mikołaj morawiec

WIEiT AGH
Informatyka III r.
Pracownia Projektowa

Analiza sentymentu recenzji przy użyciu rekursywnych sieci neuronowych

OPIS TEORETYCZNY

1 Model

1.1 Ogólny opis

Podstawowym elementem składowym sieci neuronowej w zastosowanym modelu jest bramka GRU. Zastosowano wersję bramki specjalnie dostosowaną do pracy w strukturze binarnego drzewa parsingu, tj. bramkę pobierającą dwa wektory stanu ukrytego (od dzieci w drzewie) i przekazującą jeden taki wektor do rodzica w drzewie.

Każda z bramek jest również zaopatrzona w moduł zamieniający stan ukryty na wektor prawdopodobieństw możliwy do porównania z sentymentem pobranym ze zbioru SST. Moduł zasadniczo opiera się na pojedynczej warstwie sieci redukującej wymiar wektora oraz funkcji transferu do prawdopodobieństw.

W liściach stosowana jest dużo prostsza bramka nazwana bramką wejściową. Jest ona również zaopatrzona we wspomniany moduł z funkcją transferu.

1.2 Bramka Wejściowa

Do przetworzenia reprezentacji wektorowej słów w liściach drzewa projekt wykorzystuje zwykłą sieć jednowarstwową. Sieć przetwarza wektor wejściowy (standardowo 300 wymiarowy) na wektor o mniejszej wymiarowości, odpowiadającej wymiarowości stanu ukrytego w bramce GRU.

$$h_i = \tanh(W_1^x x_i) \tag{1.2.1}$$

1.3 Bramka GRU

W projekcie wykorzystano bramkę binarną bramkę GRU, pobierającą stany ukryte od dwóch dzieci w drzewie. Równania bramki takiej bramki GRU prezentują się następująco:

$$\begin{aligned} z &= \sigma(W_1^z h_{1,t-1} + W_2^z h_{2,t-1}) \\ r &= \sigma(W_1^r h_{1,t-1} + W_2^r h_{2,t-1}) \\ \bar{h} &= \tanh(W_1^h(h_{1mt-1} \circ r) + W_2^h(h_{2,t-1} \circ r)) \\ h_t &= (1-z) \circ \bar{h} + z \circ h_{t-1} \end{aligned} \tag{1.3.1}$$

Stany ukryte dzieci w drzewie parsingu oznaczono symbolami $h_{1,t-1}$ i $h_{2,t-1}$. W tym przypadku indeks t

Stan ukryty bramki GRU zaimplementowano za pomocą tensora o wymiarze zazwyczaj między 100 a 200 (dla 300-wym. reprezentacji słów). Do jego zamiany na tensor prawdopodobieństw (*log-likelihood*) wykorzystuje się funkcję softmax, wraz z pojedynczą warstwą sieci do redukcji wymiaru ze 120 do 5 (lub 3).

1.4 Funkcje transferu

Podstawową funkcją transferu w projekcie była funkcja softmax, połączona z pojedynczą warstwą dla redukcji wymiaru stanu ukrytego do liczby klas (5 lub 3).

Wykonano próby z zastosowaniem funkcji ReLU ($ReLU(x) := max(0, x)$)). Ponieważ wyniki nie zachwycały, zdecydowano się zaniechać jej wykorzystywania.

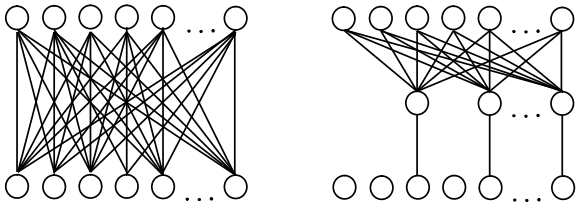
1.5 Dropout

Testy wykazały że zastosowanie warstwy dropout, tj. funkcji maskującej część wejścia dla wa w module funkcji transferu pozwala na uzyskanie nieco lepszych wyników, rzędu kilku punktów procentowych skuteczności. Zastosowany w projekcie moduł usuwa połowę składowych wejścia i mnoży pozostałe przez 2. Składowe do usunięcia są losowane (schemat Bernoulliego) spośród wszystkich składowych wektora wejściowego.

Można zapisać wyjście y_t z bramki GRU o stanie ukrytym h_t za pomocą następujących równań:

$$\begin{aligned} h'_t &= \text{dropout}(h_t) \\ y_t &= \log(\text{softmax}(W_1^y h'_t) \end{aligned} \tag{1.5.1}$$

Działanie dropout ilustruje poniższy rysunek. Po lewej przedstawiono warstwę sieci bez dropout, po prawej usunięto część wejścia.



Rysunek 1: Ilustracja dropout

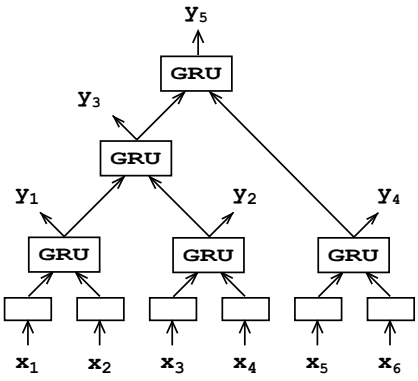
1.6 Funkcja kosztu NLLC

Do uczenia wykorzystuje się kryterium NLLC (*negative log-likelihood*), odpowiednie do problemu klasyfikacyjnego. Tensor prawdopodobieństw uzyskuje się ze stanu ukrytego bramki GRU wstawia do kryterium, wraz indeksem odpowiadającym poprawnej klasie do której powinien być zakwalifikowany dany przypadek.

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n} y_t^i + \frac{\lambda}{2}||\theta||^2 \tag{1.6.1}$$

Parametr λ mnożony przez kwadrat normy wektora parametrów służy do regularyzacji kryterium. Symbolem y_t^i oznaczono prawdopodobieństwa uzyskane z funkcji softmax (rów. wyżej).

2 Struktura drzewa



Rysunek 2: Drzewiasta sieć neuronowa

Strukturę drzewa osiągnięto za pomocą odpowiedniego mechanizmu obliczającego wejścia dla węzłów w drzewie po kolei, zaczynając od liści, tj. pojedynczych słów. Wejścia podawane są do sieci w kolejności postorder DFS.

Uczenie odbywa się algorytmem BTS. Dla każdego węzła drzewa tworzona jest kopia sieci neuronowej. Dla każdego węzła wykonywana jest propagacja do przodu i wstecz, wszystko odbywa się kolejności wynikającej z przebiegu algorytmu DFS na drzewie, przy czym dzieci po lewej stronie (wczejśiej w zdaniu) eksplorowane są jako pierwsze.

Obrazek obok ilustruje strukturę pełnej sieci. Indeksy oznaczają kolejność obliczania kolejnych wyjść przy propagacji do przodu.

3 Uczenie, Selekcja najlepszej sieci

3.1 Charakterystyka SST

Uczenie odbywa się na wspomnianym zbiorze SST. Zbiór jest podzielony na trzy części: treningową, optymalizacyjną i testową. Cały zbiór zawiera łącznie 11855 zdań zbudowanych z 14926 różnych wyrazów. Należy zaznaczyć że jako wyrazy traktowane są również znaki interpunkcyjne, nawiasy itd. Najkrótsze ze zdań ma dwa wyrazy (tj. jedno słowo i kropkę, brzmi ono: „Crummy .”). Najdłuższe zdanie ma 72 wyrazy. Zdania w zbiorze pochodzą w większości z różnorodnych recenzji tekstów kultury.

Zbiór treningowy jest zawiera 8544 zdań. Służy do uczenia sieci zgodnie z opisanym wyżej algorytmem BTS. Zbiory optymalizacyjny i testowy są wykorzystywane wyłącznie do propagacji w przód (ewaluacji). Zbiór optymalizacyjny zawiera 1101 zdań i jest wykorzystywany do oceny skuteczności sieci w trakcie procesu uczenia (po każdej z epok). Zbiór testowy zawiera 2210. Służy wyłącznie do ostatecznej oceny skuteczności wyuczonej sieci.

3.2 Wartości sentymentu

SST zawiera oprócz zdań wartości sentymentu przypisane do każdej z fraz jak i do całego zdania. Należy nadmienić, że zdania posiadają gotowe drzewa parsingu zapisane w zbiorze w notacji parent-pointer (pol.: notacja spaghetti). Sentyment jest przedstawiony jako liczba z zakresu $[0, 1]$.

Funkcja kosztu jest obliczana na podstawie prawidłowych wartości sentymentu, przy czym konieczna jest ich konwersja z wersji ciągłej do dyskretnej, realizowana odpowiednią funkcją, w zależności od rozpatrywanego wariantu problemu (tj liczby klas):

$$f_V(x) = \begin{cases} 0 & x \in [0, 0.2) \\ 1 & x \in [0.2, 0.4) \\ 2 & x \in [0.4, 0.6) \\ 3 & x \in [0.6, 0.8) \\ 4 & x \in [0.8, 1] \end{cases} \qquad \text{albo} \qquad f_{II}(x) = \begin{cases} 0 & x \in [0, 0.4) \\ 0 & x \in [0.6, 1] \end{cases} \tag{3.2.1}$$

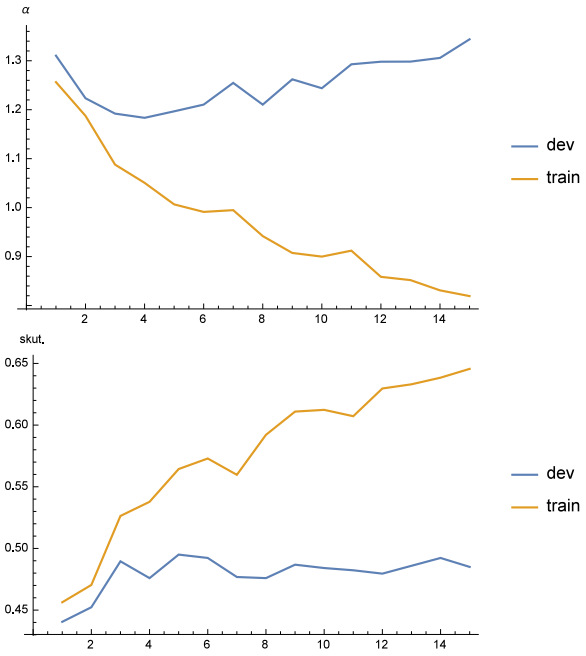
Tę samą operację stosuje się przy obliczaniu skuteczności przy ewaluacji.

3.3 Proces uczenia

Uczenie odbywa się na podzbiorze treningowym SST. Kolejność podawania zdań do sieci jest ustalana od nowa w losowy sposób dla każdej kolejnej epoki, efektem czego jest fakt, iż procesy uczenia nie są powtarzalne dla ustalonego zestawu parametrów sieci. Ten fakt wskazuje na stochastyczny aspekt całego procesu uczenia i testowania.

Do selekcji optymalnie wyuczonej sieci stosuje się przeznaczoną do tego celu część optymalizacyjną SST. Wybiera się sieć po epoce w której osiągnęła ona najlepszy wynik (tj. skuteczność dla całych zdań).

W testach stosowano przeważnie 25 epok, wybierając sieć o najlepszym wyniku w działaniu na wspomnianej części zbioru. Proces selekcji ilustruje wykres:



Rysunek 3: ilustracja 15 epok uczenia. Po lewej stronie koszt, po prawej stronie skuteczność sieci na całych zdaniach

W powyższym przypadku sieć po epoce nr 5 zostałaby wybrana jako wynik procesu uczenia i przekazana do testowania. Na wykresach zilustrowano również koszt i skuteczność sieci na zbiorze treningowym.