Projekt nr 1

Algorytmy metaheurystyczne lub klasyfikacja

Wybierz zadanie projektowe A lub zadanie projektowe B (tylko jedno!). W szczególnym przypadku możesz zaproponować inny rodzaj zadania (jeśli obie wersje Ci nie pasują).

Zadanie projektowe A

Celem zadania jest wybranie ciekawego problemu i rozwiązanie go za pomocą różnych poznanych dotychczas algorytmów (algorytm genetyczny, strategie inteligentnego roju). Całość należy umieścić w czytelnym sprawozdaniu (np. notebooku jupyterowym), które w przejrzysty sposób będzie prezentowało rozwiązanie. Sprawozdanie (.pdf, .docx, lub .ipynb) powinno zawierać wstawki kodu, ich outputy, Twoje komentarze do kodu i wyników. Sprawodanie powinno być sensownie podzielone na rozdziały: wstęp z prezentacją problemu, rozdziały z eksperymentami, podsumowanie i bibliografię.

Co należy zrobić?

1) **Wybieramy problem do rozwiązania.** Dobrym pomysłem jest wybranie problemu NP-zupełnego (czyli takiego, dla którego nie ma szybkich algorytmów rozwiązujących). Lista problemów jest dostępna tutaj:

   https://en.wikipedia.org/wiki/List_of_NP-complete_problems

   Można wybrać i problem spoza listy. Przed dokonaniem wyboru, warto się zastanowić, czy uda nam w ogóle rozwiązać algorytm za pomocą AG/PSO/ACO. Można przejrzeć źródła internetowe i artykuły naukowe, w których być może ktoś proponował rozwiązania.

   W razie niepewności, wybór tematu można skonsultować z prowadzącym zajęcia.

   Pewne propozycje tematów znajdują się też w załączniku 1 – możesz z nich skorzystać.

2) **Potwierdzenie i rezerwacja wybranego problemu.** W obrębie grupy laboratoryjnej, każdy powinien mieć unikalny temat. Po wyborze tematu, zarezerwuj go publicznie, tak by inni wiedzieli, że temat jest zajęty. Robimy to, dodając komentarz w konwersacji na Teams, założonej przez prowadzącego zajęcia. Komentarz powinien zawierać nazwę problemu. Obowiązuje zasada: kto pierwszy ten lepszy.

   W szczególnym przypadku prowadzący może pozwolić na zduplikowany temat, jeśli osoby mają inne pomysły na jego rozwiązanie.

3) **Przygotowujemy instancje problemu do rozwiązania**. Nie zależnie od wyboru tematu, należy przygotować kilka inputów problemu do rozwiązania. Proponuję przygotować 9 inputów:
   - Trzy inputy małe (np. grafy o kilku wierzchołkach, łamigłówki o rozmiarze 5x5)
   - Trzy inputy średnie (np. grafy o parunastu wierzchołkach, łamigłówki 10x10)
   - Trzy inputy duże (np. grafy o parudziesięciu wierzchołkach, łamigłówki 15x15 czy 20x20).

Oczywiście powyżej podano tylko przykłady. Twój problem może być zupełnie inny, i wielkości inputów musisz odpowiednio dopasować do jego trudności. Możesz przygotować więcej niż 9 inputów, ale 9 to minimum, poniżej którego nie schodzimy.

Inputy można stworzyć ręcznie lub znaleźć w internecie (podając źródło do strony).

Stworzone inputy kodujemy w Pythonie.

4) **Tworzymy model algorytmu genetycznego.** Dla naszego problemu należy stworzyć algorytm genetyczny rozwiązujący go. Korzystamy z paczki pygad (chyba, ze z jakichś przyczyn chcemy skorzystać z innego narzędzia). Tutaj trzeba zwrócić uwagę na następujące aspekty:
    - Jak kodowane będą rozwiązania jako chromosomy? Należy to objaśnić w notebooku.
    - Jaka będzie funkcja fitness? Jaki ma zakres? Jak działa? Tutaj też dopisujemy komentarz objaśniający. Można ją uruchomić dla kilku sztucznie stworzonych rozwiązań.
    - Czy w moim problemie trzeba wprowadzić niestandardowe krzyżowanie/mutację? Jeśli tak, to opisujemy.
    - Dopasowanie wielkości populacji do skomplikowania problemu. Powinna być na tyle duża, żeby problem się rozwiązywał, i na tyle mała, żeby rozwiązanie było szybko znajdowane.
    - Można tez poeksperymentować z innymi parametrami (np. selekcja, mutacja).
    - Podaj lub zilustruj, jak wyglądają rozwiązania algorytmu genetycznego dla różnych inputów.

5) **Sprawdź efektywność algorytmu genetycznego.** Chcemy uzyskać odpowiedź na dwa pytania:
    - Czy algorytm genetyczny w ogóle rozwiązuje problem?
      (Jeśli mamy problem optymalizacyjny i szukamy najlepszego rozwiązania, to warto wiedzieć, jakie jest najlepsze już w trakcie wyboru inputu problemu).
    - Jeśli algorytm genetyczny rozwiązuje problem, to w jakim czasie?

    Powyższe dwa pytania można zbadać następująco:

    - Uruchom algorytm genetyczny sto razy, wybierając losowo za każdym razem jeden z inputów małych. Zlicz ile spośród tych 100 prób dało idealne rozwiązania (np. 92/100 czyli 92%). Następnie dla prób z idealnym rozwiązaniem (u nas 92), podaj jaki mają średni czas wykonania. Parametry algorytmu genetycznego (np. populacja) muszą być odpowiednio dopasowane do trudności problemu.
    - Uruchom 100 razy AG dla inputów średnich. Powtórz eksperyment, jak wyżej. Pamiętaj o odpowiedniej konfiguracji AG (np. wielkość populacji).
    - Powtórz 100 razy eksperyment dla inputów dużych.
    - Wszystkie wyniki zestaw w tabelce i na sensownym wykresie.
    - Zinterpretuj wyniki w komentarzu. Czy AG działa dobrze? Źle?

6) **Wykonaj jeszcze jeden eksperyment** (tak jak w powyższych punktach 4 i 5) z innym modelem/algorytmem. Do wyboru (jeden z poniższych):
    - Inny model algorytmu genetycznego (inne chromosomy i inna funkcja fitness).
    - Particle Swarm Optimizer (Rój cząstek).
    - Ant Colony Optimizer (Kolonia mrówek).

Dla wybranego algorytmu/modelu należy powtórzyć badanie efektywności (tak jak w pkt 5). Następnie porównać efektywność drugiego algorytmu z algorytmem genetycznym z poprzedniego eksperymentu. Dodaj interpretację/komentarz.

Uwaga! Nawet jeśli stworzyliśmy dość sensowny algorytm genetyczny i testowaliśmy go na różne sposoby, może się zdarzyć, że nie będzie on efektywnie rozwiązywał naszego problemu. Taki model można uwzględnić w sprawozdaniu, ale i tak zachęcam do dość wnikliwej analizy co działa, a co nie, i ewentualnych testów innych modelów/algorytmów.

Zadanie projektowe B

Celem zadania jest wybranie większej i ciekawszej bazy danych, odpowiedniej obróbki danych, a następnie przetestowanie, jak działają na niej poznane algorytmy klasyfikujące.

Baza danych powinna być **numeryczno-kategoryczna** (bazy danych obrazkowe lub tekstowe będą tematem drugiego projektu) i testujemy podstawowe klasyfikatory (kNN, Naive Bayes, Drzewo decyzyjne, Sieci neuronowe).

Wyniki eksperymentów należy umieścić w czytelnym sprawozdaniu (np. notebooku jupyterowym), które w przejrzysty sposób będzie prezentowało rozwiązanie. Sprawozdanie (.pdf, .docx, lub .ipynb) powinno zawierać wstawki kodu, ich outputy, Twoje komentarze do kodu i wyników. Sprawozdanie powinno być sensownie podzielone na rozdziały: wstęp z prezentacją bazy danych, preprocessing, klasyfikacja (podrozdziały dla różnych klasyfikatorów), podsumowanie i bibliografię.

Co należy zrobić?

1) **Wybieramy bazę danych do klasyfikacji.** Ważny krok to wybór odpowiedniej bazy danych. Działaliśmy na prostych bazach danych (iris.csv i diabetes.csv), teraz pora wybrać coś bardziej skomplikowanego. Zachęcam do poszukania pod linkiem

   https://www.kaggle.com/datasets

   Można na tej stronie fajnie filtrować, np. zaznaczyć „classification” i pliki csv: https://www.kaggle.com/datasets?fileType=csv&sizeStart=20%2CKB&sizeEnd=50%2CMB&tags=13302-Classification

   Alternatywna strona:

   https://archive-beta.ics.uci.edu/datasets

   Jakie cechy powinna posiadać baza danych?

   - Powinna być odpowiednio duża. Minimum parę tysięcy rekordów, najlepiej powyżej 7 kolumn. Mile widziane jednak są jeszcze większe (parędziesiąt tysięcy kolumn, kilkanaście/kilkadziesiąt kolumn).
   - Powinna być przeznaczona do klasyfikacji. Tzn. łatwo w niej znaleźć kolumnę/zmienną, którą należy odgadywać i jest to kolumna z danymi kategorycznymi (lub numeryczna, którą można zamienić na kategorie).
   - Dobrze będzie, jeśli baza będzie trochę „popsuta” 😉 Jeśli będzie z błędami, brakującymi danymi lub będzie wymagała innych technik preprocessingu to zawsze plus dla rozwiązania.

- Spróbuj znaleźć bazę, która choć trochę Cię zainteresuje. Tematyka tych datasetów jest bardzo szeroka 🧐
- Kilka pomysłów na bazy danych jest w załączniku 2. Są to jednak dość oklepane bazy danych. Postaraj się znaleźć swoją.

2) **Potwierdzenie i rezerwacja wybranego problemu.** W obrębie grupy laboratoryjnej, każdy powinien mieć unikalny temat . Po wyborze tematu, zarezerwuj go publicznie, tak by inni wiedzieli, że temat jest zajęty. Robimy to, dodając komentarz w konwersacji na Teams, założonej przez prowadzącego zajęcia. Komentarz powinien zawierać nazwę problemu (bazy danych). Obowiązuje zasada: kto pierwszy ten lepszy.

3) **Preprocessing bazy danych i przygotowanie dwóch wersji datasetu**. Bazę danych należy odpowiednio przygotować do klasyfikacji.

   Na pewno warto sprawdzić czy są błędy i brakujące dane. Jeśli tak, to usunąć je w sensowny sposób. Należy przeprowadzić inne operacje, które są niezbędne do korzystania z datasetu.

   Przygotuj dwie wersje bazy danych: jedną mniej przetworzoną, a drugą bardziej. Na obu przetestujesz klasyfikację w dalszej części zadania. Druga: bardziej przetworzona baza danych może uwzględniać np. PCA lub balansowanie. Można też na niej przetestować jakieś ciekawe metody imputacji brakujących danych.

4) **Trenujemy i testujemy klasyfikatory na obu wersjach bazy danych.** Dla obu wersji bazy danych trenujemy i testujemy klasyfikatory. Na początku oczywiście dzielimy bazę danych na zbiór testowy i treningowy (i ewentualnie walidacyjny). Następnie trenujemy po kolei klasyfikatory na zbiorze treningowym. Każdy z klasyfikatorów testujemy na zbiorze testowym. Podajemy jego dokładność (accuracy) oraz macierz błędów (najlepiej w formie graficznej), a w przypadku sieci neuronowych również krzywą uczenia się (learning curve) uwzględniająca zbiór treningowy i walidacyjny. Na koniec robimy podsumowanie klasyfikatorów dla obu wersji bazy danych. Który zadziałał najlepiej?

   Klasyfikatory do testowania to:
   1) Drzewo decyzyjne (w wersji mniejszej z przyciętymi gałęziami i większej).
   2) Naiwny Bayes.
   3) K-Najbliższych Sąsiadów (dla paru różnych k, może wykres porównujący k)
   4) Sieć neuronowa (dla paru topologii, dla paru konfiguracji uczenia np. optimizer, learning rate, batch size, itp).

5) Dodatkowo, w bazie danych szukamy reguł asocjacyjnych. Należy ustalić sensowne progi dla minimalnego wsparcia i wiarygodności. Następnie wypisać reguły. Z reguł należy wypisać kilka najciekawszych i je zinterpretować.

Terminy i ocenianie

Czas na zrobienie zadania (termin oddania): **3 grudnia 2023 (niedziela), godz. 23:59.**

W tym terminie należy dołączyć sprawozdanie do założonego zadania na Teamsach. Nie dołączamy dużych plików (baz danych), można (ale nie trzeba) dołączać pliki pythonowe.

Projekt będzie prezentowany przez ich autorów na wyświetlaczu przed całą grupą laboratoryjną. Następnie będzie oceniony. Na prezentację poświecimy jedne zajęcia – **5 grudnia**. W szczególnym przypadku mogę ocenić projekt bez prezentacji.

Praca domowa oceniana jest na **maks. 6 punktów**. Oceniane będą poniższe aspekty.

	Zadanie projektowe A	Zadanie projektowe B
Trudność problemu	Czy problem jest dość trudny (nawet dla człowieka)? Czy został wybrany problem dość niestandardowy czy „oklepany”?	Czy baza danych jest skomplikowana i ma błędy lub braki danych? Czy jest ona dość słabo przebadana czy już wydaje się „oklepana”?
Przygotowanie do eksperymentów	Przygotowano wiele ciekawych instancji problemu o różnych wielkościach	Przygotowano bazę danych do klasyfikacji (preprocessing).
Zakres eksperymentów	Czy uruchomiono różne algorytmy? Czy uruchomiono algorytmy dla różnych inputów?	Czy przygotowano dwie wersje bazy danych? Czy dla obu uruchomiono komplet klasyfikatorów? Czy szukano reguł asocjacyjnych.
Konfiguracja algorytmów	Czy AG został odpowiednio skonfigurowany? Fitness/chromosomy/populacja, itp.	Czy klasyfikatory (zwłaszcza sieci neuronowe) zostały odpowiednio skonfigurowane? Czy testowano ich różne wersje? Np. różne uczenia się sieci neuronowych.
Ocena algorytmów	Czy wyniki zostały zbadane pod względem efektywności i ewentualnie czasu? (uśrednienie powtórzeń)	Czy klasyfikatory zostały odpowiednio zewaluowane (dokładność, macierz błędu, krzywa uczenia się)?
Sprawozdanie	Czy sprawozdanie jest przejrzyste i kompletne?	