Sztuczna inteligencja w informatyce biomedycznej

Projekt 03: Uczenie sfederowane w zastosowaniach klinicznych

Szymon Wilk

4 kwietnia 2024

Spis treści

1 Wprowadzenie

Cele tego cyklu zajęć laboratoryjnych są następujące:

1. Zapoznanie się z koncepcją uczenia sfederowanego (*federated learning*, FL) oraz z przykładowym środowiskiem symulacyjnym NIID-Bench[1] implementującym różne algorytmy agregacji modeli (m.in. FedAvg, FedProx) [1].
2. Zmodyfikowanie dostarczonego kodu pozwalające na symulowanie różnych rozkładów danych dla poszczególnych klientów, np. symulowanie sytuacji, gdzie w procesie uczenia biorą udział bardzo zróżnicowane (pod względem liczby obsługiwanych pacjentów i rozkładu klas) jednostki opieki zdrowotnej.
3. Wykonanie eksperymentu obliczeniowego, wzorowanego analizie przedstawionej w pracy [1], którego celem będzie sprawdzenie wpływu niespójności rozkładów danych między klientami na jakość uzyskanego modelu predykcyjnego oraz porównanie dwóch algorytmów agregacji – FedAvg oraz FedProx.

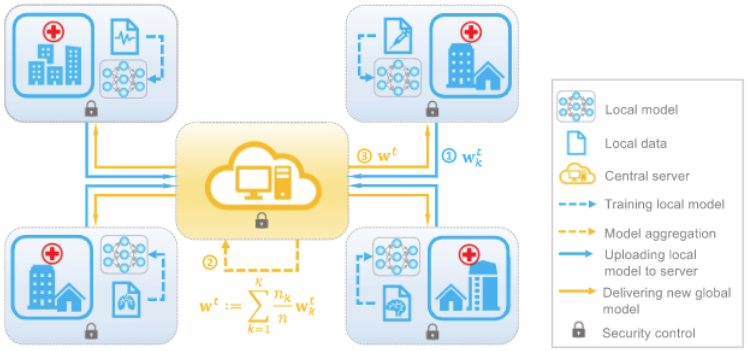
Uczenie maszynowe (*machine learning*, ML) stosowane jest z powodzeniem w wielu dziedzinach życia (np. do tłumaczenia tekstu, rozpoznawanie i oznaczania obrazów). Jest ono również stosowane w medycynie, jednak wciąż jeszcze nie osiągnęło swojego pełnego potencjału pomimo pewnych spektakularnych rozwiązań (np. certyfikowanych systemów służących do autonomicznego diagnozowania na podstawie zdjęć dna oka [4]). Głównym problemem jest ograniczony dostęp do dużych i zróżnicowanych zbiorów danych, co wynika z ograniczone współpracy pomiędzy jednostkami oraz koniecznością zapewnienia odpowiedniego bezpieczeństwa i prywatności danych medycznych.

Uczenie sfederowane (Rysunek 1) odpowiada na powyższe wyzwania zmieniając sposób postępowania podczas budowy modeli. Zamiast agregować dane w celu budowy scentralizowanego modelu, modele uczone są lokalnie (z wykorzystaniem danych dostępnych w danej lokalizacji – np. instytucji lub urządzeniu), a następnie agregowane. Dzięki temu dane nie muszą być współdzielone, a modele można dodatkowo zabezpieczyć przed możliwością

[1] https://github.com/Xtra-Computing/NIID-Bench

rekonstrukcji danych, na podstawie których powstały, stosując techniki prywatności różnicowej (*differential privacy*).

Jednym z pierwszych praktycznych zastosowań uczenia sfederowanego było tworzenie modeli predykcyjnych dla klawiatury Gboard stosowanej w systemie Android[2]. Od tego czasu (2017 rok) podejście to zyskuje na popularności, zarówno w zastosowaniach medycznych, jak i w innych dziedzinach, np. w *Internecie przedmiotów* (*Internet of things*, IoT) [5].



Rysunek 1. Schemat przedstawiający ideę uczenia sfederowanego w zastosowaniach klinicznych (źródło: [3]) z wykorzystaniem scentralizowanego serwera.

2 Zadania

2.1 Przygotowanie środowiska symulacyjnego

1. Pobierz repozytorium ze środowiskiem NIID-Bench zmodyfikowanym na potrzeby zajęć (**https://github.com/sysmon37/fl-bench**) -- stanowi on zmodyfikowaną wersję kodu opublikowaną przez autorów pracy [1]. W szczególności dodano obsługę „zewnętrznych" zbiorów danych w formacie CSV (w oryginalnej wersji rozważane zbiory były „zaszyte" na stałe).
2. Zainstaluj biblioteki niezbędne do uruchomienia obliczeń – lista w pliku `requirements.txt`.
3. Wykonaj wstępne obliczenia dla zbioru *pima* (powinien zostać on umieszczony w folderze `data`) testując różne algorytmy agregacji oraz techniki przydziału przykładów do

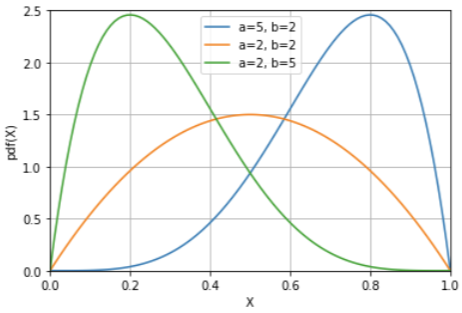
[2] https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

klientów. Obliczenia wykonaj dla 5 klientów, 20 rund komunikacyjnych oraz pozostałych ustawień przedstawionych poniżej (w szczególności stosujemy prosty model MLP):

```
--dataset=pima
--model=mlp
--device=cpu
--comm_round=20
--n_parties=10
--partition=homo | noniid-diff-quantity
--alg=fedavg | fedprox
```

2.2 Zadanie do samodzielnego wykonania

1. Funkcja custom_partition. `assign_samples_to_clients` odpowiada za przydzielanie przykładów uczących do poszczególnych klientów. Obecnie implementuje ona rozkład jednostajny – przykłady dzielne są równomiernie pomiędzy zadaną liczba klientów. Zmodyfikuj tę funkcję, aby można było zasymulować dodatkowo jedną z następujących sytuacji: (1) przewaga klientów ze „średnią” (względnie) ilością danych, (2) przewaga klientów z małą ilością danych (dużo małych ośrodków i kilka dużych) oraz (3) przewaga klientów z dużą ilością danych. Możesz zainspirować się rozkładem prawdopodobieństwa *beta* (Rysunek 2) albo zastosować rozkład Poissona. Dodatkowo rozważ dwa warianty każdej z powyższych sytuacji, w których rozkład klas jest zachowany (taki sam, jak dla całego zbioru – rozbieżności dotyczą liczby obiektów) oraz losowy (w tym celu można wykorzystać rozkład Dirichleta bazując na obecnej implementacji → funkcja `partition.partition_data`, dodatkowy opis w pracy [6]).
   Z technicznego punktu widzenia dodać nowe wartości dla parametru `partition`, które będą przetwarzanie w funkcji `assign_samples_to_clients`.



Rysunek 2. Rozkłady beta i ich parametry wykorzystane jako podstawa eksperymentów symulacyjnych w pracy [2].

2. Wykonaj eksperyment dla wskazanego zbioru medycznego – *clinical*. Są to rzeczywiste dane opisujące pobytu pacjentów na oddziale intensywnej terapii (*intensive care unit*, ICU). Aby utrudnić identyfikację zbioru, cechy i ich wartości zostały zakodowane. Atrybut decyzyjny oznacza śmieć pacjenta na oddziale – problem jest niezrównoważony (śmiertelność na poziomie 10%).
3. W eksperymencie rozważ 6 wariantów rozkładów danych oraz dwa algorytmy do agregacji – *FedAvg* oraz *FedProx* (drugi z nich powinien pozwolić na uzyskanie lepszej zbieżności i trafności zagregowanego modelu). W eksperymentach przyjmij liczbę klientów równą 20 oraz 30 rund komunikacyjnych. Wykorzystaj model MLP (podobnie jak poprzednio) oraz domyślne wartości pozostałych parametrów.

Po zrealizowaniu zadania przygotuj krótki raport (3-4 strony) opisujący sposób zamodelowania zmieniającego się rozkładu danych oraz wyniki uzyskane podczas eksperymentów obliczeniowych. **Zalecane jest wykonanie zadania w grupach 4 osobowych**.

Zadanie wykonaj do 25 kwietnia.

3 Literatura

1. Li, Q., Diao, Y., Chen, Q., & He, B. (2021). *Federated Learning on Non-IID Data Silos: An Experimental Study*. https://arxiv.org/abs/2102.02079

2. Budrionis, A., Miara, M., Miara, P., Wilk, S., & Bellika, J. G. (2021). Benchmarking PySyft Federated Learning Framework on MIMIC-III Dataset. *IEEE Access*. https://doi.org/10.1109/ACCESS.2021.3105929
3. Xu, J., Glicksberg, B. S., Su, C., Walker, P., Bian, J., & Wang, F. (2021). Federated Learning for Healthcare Informatics. *Journal of Healthcare Informatics Research*, 5(1), 1–19. https://doi.org/10.1007/S41666-020-00082-4/TABLES/2
4. Grzybowski, A., Brona, P., Lim, G., Ruamviboonsuk, P., Tan, G. S. W., Abramoff, M., & Ting, D. S. W. (2020). Artificial intelligence for diabetic retinopathy screening: a review. *Eye*, 34(3), 451. https://doi.org/10.1038/S41433-019-0566-0
5. Hosseinzadeh, M., Hemmati, A., & Rahmani, A. M. (2022). Federated learning-based IoT: A systematic literature review. *International Journal of Communication Systems*, e5185. https://doi.org/10.1002/DAC.5185
6. Li, L., Zhan, D. Chuan, & Li, X. Chun. (2022). Aligning model outputs for class imbalanced non-IID federated learning. Machine Learning. https://doi.org/10.1007/s10994-022-06241-5