Pakiet perceptronu wielowarstwowego

Mateusz Bochenek, Karol Rzepka i Piotr Zawadka

Styczeń 2019

1 Temat projektu

Naszym zadaniem było napisanie biblioteki implementującej perceptron wielowarstwowy, który do uczenia wykorzystuje metodę najszybszego spadku z bezwładnością.

2 Sposób rozwiązania/najważniejsze decyzje projektowe

Z założenia, na wejście klasy reprezentującej sieć neuronową przekazywana jest informacja o ilości wejść sieci oraz ilości neuronów w kolejnych warstwach.
Inicjalizacja sieci przebiega następująco:
Wagi neuronów wyjściowych są bliskie zeru, a wagi neuronów warstw ukrytych są losowane zgodnie z rozkładem: $U(-1/\sqrt{dim(we)}, 1/\sqrt{dim(we)})$
Jako funkcja aktywacji neuronów warstwy ukrytej została przyjęta następująca funkcja sigmoidalna:

$$\psi(z) = \frac{e^z}{1 + e^z} \tag{1}$$

2.1 Wsteczna propagacja gradientu

Na początku przechodzimy przez sieć, aby uzyskać wyjścia z neuronów, następnie stosujemy wsteczną propagację gradientu. Tutaj do obliczeń w skorzystaliśmy ze wzoru:

$$\frac{\mathrm{d}q}{\mathrm{d}\theta_{i,j}^{K}} = \frac{\mathrm{d}q}{\mathrm{d}s_i^K y_j^{K-1}} = (y_i - y_i^d) y_j^{K-1} \tag{2}$$

do obliczenia gradientów funkcji kosztu po wagach neuronów warstwy wyjściowej oraz:

$$\frac{\mathrm{d}q}{\mathrm{d}\theta_{i,j}^{k}} = \frac{\mathrm{d}q}{\mathrm{d}s_i^k} y_j^{k-1} = \frac{\mathrm{d}q}{dy_j^k} \frac{\mathrm{e}^{s_j^k}}{(1+\mathrm{e}^{s_j^k})^2} y_j^{k-1} = \frac{\mathrm{e}^{s_j^k}}{(1+\mathrm{e}^{s_j^k})^2} y_j^{k-1} \sum_i \frac{\mathrm{d}q}{\mathrm{d}s_i^{k+1}} \theta_{i,j}^{k+1} \tag{3}$$

do obliczenia gradientów funkcji kosztu po wagach neuronów warstw ukrytych, a z kolei do policzenia pochodnej $\frac{\mathrm{d}q}{\mathrm{d}s_i^{k+1}}$, skorzystaliśmy z zależności:

$$\frac{\mathrm{d}q}{\mathrm{d}\theta_{i,j}^{k}} = \frac{\mathrm{d}q}{\mathrm{d}s_i^k} y_j^{k-1} \implies \frac{\mathrm{d}q}{\mathrm{d}\theta_{i,j}^{k+1}} = \frac{\mathrm{d}q}{\mathrm{d}s_i^{k+1}} y_j^k \implies \frac{\mathrm{d}q}{\mathrm{d}s_i^{k+1}} = \frac{\mathrm{d}q}{\mathrm{d}\theta_{i,j}^{k+1}} \frac{1}{y_j^k} \tag{4}$$

gdzie dla k = 1,...,K:
s_i^k jest to suma obliczana przez i-ty neuron k-tej warstwy,
y_j^{k-1} jest wartością obliczaną przez j-ty neuron warstwy *k-1*-szej,
q jest funkcją kosztu,
a $\frac{\mathrm{d}q}{\mathrm{d}\theta_{i,j}^{k}}$ jest pochodną, która mówi nam o oddziaływaniu wagi $\theta_{i,j}^k$ na q.

2.2 Najszybszy spadek

Ta metoda polega na iteracji po wagach i ich zmianie za pomocą wzoru:

$$\theta_{t+1} = \theta_t - \beta_t g_t, \tag{5}$$

gdzie β_t jest parametrem kroku (w naszym przypadku - inicjowanym zawsze tą samą wartością)
a g_t jest nieobciążonym estymatorem $\nabla J(\theta_t)$, czyli jest równe $\frac{d}{d\theta_t^T} \frac{1}{2} ||\mathrm{f}(\mathrm{x_i}; \theta_\mathrm{t}) - \mathrm{y_t}||^2$

3 Krótki opis najważniejszych elementów

3.1 Klasy

neuron - klasa reprezentująca neuron, czyli określająca jego wejścia, wagi oraz zawierająca metody do podstawowych operacji na neuronie, np. sumator wejść, funkcja aktywacyjna

mlp - klasa reprezentująca sieć neuronową, przechowuje m.in. informacje o ilościach warstw, liczbie neuronów w każdej z warstw oraz zawiera w sobie metody służące m.in. do wykonywania propagacji wstecznej czy też estymujące gradient za pomocą sochastycznego najszybszego spadku

3.2 Metody

processData() - metoda symulująca przepływ danych przez perceptron.

sochasticDescent() - metoda przeprowadzająca sochastyczny najszybszy spadek, korzysta ze wzoru nr. 5

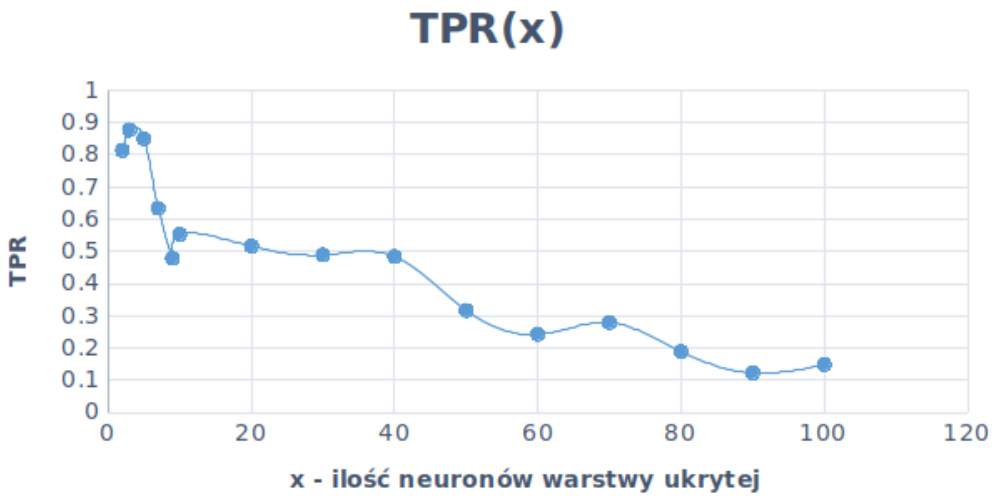
propagateBackwards() - metoda przeprowadzająca propagację wsteczną, wykorzystuje wzory nr. 2, 3, 4

processDataAndLearn() - metoda symulująca przepływ danych (metoda *processData()*) oraz naukę (metody *propagateBackwards(), sochasticDescent()*), aż do określonej precyzji wyniku

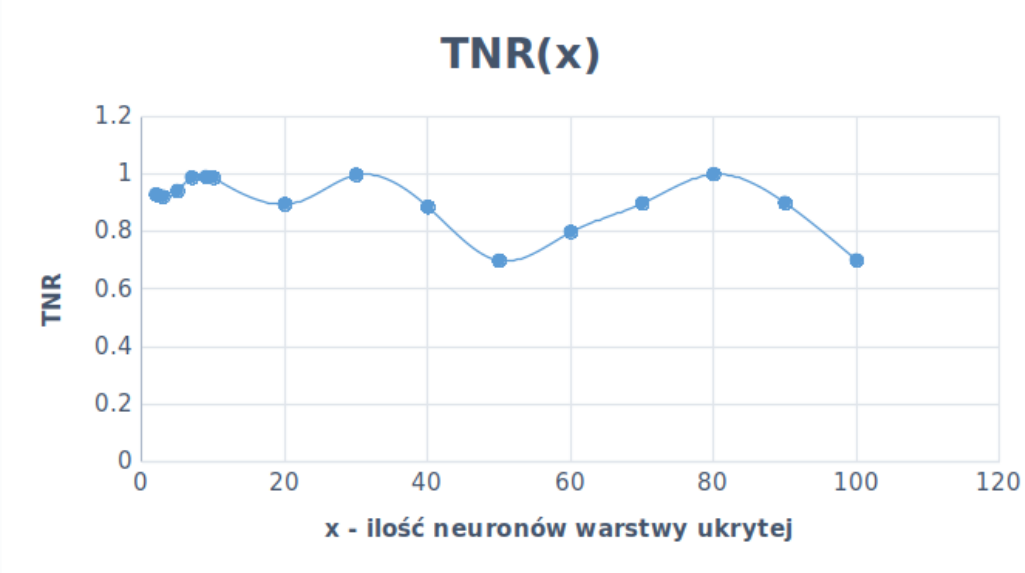
countOuterError(outputValue, expectedOutputValue, inputNumber),
countInnerError(errors, layer, neuronNumber, input)
countInputError(errors, neuronNumber, input)
- metody odpowiedzialne za obliczanie błędu w neuronach odpowiednio warstwy wyjściowej, warstw ukrytych oraz warstwy wejściowej

4 Badanie zdolności aproksymacyjnych sieci

Do zbadania zdolności aproksymacyjnych sieci użyty został „Breast Cancer Wisconsin (Diagnostic) Data Set„. Zawiera on opis 569 komórek. Każda komórka jest opisana przy pomocy 32 atrybutów przy czym na wejście sieci podawane jest ostatnie 30 (pierwsze dwa to id oraz diagnoza). Komórkę można zakwalifikować do jednej z dwóch grup: nowotwór złośliwy, nowotwór łagodny. Zadaniem sieci jest zwrócenie prawdopodobieństwa z jakim u danej komórki zdiagnozowano nowotwór złośliwy.
Badania były prowadzone głównie dla perceptronów dwuwarstwowych o różnej liczbie neuronów warstwy ukrytej. Losowe 25% instancji problemu stanowiło zbiór walidacyjny, a reszta zbiór testowy. Dla każdej sieci wykonywane było 10 testów na kopiach danego obiektu przy czym za każdym razem dobierane były inne zbiory walidacyjne i testowe. Po każdym teście zliczana była ilość obiektów należących do każdej z klas, liczba poprawnych klasyfikacji oraz liczona była macierz pomyłek (confusion matrix).
Przy ocenie wpływu ilości neuronów warstwy ukrytej na jakość klasyfikacji używane były 2 współczynniki: **TPR** (*True Positive Rate*) i **TNR** (*True Negative Rate*). Wartości na wykresie odpowiadają średnim arytmetycznym tych współczynników z 10 testów dla każdej sieci.



$$TPR = \frac{TP}{TP + FN}$$



$$TPR = \frac{TN}{TN + FP}$$

4.1 Wyniki testów

Z wykresów można odczytać, że funkcja TPR(x) osiąga najwyższe wartości dla x<10, co oznacza, iż w tym przedziale sieć radzi sobie najlepiej z wykrywaniem komórek z nowotworem złośliwym.

Domyślna wartość na wyjściu sieci jest równa 0, dlatego dla x>50 można zaobserwować wysokie wartości współczynnika TNR, podczas gdy w wartości współczynnika TPR w tym samym przedziale są bliskie 0. Oznacza to, że wraz z zwiększaniem się ilości neuronów warstwy ukrytej, aproksymowana funkcja coraz bardziej przypomina funkcję stałą. Lokalne spadki wartości współczynnika TNR są prawdopodobnie spowodowane tym, że dla odpowiednio wysokiej liczby neuronów warstwy ukrytej model predykcyjny odzwierciedla też szum zawarty w konkretnych danych.

Po dokładniejszym zbadaniu zdolności aproksymacyjnych sieci dla okazało się, że najlepsze wyniki osiągane są przez sieć o czterech neuronach warstwy ukrytej. Oto wyniki zwracane przez program (jeden zestaw testów składający się z 10 testów):

Test : {4, 1}
Distribution M:B ->55:88 Properly classified: 135 in 143 examples. That is 94.4056 percent.
TRUE POSITIVE: 52 FALSE POSITIVE: 5 TRUE NEGATIVE: 83 FALSE NEGATIVE: 3
Distribution M:B ->59:84 Properly classified: 135 in 143 examples. That is 94.4056 percent.
TRUE POSITIVE: 58 FALSE POSITIVE: 7 TRUE NEGATIVE: 77 FALSE NEGATIVE: 1
Distribution M:B ->50:93 Properly classified: 129 in 143 examples. That is 90.2098 percent.
TRUE POSITIVE: 46 FALSE POSITIVE: 10 TRUE NEGATIVE: 83 FALSE NEGATIVE: 4
Distribution M:B ->50:93 Properly classified: 134 in 143 examples. That is 93.7063 percent.
TRUE POSITIVE: 46 FALSE POSITIVE: 5 TRUE NEGATIVE: 88 FALSE NEGATIVE: 4
Distribution M:B ->59:84 Properly classified: 133 in 143 examples. That is 93.007 percent.
TRUE POSITIVE: 55 FALSE POSITIVE: 6 TRUE NEGATIVE: 78 FALSE NEGATIVE: 4
Distribution M:B ->47:96 Properly classified: 119 in 143 examples. That is 83.2168 percent.
TRUE POSITIVE: 24 FALSE POSITIVE: 1 TRUE NEGATIVE: 95 FALSE NEGATIVE: 23
Distribution M:B ->52:91 Properly classified: 125 in 143 examples. That is 87.4126 percent.
TRUE POSITIVE: 48 FALSE POSITIVE: 14 TRUE NEGATIVE: 77 FALSE NEGATIVE: 4
Distribution M:B ->50:93 Properly classified: 126 in 143 examples. That is 88.1119 percent.
TRUE POSITIVE: 33 FALSE POSITIVE: 0 TRUE NEGATIVE: 93 FALSE NEGATIVE: 17
Distribution M:B ->50:93 Properly classified: 131 in 143 examples. That is 91.6084 percent.
TRUE POSITIVE: 49 FALSE POSITIVE: 11 TRUE NEGATIVE: 82 FALSE NEGATIVE: 1
Distribution M:B ->50:93 Properly classified: 127 in 143 examples. That is

88.8112 percent.
TRUE POSITIVE: 34 FALSE POSITIVE: 0 TRUE NEGATIVE: 93 FALSE NEGATIVE: 16

Zdolności aproksymacyjne sieci były też badane dla kilku perceptronów wielowarstwowych. Zgodnie z oczekiwaniami dla perceptronów o odpowiednio „szerokich" warstwach ukrytych jakość aproksymacji była bardziej zadowalająca. Przykładowo dla sieci 7, 3, 2, 1 średnie arytmetyczne TPR i TNR wynoszą około 0.91.