Sieć Neuronowa Rozpoznająca Litery
Sprawozdanie

Bartłomiej Bułat
Konrad Malawski

I rok, 2 stopień, Informatyka Stosowana, EAIiE

12 kwietnia 2012

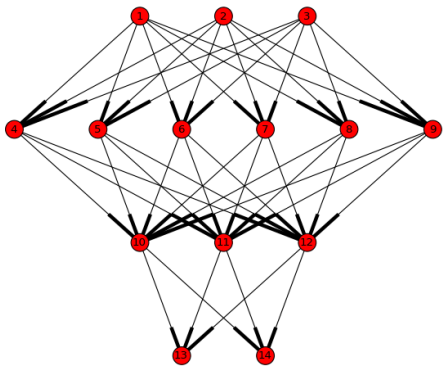
1 Wstęp

Celem projektu była implementacja oraz zbadanie właściwości sieci neuronowej mającej za cel rozpoznawanie wielkich liter alfabetu łacińskiego, dostępnych jako macierze o rozmiarach 5x7.

Sieci zaprojektowano przy wykorzystaniu biblioteki Feed-Forward Neural Network for Python - ffnet. Zbudowane sieci uczono zadanych wzorców różnymi metodami (Patrz sekcja 2), a następnie sprawdzano ich efektywność w rozpoznawaniu obrazów liter poddawanych coraz to większym zakłóceniom.

2 Wybrane struktury i metody uczenia sieci neuronowych

Podczas przeprowadzanych testów struktura sieci pozoztawała bez zmian. Modyfikowaliśmy jedynie zastosowane algorytmu uczące. Zastosowaliśmy metodę `mlgraph` generującą standardową wielopoziomową strukturę sieci neuronowej, analogiczną do przedstawionej na Rysunku 1.



Rysunek 1: Schemat warstwowej sieci neuronowej

Sieć złożona jest z 3 warstw (w tym jednej ukrytej):

- 35 neuronów w warstwie wejściowej - odpowiadojącej ilości elementów macierzy przedstawiającej litery,
- 10 neuronów warstwy ukrytej,
- oraz 26 neuronów warstwy wyjściowej odpowiadające kolejno każdej literze alfabetu.

Wybór metod uczenia sieci, celem uzyskania możliwie dużego przeglądu, postanowiliśmy przetestować wszystkie udostępniane przez „ffnet” metody, tj:

- `train_momentum` - prostą metodę ze wsteczną propagacją błędów z zachowaniem “momentu”, służącemu zapobieganiu utkwienia sieci w lokalnym minimum (lub siodle),

- `train_rprop` - metoda z wsteczną propagacją błędu. Tę samą metodę zastosowano również na zajęciach podczas 2gich zajęć laboratoryjnych,
- `train_cg` - metoda gradientu sprzężonego,
- `train_bfgs` - metoda opierająca się na algorytmie BFGS,
- `train_tnc` - metoda uczenia wielozadaniowego,

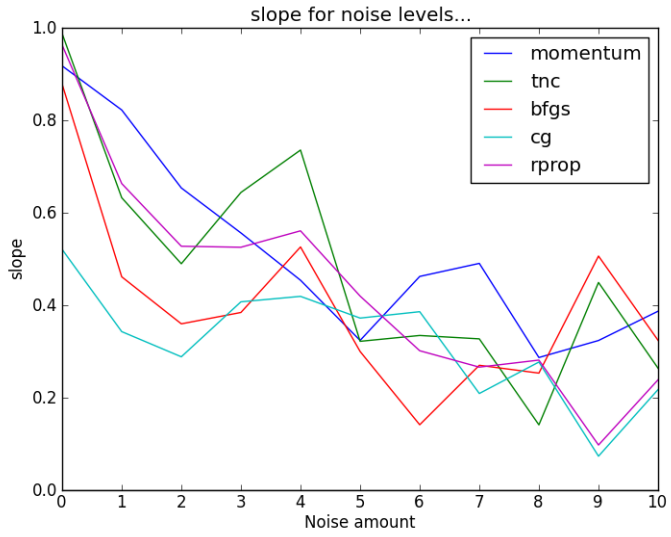
Na podstawie poniżej umieszczonych wyników testów będzie można ocenić która z metod uczenia sieci najlepiej sprawdziła się w naszym przypadku.

3 Analiza regresji liniowej

Poniżej przedstawiono przebiegi testów dla każdego z wyżej wspomnianych algorytmów. Na osi X zaznaczono poziom szumów które dodano do badanych obrazów. Dodawanie szumu wykonano poprzez N-ktorne (gdzie N to "poziom szumu") wylosowanie punktu w macierzy 5x7 (reprezentującej badaną literę) w którym wykonano odwrócenie wartości logicznej. To znaczy jeżeli dane pole było częścią litery, oznaczaną wartościami 1, to zmieniano ją na 0, oraz analogicznie w przeciwnym przypadku.

Przedstawione poniżej wyniki zostały uzyskane przy pomocy wbudowanej funkcji `test` służącej zbadaniu charakterystyk sieci poddawanej testom na podstawie podanego wejścia oraz oczekiwanego wyjścia.

3.1 Nachylenie

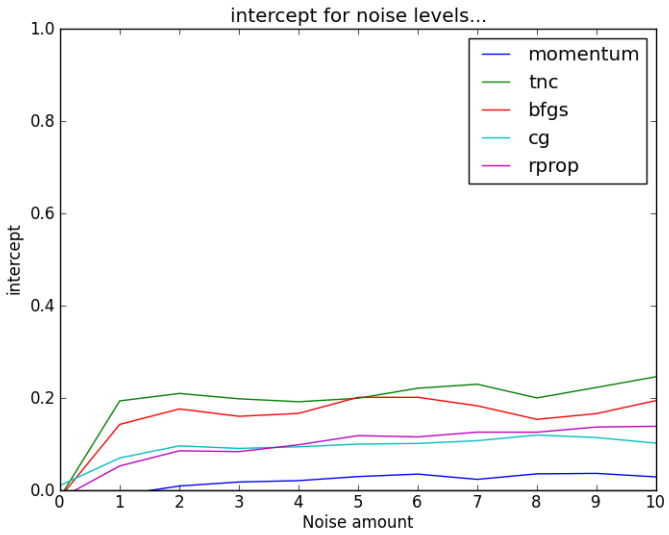


Rysunek 2: Wykres wartości slope

Nachylenie (ang. „*slope*") w najlepszym przypadku może przyjąć wartość 1, ponieważ krzywa regresji powinna przebiegać pod kątem 45 stopni przez punkt

0,0. Jak widać na wykresie 6, metody proste bezproblemowo radzą sobie w tym przypadku zanim dodamy zakłócenia do badanych liter. Metoda gradientu sprzężonego (cg) nie poradziła sobie zbyt dobrze przy zerowym zakłóceniu, jednak warto zauważyć iż dodając kolejne zakłócenia wartość nachylenie uzyskana przez ten algorytm uczenia nie pogarsza się tak drastycznie jak w przypadku innych metod.

3.2 Przesunięcie



Rysunek 3: Wykres wartości slope

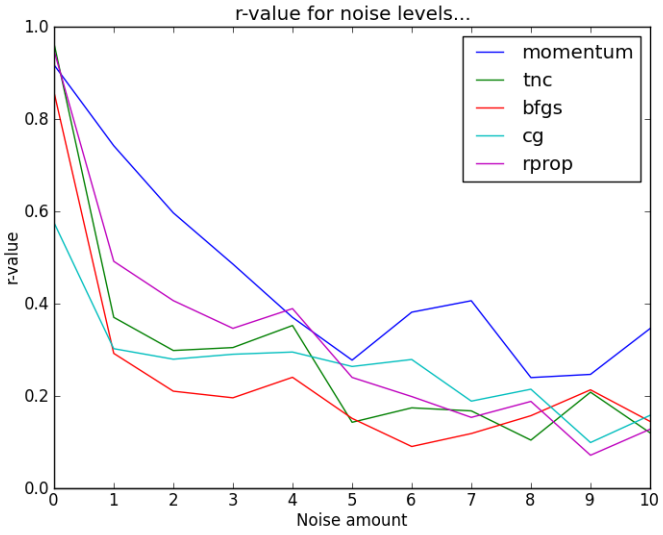
Optymalną wartością przesunięcia (ang. „*intercept*") jest 0. Ponownie widać, że metoda momentum niewiele oddala się od wartości optymalnej wraz z wzrostem zaszumienia wzorów liter poddawanych próbom rozpoznania.

3.3 R-Value

Współczynnik korelacji (r-value) między wartościami oczekiwanymi oczekiwanym a otrzymanym wyjściem. Powinien być oczywiście możliwie bliski 1. Najefektywniejszą przy nawet dużym zaszumieniu tutaj ponownie okazała się metoda ze wsteczną propagacją błędu z momentem.

3.4 Błąd Standardowy - Stderr

Analizując błędy standardowe zastosowanych algorytmów łatwo jest ocenić który z nich najlepiej „nauczył" naszą sieć. Najgorzej sprawdziła się metoda uczenia wielozadaniowego (wykorzystująca BGFS), posiadając we wszystkich pomiarach (poza początkowym - z brakiem szumów). Kolejnym algorytmem jest BFGS wspomniany już, z tym, że tym razem w wersji jedno zadaniowej.



Rysunek 4: Wykres wartości R-Value

Pozytywnie wyróżniają się bardziej zaawansowane metody, to znaczy: metoda uczenia z gradientem sprzężonym (cg) oraz metoda ze wsteczną propagacją błędu z momentum. Wyraźnie widoczna jest różnica w jakości sieci wygenerowanej przy pomocy metody ze wsteczną propagacją „z" i „bez" elementu momentum - okazuje się on na tyle dobry iż przyrównuje metodę tą do skomplikowanej metody wykorzystującej gradient.

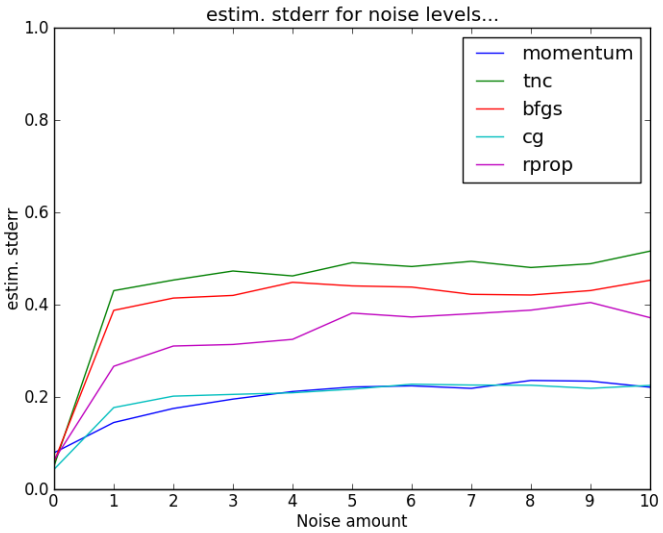
4 Podsumowanie

Najlepszą metodą okazała się być metoda uczenia ze wsteczną propagacją błędu z „momentem". Okazała się nawet skuteczniejsza niż bardziej zaawansowana metoda jaką jest metoda gradientu sprzężonego.

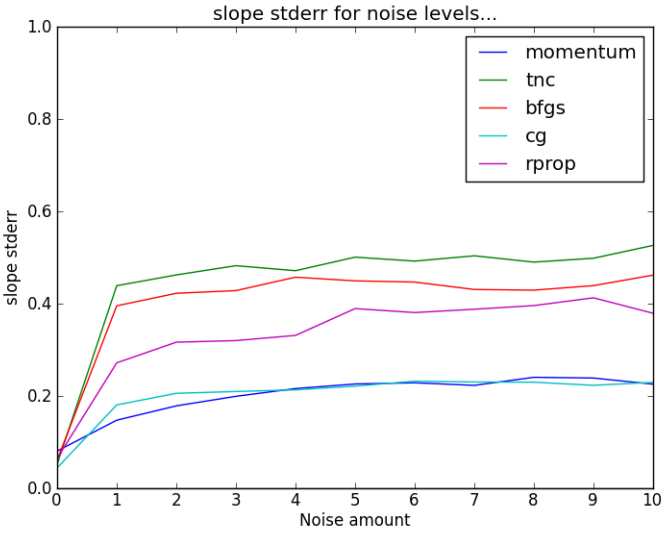
Warto wspomnieć iż przy wszystkich wybranych algorytmach uczących zastosowano domyślne parametry. Dalsze testy oraz próby optymalizacji dobranych parametrów mogłyby pozytywnie wpłynąć na uzyskiwaną przez sieci efektywność.

5 Bibliografia

- Artykuł o analizie regresji liniowej: http://www.mp.pl/artykuly/index.php?aid=10899
- Strona domowa projektu ffnet: http://ffnet.sourceforge.net/



Rysunek 5: Wykres wartości Estimate StdErr



Rysunek 6: Wykres wartości Slope StdErr