Zadanie 01

Bliskie spotkania trzeciego stopnia z ${FAVOURITE_FRAMEWORK}

Pozornie celem zadania jest zaznajomienie się z prostym klasyfikatorem binarnym jakim jest **regresja logistyczna**. Tak naprawdę jednak jest to pretekst, by zacząć się oswajać z wybranym przez siebie frameworkiem wspomagającym obliczenia tensorowe (najlepiej gdyby wspierał też *out-of-the-box* uczenie maszynowe - i.e. zawierał gotowe implementacje funkcji, o których mówimy na zajęciach).

[UWAGA: w ramach tego ćwiczenia NIE korzystamy z gotowej pełnej implementacji regresji logistycznej - naszym zadaniem jest poskładanie jej samodzielnie z nieco drobniejszych klocków.]

Właściwe narzędzia

O jaki framework chodzi? Wybór jest po twojej stronie! Najpopularniejsze opcje to:

- TensorFlow [https://www.tensorflow.org/], zwykle wspierany przez (od niedawna w pełni zintegrowany) Keras [https://keras.io/];
- PyTorch [https://pytorch.org/], często w "błyskawicowym" opakowaniu [https://www.pytorchlightning.ai/];
- MXNet (https://mxnet.apache.org/);
- Caffe [https://caffe.berkeleyvision.org/];

(z czego w praktyce rynek zdominowały dwie pierwsze: https://www.kaggle.com/kaggle-survey-2021). Opcji jest oczywiście jeszcze więcej - ale im bardziej niszowe rozwiązanie, tym mniejsza dostępność materiałów wspomagających naukę i odpowiedzi na typowe pytania. Tak czy siak: wybór należy do ciebie!

1. Wybierz framework, z którego będziesz chciał korzystać na przedmiocie (przyda się na wszystkich zajęciach). Poinformuj o swoim wyborze i motywacjach ("ma ładniejsze logo" to czasem też jest dobry argument ;]).

Ciuchy z Zalando

Do realizacji zadania potrzebny nam będzie odpowiednio przygotowany zbiór danych. Zaczniemy od czegoś prostego: skorzystamy ze zbioru **Fashion-MNIST** zawierającego wycentrowane zdjęcia różnych typów ubrań, zebrane na portalu Zalando. Zbiór jest dostępny na portalu GitHub: https://github.com/zalandoresearch/fashion-mnist (w praktyce większość wymienionych w poprzedniej sekcji bibliotek zawiera dedykowane wsparcie do jego pobierania - bo to bardzo popularny startowy problem do wszelkich eksperymentów).

W przypadku tego zadania będziemy musieli dostosować pierwotny stan zbioru do naszych potrzeb.

2. Po pierwsze - potrzebujemy ograniczyć go tylko do dwóch klas Najlepiej mocno różniących się - ułatwi nam to wizualizację wyników. Prowadzący poleca wybrać sukienki (klasa numer 3) i sandały (klasa numer 5).

1. Prawdopodobnie wczytaliśmy nasz zbiór do czterech struktur - po dwóch na zbiór treningowy i testowy. Dwie z nich to tensory trójwymiarowe o wymiarach (60000, 28, 28) i (10000, 28, 28) - zawierające poszczególne zdjęcia. Pozostałe dwa to tensory jednowymiarowe (wektory) z informacją o tym, do jakiej klasy należą.
2. Potrzebujemy odfiltrować elementy należące do interesujących nas klas (w efekcie zmniejszając zbiory 5 razy).
    1. By to zrobić, możemy zacząć od przygotowania nowych wektorów, zawierających binarną maskę z oznaczonymi elementami do zachowania.
    2. Taką maskę trzeba następnie zaaplikować, wycinając to co nas interesuje.
3. Wykonaj selekcję właściwych zdjęć.
    1. Napisz w raporcie informację o tym, jakich funkcji użyłaś/eś do tego celu.
    2. Sprawdź, czy na pewno się udała - podejrzyj zawartość kilku zdjęć ze skurczonego zbioru.

Model decyzyjny - regresja logistyczna

Teraz zajmiemy się implementacją samego modelu. Wiele bibliotek oferuje dedykowane mechanizmy do określania, że coś jest modelem - wymagając podania jakie jest wejście, wyjście, z czego się składa, jak będą dobierane i oceniane jego parametry.

3. Jak rozwiązano sprawę konstrukcji modeli w wybranym przez ciebie frameworku? Napisz o tym (bardzo krótko i ogólnikowo) w raporcie.

Wstępna obróbka i skrawanie

4. Musimy zacząć od dostosowania naszych wejściowych fotografii do formatu, na którym dobrze pracuje regresja logistyczna (wektorów liczbowych o małym rozrzucie wartości). W tym celu wykonujemy poniższe kroki.
    1. Najpierw zamieniamy obrazy (macierze) ze zbioru danych na wektory [biblioteka powinna oferować funkcję "spłaszczającą"].
    2. Potem normalizujemy wartości w poszczególnych wektorach, tak by zawierały się w zakresie od 0 do 1 [ponownie - powinna istnieć gotowa funkcja normalizująca/skalująca].
    3. Na koniec należy zamienić etykiety klas na 1 (klasa A) i 0 (klasa B) - tak by można je traktować jak prawdopodobieństwo.
    4. Napisz w raporcie jakich funkcji użyłeś i co robią z danymi. Pamiętaj, że powinny być częścią struktury modelu (i obsłużyć też nowe obserwacje przychodzące w przyszłości).
    5. Na tym etapie warto sprawdzić, że to co robimy rzeczywiście działa. Czy biblioteka, której używasz wspiera debugowanie "na żywo"? Napisz o tym w raporcie.

Tu podejmowane są decyzje

5. Kontynuujmy potok przetwarzania w modelu decyzyjnym, implementując samą regresję logistyczną.
    1. Na początek potrzebujemy funkcji liniowej działającej na wejściowych wektorach.
        1. Najłatwiejszy sposób na jej zrealizowanie to wykorzystanie mechanizmu, który implementuje gęsto połączoną (*dense*) warstwę sieci neuronowej o jednym wyjściu [znów - powinien nam tu pomóc framework]. Dokładne wyjaśnienie dlaczego pojawi się na kolejnych zajęciach.
    2. Potem potrzebujemy przetworzyć uzyskane wyniki na prawdopodobieństwo.
        1. Tutaj pomoże funkcja logistyczna (*logistic function*), znana też jako sigmoida (*sigmoid*). Takie dalsze przetwarzanie wyniku funkcji liniowej jest często nazywane

aktywacją (*activation*) [i prawdopodobnie pod taką nazwą trzeba szukać go w wybranej bibliotece].
3. Zaimplementuj całość, sprawdź czy uzyskiwane wyniki mają sens (od strony matematycznej).

6. Wagi funkcji liniowej (te zawarte w gęstej warstwie) można potraktować jako formę "filtra" albo "soczewki" spoglądającej na wejściową fotografię - jest ich przecież dokładnie 28*28.
    1. Dowiedz się, jak dobrać się do nich w swojej bibliotece.
    2. Pobierz ich startowe wartości.
    3. Zamień wektor wartości na macierz o rozmiarze (28, 28) - proces odwrotny do tego, który był stosowany przy zamianie fotografii na wektor.
    4. Wyplotuj taki obraz - co widzisz? Prawdopodobnie losowy bezsensowny szum. ;]

Pora na edukację

7. Pozostaje tylko dopasować nasz model do rzeczywistości (danych treninowych),
    1. W tym celu skorzystamy z mechanizmu dopasowywania (*fit*) modelu do danych [powinna wspierać go biblioteka].
        1. Potrzebny jest nam sposób oceny jak "dobre" jest dopasowanie - skorzystamy z binarnej entropii krzyżowej (*binary cross entropy*) [powinna istnieć gotowa implementacja]. O tym co to właściwie jest było już trochę na PSI - temat ma też szansę wrócić na kolejnych zajęciach.
        2. Potrzebujemy też mechanizmu, który znajdzie takie parametry, dla których jest dopasowanie będzie ocenione "najlepiej" - tu użyjemy stochastycznego spadku po gradiencie (*stochastic gradient descent*) - jak wyżej, było o nim na PSI, może jeszcze wróci jako temat [implementację powinna oferować biblioteka].
    2. Zaimplementuj taki sposób strojenia modelu.
        1. Napisz w raporcie jakich elementów biblioteki użyłeś.
    3. Uruchom strojenie na danych treningowych i poczekaj, aż się skończy. Najlepiej wykonaj kilka-kilkanaście iteracji po wszystkich obserwacjach: tzw. epok (*epochs*).
    4. Jak teraz wygląda "filtr" uzyskany z wag modelu (ten sam, który podglądaliśmy w punkcie 6.)? Czy ma jakiś sens, biorąc pod uwagę co chcieliśmy rozpoznawać?
    5. Czy nasz model podejmuje właściwą decyzję (odróżnia sukienki od sandałów) w przypadku danych treningowych? Sprawdź na kilku przykładach.

Zrobione, co teraz, jak żyć?

W ramach zaliczenia zadania wrzuć dwa pliki:

- PDF z raportem zawierającym wspomniane w tekście informacje.
- ZIP zawierający kod, który realizuje wszystkie powyższe zadania.

PS. Nie panikuj (choć biblioteka może na początku przytłoczyć). W razie problemów - pytaj prowadzącego! Dobrej zabawy i powodzenia!