba "steny" ('#') a prázdne miesto (' '), napr.:

```
#####
#   #
# # #
#   #
# # #
#   #
#####
```

Program:

Máte pripravenú kostru programu (*crossword.py*), ktorú môžete využiť. Tentoraz v nej môžete ľubovoľne meniť všetko čo budete potrebovať, príp. ju vôbec nepoužiť. Krížovka je implementovaná v triede *CrossWord*, samotný plánik je uložený ako *CrossWord.grid = list[list[string]]*, kde každá bunka poľa obsahuje jedno políčko krížovky, a indexovanie je formou $[row][column]$ (t.j. nie $[x][y]$!).
Okrem samotného programu **odovzdajte aj súbor *krizovky_out.txt***, ktorý bude obsahovať krížovky vylúštené vaším programom, a samozrejme $README$.

Algoritmus:

Na riešenie krížoviek použite backtracking (ako základ môžete použiť kód z cvičení), do ktorého postupne naprogramujete rôzne heuristiky, či iné "zlepšováky", umožňujúce správne vyriešenie krížoviek.
Porozmýšľajte hlavne nad tým, čo sa spomínalo v prednáške pri CSP problémoch ako ich riešiť efektívnejšie. Niekoľko myšlienok/návrhov, ktoré môžu pomôcť:

- **heuristika** - už na cvičení ste videli, ako veľmi heuristika pomáha pri riešení CSP problémov. Mali sme heuristiky na voľbu premenných ale aj na voľbu hodnôt.
  - voľba premenných: MRV, degree heuristics, ...
  - voľba hodnôt: least-constraining-value, ...
  - heuristiky sa dajú kombinovať.
  - dajú sa vymyslieť heuristiky vzhľadom na zadanú úlohu?
  - dajú sa vymyslieť heuristiky vzhľadom na dátovú štruktúru, v ktorej sú premenné alebo hodnoty?
  - TIP: *na každú krížovku môžete použiť iné heuristiky (prípadne inú sadu heuristík)!*
- **uzlová (node) a hranová (arc) konzistencia** - spôsob ako efektívne zúžiť domény premenných.
  - môže byť aplikovaná na počiatočné domény alebo sa môže stále zachovávať pri zmene krížovky.
  - pseudokód na hranovú konzistenciu nájdete napr. TU - str. 13

- **dátová štruktúra** - prechádzať zoznam všetkých slov môže byť nepraktické. Vieme vzhľadom na náš účel použiť niečo lepšie ako zoznam?
- **iné** - môžete použiť akúkoľvek inú, vami naimplementovanú, techniku na zefektívnenie hľadania riešení.

Z uvedených možností môžete implementovať všetko alebo aj nič. Hodnotenie je založené takmer čisto na tom, koľko krížoviek váš algoritmus dokáže vyriešiť (a ktoré) za určený čas.

Úloha (15b + bonus 2b):

Vašou úlohou bude vyplniť prázdne miesta v krížovke nasledovne:

- každé prázdne miesto musí byť vyplnené
- smery krížovky sú iba dole a vpravo, t.j. žiadne slovo nepôjde zdola hore alebo diagonálne
- každá postupnosť (aspoň dvoch) písmen medzi dvoma stenami (#) musí byť slovo z priloženého slovníka - to platí pre oba smery

Napr. krížovka hore sa dá vyplniť nasledovne:

```
#####
#era#
#x#r#
#ice#
#s#n#
#tea#
#####
```

V priloženom súbore máte 10 rôznych krížoviek, pričom posledná je bonusová. Riešiť ich môžete v ľubovoľnom poradí.

Bodovanie:

Podmienkou je korektná implementácia backtrackingu prispôsobená k úlohe. Potom, každá krížovka má určený počet bodov, ktoré dostanete za jej správne vyriešenie. Body sú rozdelené nasledovne: [0.5, 1, 1, 1, 1.5, 1.5, 2, 2, 1.5, 2]. Krížovky sú zoradené "zhruba" podľa zložitosti, no veľmi to závislosti od heuristík, ktoré používate. Preto, ak chcete hodnotiť krížovky v inom poradí, stačí nastaviť iné poradie indexov vo for cykle, v ktorom sa riešia jednotlivé krížovky. Určite **nemodifikujte** súbor "krizovky.txt" ani "words.txt", testovať to budem s pôvodnými súbormi.
Za prehľadný kód (bude sa dať spustiť bezproblémovo bez prepisovania a bude dobre čitateľný) je 1 bod. Za README sa dajú získať ďalšie 2 body.
Časový limit na beh vášho programu je 30 minút. Hodnotiť to budem na notebooku s procesorom Intel(R) Core(TM) i7-10510U. V prípade, že ste si neistý v tom, že ako dlho vám bude bežať kód a koľko bodov za to dostanete, ozvite sa mi a otestujem.

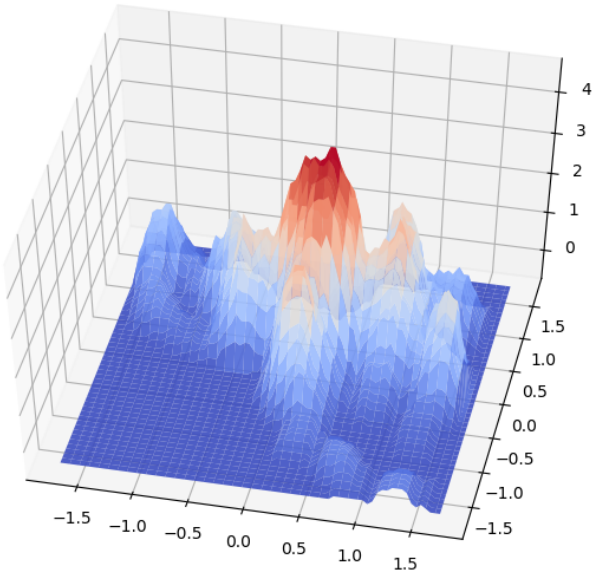
Ak by ste mali akékoľvek otázky k tomuto projektu, kontaktujte ma (email: stefan.pocos@fmph.uniba.sk alebo cez MS Teams). Email si pravdepodobne všimnem skôr.

PROJEKT 2: VIACVRSTVOVÁ NEURÓNOVÁ SIEŤ, ERROR BACKPROPAGATION

Naprogramujte viacvrstvový perceptrón a zvoľte ideálnu architektúru tak, aby sieť aproximovala 2D funkciu.

Dáta:

Aproximovať budete funkciu $f : \mathbb{R}^2 \rightarrow \mathbb{R}$, teda s dvoma vstupmi a jedným výstupom:



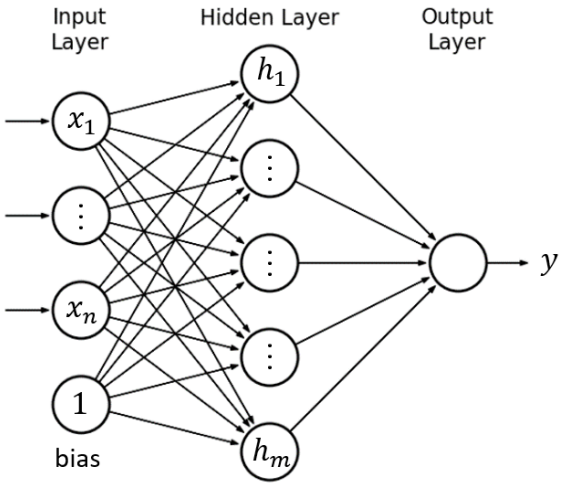
Na cvičení sme pracovali s jednoduchšou sieťou a len jednou množinou dát. No v praxi dáta zvyčajne rozdeľujeme na trénovacie a testovacie (v niektorých prípadoch aj validačné). Testovacie dáta použijeme my pri hodnotení (vy ich nemáte k dispozícii), zbytok máte aj vy a môžete ho použiť na tréning a výber modelu. Preto dáta, ktoré máte k dispozícii rozdeľte na trénovaciu a validačnú množinu. Celý tréning siete prebieha iba na trénovacích dátach, na validačných sa kontroluje generalizačná chyba (t.j. ako dobre sa sieť správa na dátach, ktoré dovtedy nevidela). Pri určovaní trénovacej a validačnej množiny treba dávať pozor na to, aby sa delili náhodne (to nám zabezpečí, že sú dáta z rovnakého rozdelenia). Preto nestačí zobrať prvých niekoľko bodov a tie prehlásiť za trénovacie a zvyšok za validačné, poradie dát treba znáhodniť. Pomer veľkostí trénovacej a validačnej množiny môžete nastaviť napríklad na 80:20, v každom prípade, počet bodov trénovacej množiny má byť oveľa väčší ako validačnej.
Prípadne môžete použiť k-fold cross validation.

K dispozícii máte 1771 dát, ktoré môžete použiť na trénovanie a validáciu, ďalších 443 testovacích dát (ktoré máme k dispozícii iba my) bude použitých na hodnotenie modelu. Dáta jednoducho načítate cez $np.loadtxt(...)$, pričom prvé dva stĺpce sú vstupy, tretí je výstup.

Model:

Váš model musí spĺňať tri podmienky:

- Pôjde o viacvrstvovú sieť, t.j. vstup, aspoň jednu strednú (skrytú) vrstvu, a výstupnú vrstvu.
- Na vstupe pridávajte aj bias, na skrytej vrstve nemusíte.
- Model bude trénovaný metódou spätného šírenia chýb (error backpropagation), vzorce na trénovanie sú uvedené nižšie.



Zvyšné parametre zvoľte tak, aby ste dosiahli čo najlepšiu presnosť. Nezabúdajte, že sieť sa bude testovať na dátach, ktoré nemáte k dispozícii, preto optimalizujte najmä generalizačnú chybu, teda maximalizujte presnosť na validačných dátach (t. j. dátach, ktoré neboli použité na tréning).
Vzorce pre počítanie výstupu viacvrstvovej siete s n vstupmi, m skrytými neurónmi h_i a jedným výstupom y:

$$net_i^{hid} = \sum_{j=1}^{n} w_{ij}^{hid} \cdot x_j$$

$$h_i = f_{hid}(net_i^{hid})$$

$$net^{out} = \sum_{i=1}^{m} w_i^{out} \cdot h_i$$

$$y = f_{out}(net^{out})$$

Vzorce pre trénovanie (backpropagation) takejto siete:

$$\delta^{out} = (d - y) \cdot f'_{out}(net^{out})$$

$$\delta_i^{hid} = (w_i^{out} \cdot \delta^{out}) \cdot f'_{hid}(net_i^{hid})$$

$$w_i^{out} := w_i^{out} + \alpha \cdot \delta^{out} \cdot h_i$$

$$w_{ij}^{hid} := w_{ij}^{hid} + \alpha \cdot \delta_i^{hid} \cdot x_j$$

Nezabúdajte, že derivácie f'_{hid} a f'_{out} musíte dať podľa toho, aké ste zvolili aktivačné funkcie f_{hid} a f_{out}. Tieto derivácie si zistite/vypočítajte presne, derivácia nemusí byť vždy tak jednoduchá ako pri sigmoide.

Povinne vyskúšajte rôzne nastavenia skrytej vrstvy (počet neurónov, aktivačná funkcia) a rôzne hodnoty rýchlosti učenia.
Okrem toho môžete zlepšiť úspešnosť vášho modelu aj ďalšími spôsobmi:

- viac skrytých vrstiev
- *momentum* pre prekonanie lokálnych miním
- *learning rate schedule* pre dynamickú úpravu rýchlosti učenia
- rôzne metódy inicializácie váh
- *early-stopping*

Program:

Ako kostru kódu použite napr. 9. cvičenie (perceptrón). Priložený máte *utils.py*, kde nájdete funkcie na kreslenie grafov a vizualizáciu dát (výstup y je reprezentovaný farbou).

Úloha (15b + bonus do 2b podľa úspešnosti):

Naprogramujte sieť tak, aby sa čo najlepšie vedela naučiť zadanú funkciu. Vyskúšajte rôzne kombinácie parametrov (povinne počet neurónov, aktivačná funkcia a learning rate) a iných vlastností siete tak, aby ste získali čo najúspešnejší model (tu vyhodnocujeme chybu na testovacej množine) - sieť potom s najúspešnejšími parametrami môžete natrénovať na všetkých dátach, ktoré máte k dispozícii. Zdrojový kód odovzdajte v takom stave, že keď sa spustí, tak sa začne učiť z trénovacích dát (t.j. netreba kód 15 minút upravovať, aby sme donútili vašu sieť učiť sa).

Implementujte aj funkcie:

- *evaluate(file_path):* načíta testovacie dáta zo súboru, sieť vypočíta svoj výsledok, a vypíše sa priemerná chyba výsledku od požadovaného výstupu
  priemerná chyba = $average((d-y)^2)$
- *save_weights(file_path):* uloží natrénované váhy do súboru/súborov
- *load_weights(file_path):* načíta pretdým uložené váhy zo súboru/súborov

Môžete priložiť aj súbor s exportovanými váhami svojho najlepšieho modelu - počas hodnotenia sa načíta pomocou *load_weights()* a vyhodnotí sa úspešnosť vášho top modelu na testovacej množine pomocou *evaluate()*.

Bodovanie:

Prvé kolo: spustí sa učenie siete na trénovacích dátach, spočíta sa priemerná chyba na trénovacích dátach (všetky dáta, ktoré máte k dispozícii), a spočíta sa priemerná chyba na testovacích dátach (naše dáta). Tieto chyby sa spriemerujú v pomere 30:70 (testovacia chyba má väčšiu váhu).
Druhé kolo: pokiaľ priložíte aj súbor s exportovanými predtrénovanými váhami, tak sa spustí druhé kolo testovania - opäť sa spočíta priemerná trénovacia a testovacia chyba, a spriemeruje v pomere 30:70.
Výsledky z dvoch kôl sa tiež skombinujú v pomere 70:30 (sieť trénovaná pri hodnotení má väčšiu váhu, predtrénovaná sieť menšiu).

Body sa pridelia podľa vzorca $b = 15 - 65 \cdot err$, kde err je priemer chýb popísaný vyššie. Ďalšie tri body hodnotia váš postup pri voľbe najlepšieho modelu (architektúry), t.j. zisťovanie ktoré parametre a vylepšenia siete zlepšujú jej úspešnosť a o koľko. Za prehľaný a okomentovaný kód a *README* sú ďalšie dva body. Plný počet (15b) teda získate, ak dôkladne vyberiete najlepší model, dosiahnete s ním priemernú chybu 1/13 (cca 0.075) alebo menšiu, a váš kód bude vyzerať slušne. Bonusové body môžete získať za presnosť vášho modelu, ak bude dosahovať priemernú chybu menšiu ako 1/13.
Časový limit na učenie siete počas hodnotenia (prvé kolo) je cca 5 minút. Sieť, ktorej exportované váhy priložíte (pre druhé kolo), môžete doma trénovať ľubovoľne dlho.
Ak by ste mali akékoľvek otázky k tomuto projektu, kontaktujte cvičiacu I. Bečkovú (email: iveta.beckova@fmph.uniba.sk alebo cez MS Teams).

Poznámka pre špekulantov: použitie knižníc ako TensorFlow, Theano, Keras, Lasagne, atď je zakázané.