Politechnika Wrocławska
Wydział Elektroniki
Informatyka

Zastosowania informatyki w medycynie

Analiza stopnia złośliwości komórek nowotworowych wykorzystująca sieci neuronowe.

Autorzy:
Łukasz Matysiak
Kamil Machnicki

Prowadzący:
Dr inż. Bartosz Krawczyk

Termin zajęć:
Czwartek, godz. 11:15

Wrocław
25 maja 2016

Spis treści

Rozdział 1

Wstęp

1.1 Cele i założenia projektowe

Celem projektu było zapoznanie się z tematem uczenia maszynowego, a konkretnie z sieciami neuronowymi. Realizacja projektu miała umożliwić zrozumienie zasady działania oraz porównanie wydajności sieci neuronowych używających propagacji wstecznej oraz wariantu o nazwie *Extreme Learning Machines.*

1.2 Wstęp teoretyczny

Sieci neuronowe stanowią jedno z możliwych podejść do tematu uczenia maszynowego. Są one modelowane na podobieństwo neuronów w mózgu, a zastosowanie znajdują między innymi w problemach klasyfikacji – np. klasyfikacji stopnia złośliwości raka, tak jak to miało miejsce w projekcie.

Wzrost popularności zawdzięczają pojawieniu się szybszych komputerów oraz ogromnej ilości danych dostępnych do uczenia ich, co pozwoliło na zredukowanie wpływu głównych wad sieci neuronowych – wolnego uczenia się oraz wymagania dużych zestawów danych uczących.

1.2.1 Budowa sieci neuronowych

Prosta sieć neuronowa może być zbudowana z trzech warstw neuronów:

- warstwy neuronów wejściowych,
- warstwy neuronów ukrytych,
- warstwy neuronów wyjściowych.

Każda z warstw odbiera dane, a następnie po obliczeniu wartości funkcji aktywacji, przesyła dane do kolejnej warstwy.

Rozdział 2

Aplikacja

2.1 Wykorzystane technologie

Aplikacja implementująca sieci neuronowe na potrzeby realizacji tematu oraz wykonująca badania, zrealizowana została przy wykorzystaniu języka `Python` w wersji `3.5`.

Głównym modułem, na którym oparto implementację, był `scikit-learn`. Wykorzystano go przede wszystkim do zbudowania sieci neuronowej uczoną metodą wstecznej propagacji błędu, a także skorzystano z wielu udostępnionych tam rozwiązań, jak na przykład walidacja krzyżowa i selekcja cech. Użyto modułu w wersji deweloperskiej `0.18.dev0`[1], gdyż dopiero od tej wersji można tam korzystać z sieci neuronowych.

Jako implementację drugiego badanego algorytmu - *Extreme Learning Machines*, posłużył moduł `Python-ELM`[2]. Wyznaczony on był na oficjalnej stronie *ELM*[3] jako jedna z bazowych implementacji tegoż algorytmu. Udostępniony kod nie był, niestety, przystosowany do działania pod wersją *Pythona 3*, na potrzeby projektu przerobiono więc kod tak, aby dało się go używać.

Wszystkie wykorzystane moduły i ich przeznaczenie widnieją na poniższej tabeli:

Tabela 2.1: Wykorzystane moduły.

Nazwa	**Wersja**	**Opis**
`scikit-learn`	0.18	sieć neuronowa uczona metodą wstecznej propagacji błędu, selekcja cech, walidacja krzyżowa, macierz błędu.
`Python-ELM`	0.3	*Extreme Learning Machines*.
`numpy`	1.11.0	Obliczenia numeryczne.
`matplotlib`	1.5.1	Tworzenie wykresów.

[1] http://scikit-learn.org/dev/documentation.html
[2] https://github.com/dclambert/Python-ELM
[3] http://www.ntu.edu.sg/home/egbhuang/elm_codes.html

2.2 Opis implementacji

Na aplikację składa się kilka modułów:

- `main.py` - główny moduł uruchamiający badania.
- `algorithms.py` - uruchamia oba algorytmy.
- `dataset.py` - importuje dane z pliku CSV.
- `grapher.py` - tworzy wykresy z wyników badań.
- `helper.py` - zawiera klasy pomocnicze do przechowywania danych.
- `consts.py` - zawiera ustawienia badań i parametry dla algorytmów.

2.3 Wdrożenie

W celu przygotowania aplikacji do działania, należy mając zainstalowaną wersję `Pythona 3.5` dodać katalog projektu do zmiennej środowiskowej `PYTHONPATH`. Aby tego dokonać, w katalogu projektu trzeba uruchomić komendę:

```
$ export PYTHONPATH="${PYTHONPATH}:${PWD}"
```

Kolejnym krokiem jest uruchomienie skryptu instalacyjnego `install.sh`, który to zainstaluje wymagane pakiety z pliku `requirements.txt` oraz pobierze poprawiony moduł `Python-ELM` do katalogu `modules`:

```
./install.sh
```

Teraz można uruchomić aplikację poprzez wywołanie skryptu głównego `main.py`:

```
python3 main.py
```

Działanie aplikacji powinno się zakończyć wygenerowanymi wykresami.

Rozdział 3

Badania

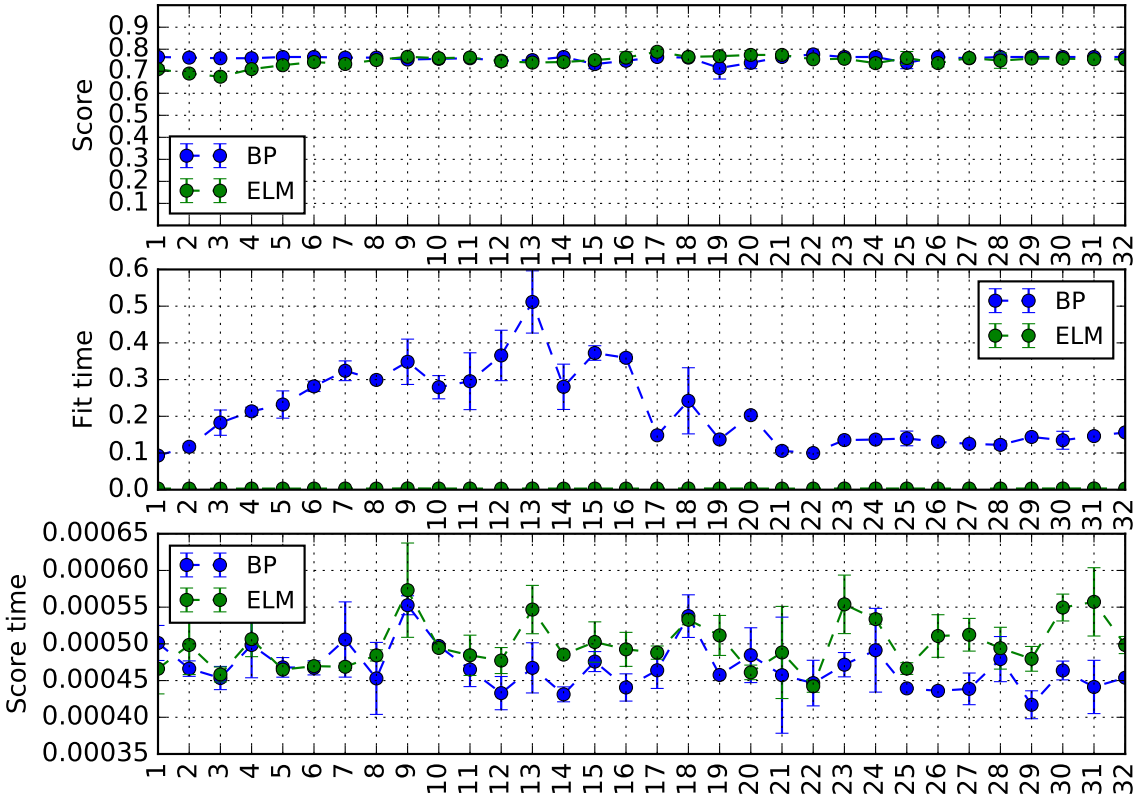
3.1 Przeprowadzone badania

Badania przeprowadzono dla 5 różnych liczb neuronów w warstwie ukrytej: 20, 30, 40, 50, 60.

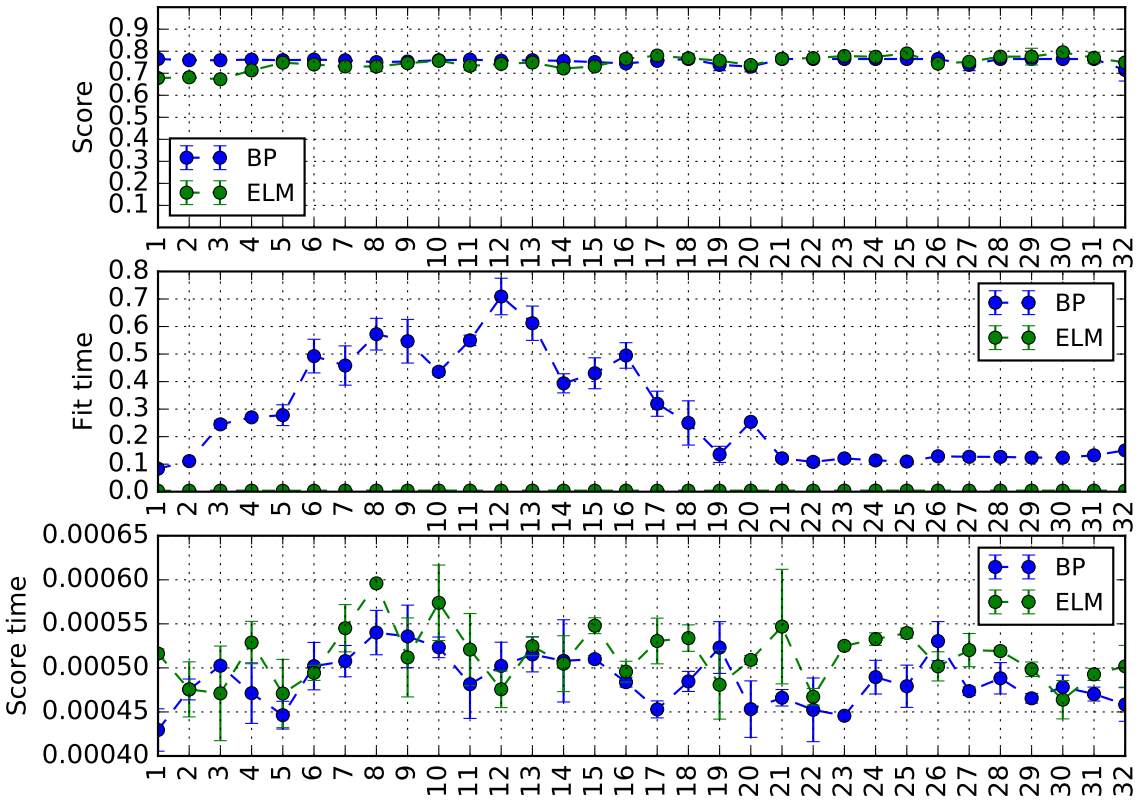
3.2 Porównanie algorytmów

3.3 Selekcja cech

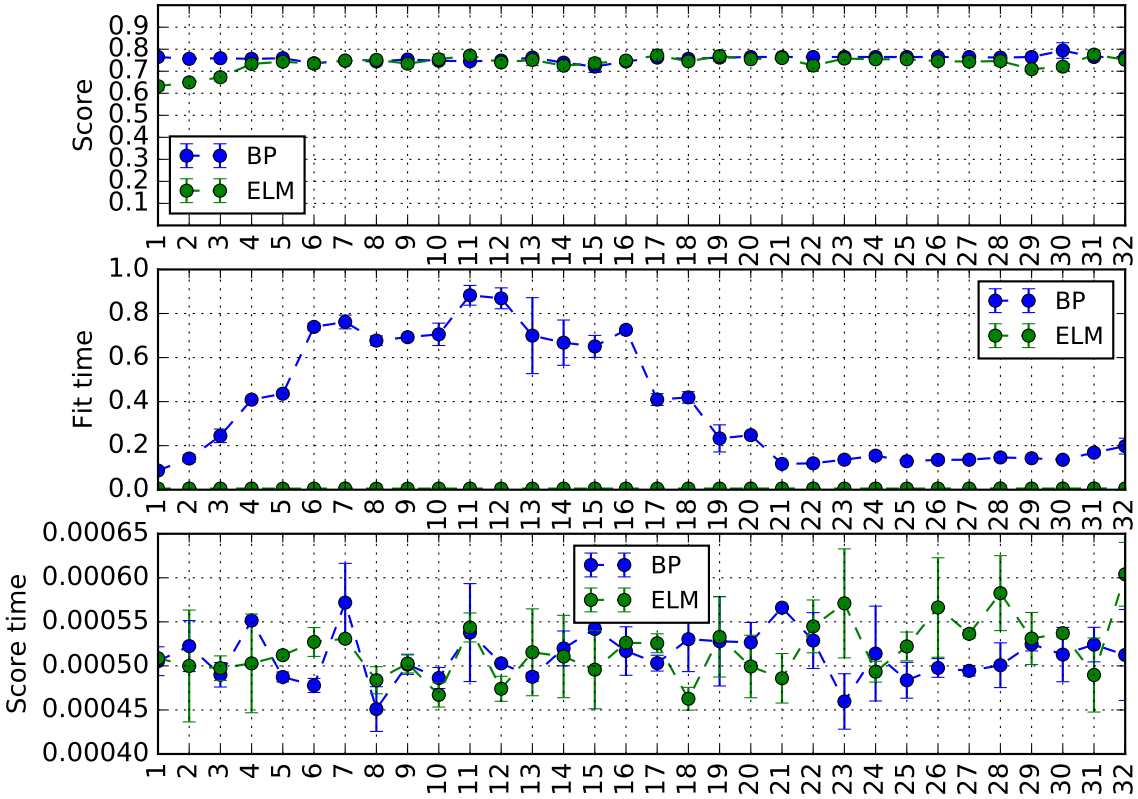
3.4 Macierz pomyłek



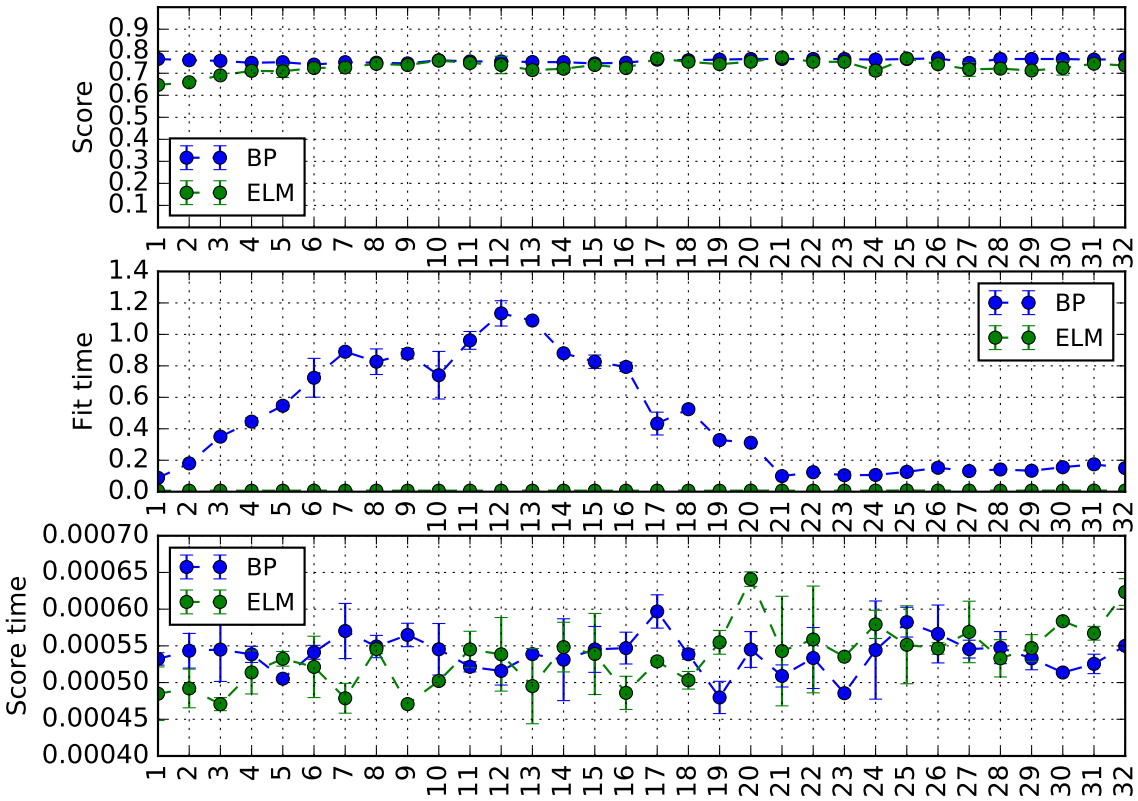
Rysunek 3.1: 20 neuronów w warstwie ukrytej - porównanie obu algorytmów



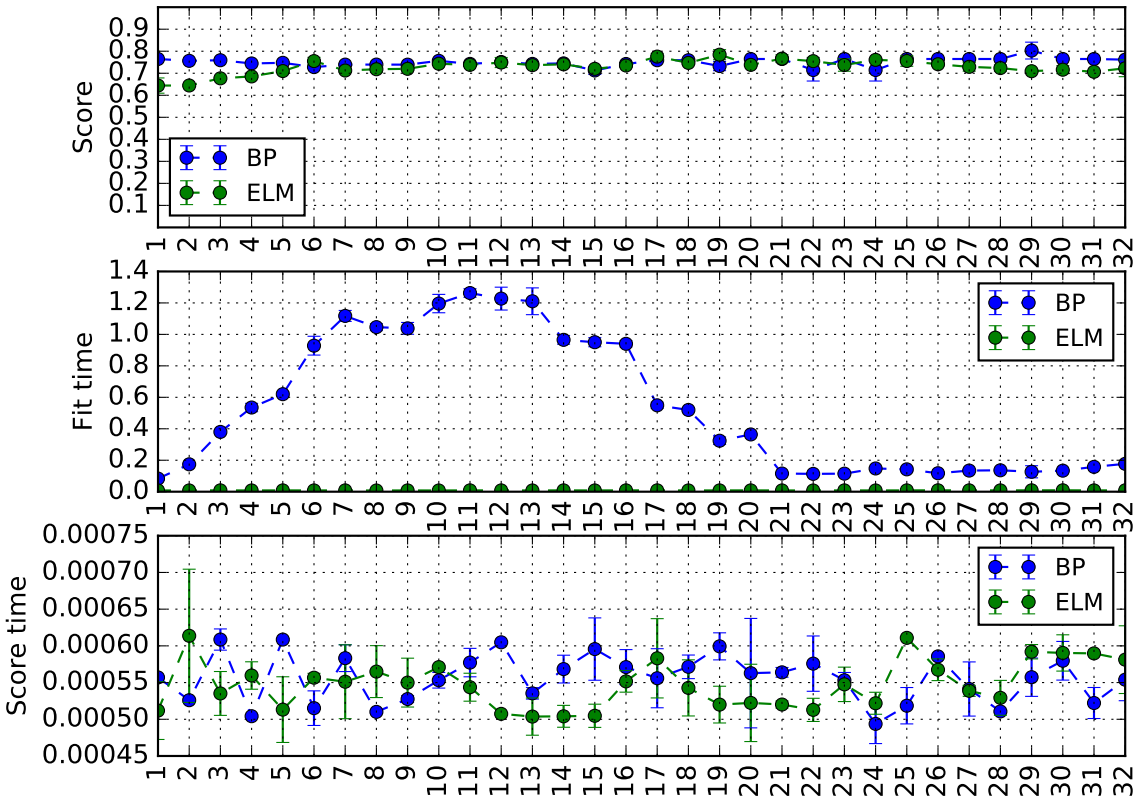
Rysunek 3.2: 30 neuronów w warstwie ukrytej - porównanie obu algorytmów



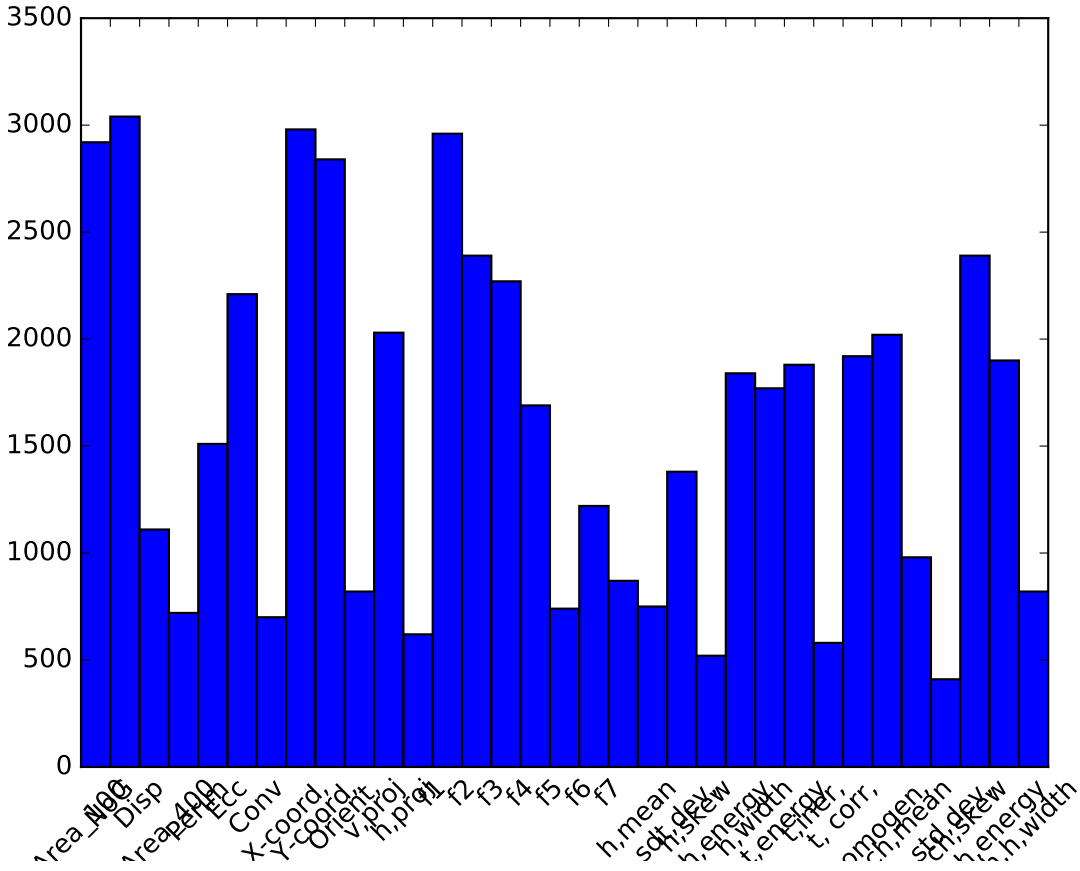
Rysunek 3.3: 40 neuronów w warstwie ukrytej - porównanie obu algorytmów



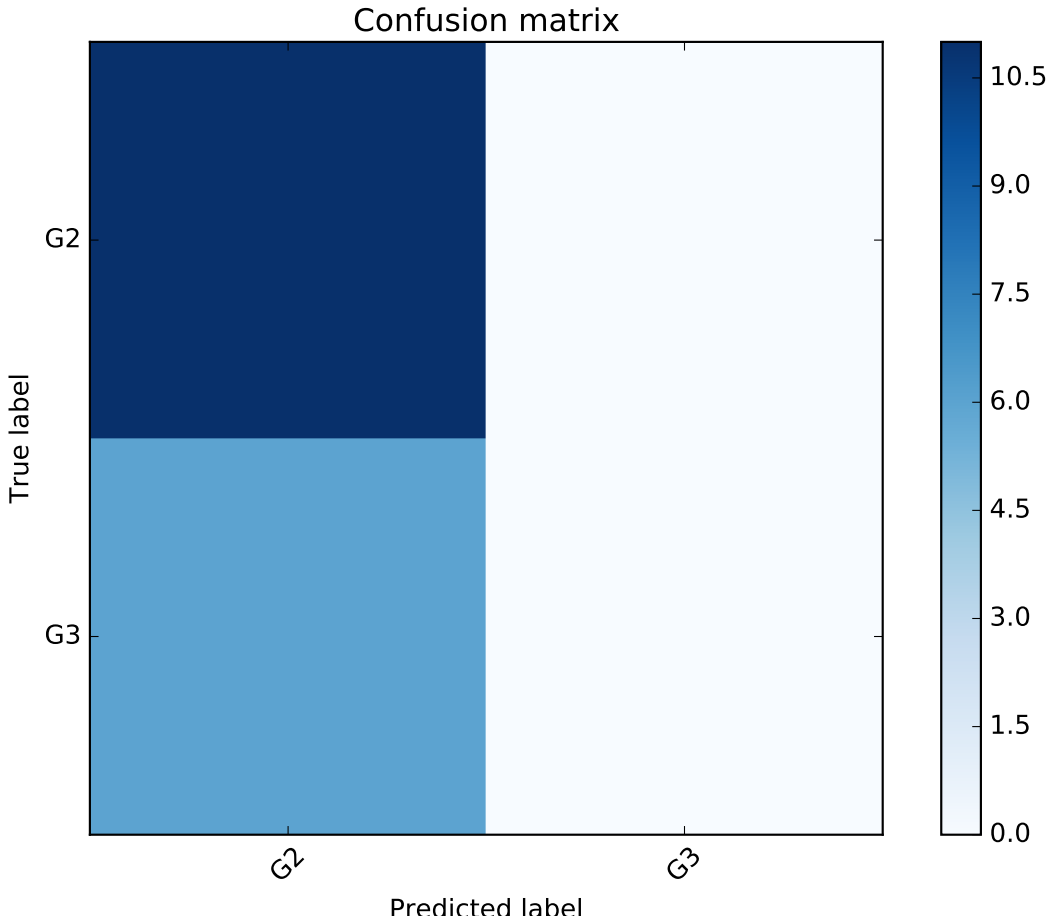
Rysunek 3.4: 50 neuronów w warstwie ukrytej - porównanie obu algorytmów



Rysunek 3.5: 60 neuronów w warstwie ukrytej - porównanie obu algorytmów



Rysunek 3.6: Selekcja cech po częstotliwości wybierania



Rysunek 3.7: Macierz pomyłek

Rozdział 4

Wnioski

- Sieci neuronowe potrzebują sporej ilości danych uczących, aby poprawnie generalizować problemy.
- ELM daje bardzo zbliżone jakościowo wyniki do sieci neuronowych ze wsteczną propagacją, jednocześnie proces uczenia jest o wiele szybszy.
- Wbrew początkowej intuicji, kilka najlepszych cech zebranych razem wcale nie musi dawać najlepszych wyników.
- Procesy selekcji cech oraz walidacji krzyżowej nie mogą być przeprowadzane osobno.