Sztuczna Inteligencja i Inżynieria Wiedzy
Lista 5: Sieci neuronowe

Przemysław Dolata, Julita Bielaniewicz

maj 2023

1 Cel listy

Celem ćwiczenia jest zapoznanie się z sieciami neuronowymi - jednym z modeli najpowszechniej dziś używanych w uczeniu maszynowym. Realizując zadania, dowiesz się jak zastosować sieci neuronowe do praktycznego problemu regresji. Poznasz podstawy działania tego modelu oraz przebadasz wpływ jego najistotniejszych hiperparametrów, a także nauczysz się rozpoznawać typowe dla niego zjawiska powodujące obniżenie jakości predykcji.

2 Wprowadzenie

Termin „sieci neuronowe" obejmuje niezwykle szeroką klasę systemów służących do przetwarzania informacji, których wspólną cechą jest struktura: składają się z *neuronów*, z reguły zorganizowanych w *warstwy*, połączonych między sobą połączeniami o przypisanych *wagach*. W tym ćwiczeniu skupisz się na modelu MLP (od ang. *Multilayer Perceptron*, w polskiej literaturze *perceptron wielowarstwowy*). Jest to jednokierunkowa sieć neuronowa o jednej warstwie wejściowej, jednej lub dwóch ukrytych, i jednej wyjściowej. Rozmiar warstwy wejściowej jest zdefiniowany przez wymiar wektora danych X, podobnie rozmiar warstwy wyjściowej zależy od wymiaru wektora wynikowego Y; dobór liczby i rozmiarów warstw ukrytych jest zadaniem osoby projektującej sieć.

Model MLP, jak wiele innych sieci neuronowych, można uczyć metodą propagacji wstecznej w trybie nadzorowanym, i takie podejście zastosujesz w tym zadaniu. Baza ucząca będzie złożona z pewnego zbioru tekstów wraz z etykietami; celem procesu uczenia będzie znalezienie funkcji mapującej tekst X w etykietę Y, innymi słowy pozwalającej na automatyczne etykietowanie przyszłych tekstów.

Przy dobieraniu sieci neuronowych do konkretnego problemu, kluczowy jest nie tylko dobór architektury sieci oraz *hiperparametrów* opisujących jej strukturę i proces uczenia, ale też funkcji kosztu, która będzie optymalizowana podczas uczenia. Wybór tej funkcji z reguły zdefiniowany jest przez typ problemu do rozwiązania i dostępne dane. Z tego punktu widzenia problemy najczęściej dzieli się na:

- klasyfikację, gdzie poszukuje się rzutowania $f(\theta, X) \to Y$, gdzie Y jest zmienną nominalną; zmienne nominalne nie podlegają uporządkowaniu, przykładami są np. gatunek zwierzęcia, marka pojazdu,

- regresję, gdzie szukane jest rzutowanie $f(\theta, X) \to Y : Y \in \mathbb{R}$, czyli Y jest zmienną porządkową, a więc podlegającą uporządkowaniu, np. wiek, długość.

W problemach klasyfikacji z reguły stosuje się funkcję logistyczną (*regresja logistyczna*), natomiast w problemach typu regresji jedną z częściej stosowanych funkcji kosztu jest błąd średniokwadratowy. W obu wariantach proces uczenia polega na minimalizacji funkcji kosztu. Wartość tej funkcji stanowi zatem najistotniejszy sygnał, jaki należy obserwować podczas uczenia modelu, by ocenić postęp procesu.

Uczenie modeli neuronowych metodą gradientu prostego lub stochastycznego jest procesem iteracyjnym. W każdej iteracji wykonywana jest *propagacja w przód*, tj. model odpytywany jest na próbce ze zbioru uczącego, następnie obliczana jest wartość funkcji błędu, co umożliwia wreszcie wykonanie *propagacji wstecz* (ang. *backpropagation*), czyli obliczenia wartości gradientu ∇ dla każdego neuronu w sieci za pomocą *reguły łańcuchowej*. Następnie wagi neuronów są aktualizowane według wzoru

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla$$

gdzie η jest hiperparametrem nazywanym *współczynnikiem uczenia* (ang. *learning rate*). Istnieją jednak bardziej skomplikowane metody optymalizacji, dokonujące aktualizacji wag w bardziej wyszukany sposób. Wykonanie tej procedury dla wszystkich próbek w zbiorze uczącym nazywa się *epoką* (ang. *epoch*); z reguły nauczenie modelu neuronowego wymaga powtarzania procesu przez wiele epok.

Sieci klasy MLP, jak wiele innych algorytmów uczących się, wymagają by dane miały reprezentację wektorową. Jeśli więc dane nie mają takiej struktury, kluczowe jest odpowiednie przekształcenie ich do takiej postaci. W ogólności proces ten nazywa się *ekstrakcją cech*, a w kontekście zastosowań do przetwarzania języka naturalnego często mówi się o *tokenizacji*, czyli rozbiciu tekstu na sekwencję unikatowo oznaczonych „tokenów" (np. pojedynczych słów lub sylab). Ponieważ proces pełnej tokenizacji jest dość skomplikowany, na potrzeby niniejszego zadania przeprowadzisz prostszy wariant wektoryzacji danych tekstowych oparty o tzw. słownik. Procedura polegać będzie na wyznaczeniu zbioru D najbardziej interesujących słów i nadaniu im unikatowych indeksów; następnie każdy tekst z bazy danych zostanie przekształcony na wektor $x \in \mathbb{N}^D$, zawierający na pozycji x_i liczbę wystąpień słowa o indeksie i. Pamiętaj, by na wszystkich etapach pracy z siecią - uczenia, testowania, czy odpytywania na nowych danych - dokonywać wstępnego przetwarzania danych w ten sam sposób (m.in. dokonywać wektoryzacji w oparciu o ten sam słownik).

3 Zadania

W ramach tej listy podejmiesz problem predykcji poziomu śmieszności podanego żartu ze zbioru Jester. Zbiór ten zawiera około 25 tys. ocen śmieszności 100 różnych żartów, które obejmują przedział (-10.0, 10.0).

Aby uprościć ścieżkę i umożliwić Ci skupienie się na przebadaniu samej sieci neuronowej, moduł ekstrakcji cech stanowi załącznik do tej instrukcji. Twoim zadaniem będzie nauczenie prostej sieci neuronowej typu MLP i przebadanie kluczowych hiperparametrów. Wykorzystaj pozyskaną podczas realizacji poprzedniej listy wiedzę dotyczącą podziału danych na zbiór uczący i walidacyjny. Do realizacji zadań wykorzystaj pakiet Weka lub możliwości języka Python (głównie pakiet `sklearn` oraz `matplotlib`).

Istotne jest, by zacząć eksperymenty od możliwie jak najbardziej surowej sieci neuronowej. Jeśli korzystasz z Pythona i pakietu `sklearn`, konkretnie klasy `sklearn.neural_network.MLPRegressor`, upewnij się, że przekazujesz następujące wartości hiperparametrów:

- `solver = 'sgd'`
- `alpha = 0.0`
- `learning_rate = 'constant'`

W przypadku Weki i modułu `MultilayerPerceptron` ustaw:

- `learningRate = 0.001`
- `momentum = 0.9`
- ...i pamiętaj, by kolumna zawierająca etykiety dla danych miała format numeryczny - w przeciwnym razie sieć zostanie uruchomiona w trybie klasyfikacji!

Oczywiście, nie musisz sztywno trzymać się podanych wartości - patrz: lista zadań, punkt 6.

Szczegółowa lista zadań:

1. Przygotuj zbiór uczący i walidacyjny, wykorzystując dołączony do listy kod procedury ekstrakcji cech. Jeśli zamierzasz korzystać z Weki, zalecane jest wykonanie jednorazowego przekształcenia danych i eksportu do jednego ze zgodnych formatów. (5 punktów)
2. Przetestuj działanie podstawowego modelu MLP o domyślnej konfiguracji hiperparametrów, ucząc go na danych ze zbioru Jester. Prześledź zachowanie modelu w czasie, wizualizując wartość funkcji kosztu w funkcji liczby epok, zwracając uwagę na wartości dla zbioru uczącego i zbioru walidacyjnego. (20 punktów)
3. Zbadaj wpływ tempa uczenia (*learning rate*) na osiągane wyniki: powtórz uczenie dla 3 różnych wartości parametru. Dobierz odpowiednią długość procesu uczenia (liczbę epok) jeśli to konieczne. Przedstaw wyniki na wykresach jak w zadaniu poprzednim. Co dzieje się, gdy tempo uczenia jest zbyt niskie? Co, gdy zbyt wysokie? (30 punktów)
4. Zbadaj wpływ rozmiaru modelu MLP na jakość działania: wykonaj co najmniej 3 eksperymenty dla modeli różniących się liczbą neuronów. Kiedy model przestaje dobrze dopasowywać się do danych? Kiedy zaczyna zanadto dopasowywać się do zbioru uczącego? (30 punktów)
5. Wybierz najlepszy uzyskany w drodze powyższych eksperymentów model i przetestuj go w praktyce: znajdź (lub napisz własny) tekst o charakterze dowcipu, przetwórz go na wektor za pomocą używanej w zadaniach metody ekstrakcji cech, a następnie odpytaj model neuronowy. Czy predykcja zgadza się z Twoim oczekiwaniem? (15 punktów)
6. $\star$ Wykonaj badanie dowolnego innego parametru, np. regularyzacji. Jakie jest jego działanie? Z jakim problemem pozwala on walczyć? Jak wpływa na wyniki? (bonus do 10 punktów)

Wynikiem Twojej pracy powinna być implementacja eksperymentów w Pythonie lub Wece, oraz raport zawierający wyniki tych eksperymentów w formie tabel i/lub wykresów, wraz z krótkim omówieniem. Pamiętaj by podać pełen zestaw hiperparametrów użytych przy każdym eksperymencie. Raport wyślij prowadzącemu przynajmniej na 24 godziny przed oddaniem listy.

4 Literatura

1. Wykłady prof. Piaseckiego (wykład 8)
2. Zbiór Jester
3. Tadeusiewicz R., M. Szaleniec - *Leksykon sieci neuronowych*
4. Przewodnik użytkownika `scikit-klearn`, „Neural network models (supervised)"