

POLITECHNIKA ŚLĄSKA W GLIWICACH

BIOLOGICALLY INSPIRED ARTIFICIAL INTELLIGENCE

Przewidywanie kursu walutowego

AUTORZY:

Forczmański Mateusz

Szukała Patryk

Informatyka, semestr VI
Rok akademicki 2014/2015
Grupa GKiO3

23 czerwca 2015

Spis treści

I	Temat projektu	2
1	Cel zadania	2
2	Analiza problemu	2
3	Wynik obliczeń	2
II	Specyfikacja zewnętrzna	3
1	Interfejs graficzny	3
1.1	Struktura sieci	3
1.2	Zdolność nauki	4
1.3	Inicjalizacja funkcji	4
1.4	Analiza plików	5
1.5	Trening sieci	5
1.6	Trening sieci	6
1.7	Graf błędu	6
2	Korzystanie z programu	7
III	Wykonanie i specyfikacja wewnętrzna	7
1	Dane testowe	7
1.1	Zbieranie	7
1.2	Format pliku	7
1.3	Parsowanie pliku	8
2	Obliczanie współczynnika RSI	9

Część I

Temat projektu

1 Cel zadania

Naszym zadaniem było przewidywanie kursu walutowego EUR/USD (euro / dolar) na rynku Forex. W założeniach naszego projektu mieliśmy wykonać je na sieciach neuronowych.

Idea przewidywania kursu walutowego polega na analizie danych historycznych kursu oraz trendu, jakim wykres kursu zmierza. Celem naszego programu jest takie wytrenowanie sieci neuronowej, aby sama mogła przewidzieć przyszły kursy. Wartości te są niezwykle zmienne w czasie i trudne do przewidzenia, a w profesjonalnym oprogramowaniu, gdy prawdopodobieństwo prawidłowego oszacowania przyszłego kursu przekracza 60% , jest to uznawane za sukces. Korzyścią takiego programu jest oczywiście ułatwienie podejmowania decyzji finansowych.

2 Analiza problemu

Wartości kursów na rynku Forex zmieniają się z dokładnością co do sekund (tzw. *tick*, który nie ma stałej wartości). By móc uprościć nasze zadanie zdecydowaliśmy się pobierać dane z dokładnością co do *dnia*. W tej jednostce można wyróżnić 4 wartości kursu:

- **Open** - wartość otwarcia danego dnia
- **Low** - minimalna wartość w ciągu dnia
- **High** - maksymalna wartość w ciągu dnia
- **Close** - wartość zamknięcia danego dnia

Te cztery wartości z każdego dnia stanowią nasze dane wejściowe. Nie są to również jedyne parametry, które określają wartość kursu. Zdecydowaliśmy się naszej sieci przekazywać również współczynnik **RSI** (*Relative Strength Index*), który określa siłę trendu. Oblicza się go w następujący sposób:

$$RSI = 1 - \frac{1}{1 + \frac{A_k}{B_k}}$$

gdzie:

- A_k - średnia wartość wzrostu cen zamknięcia z k dni
- B_k - średnia wartość spadku cen zamknięcia z k dni

3 Wynik obliczeń

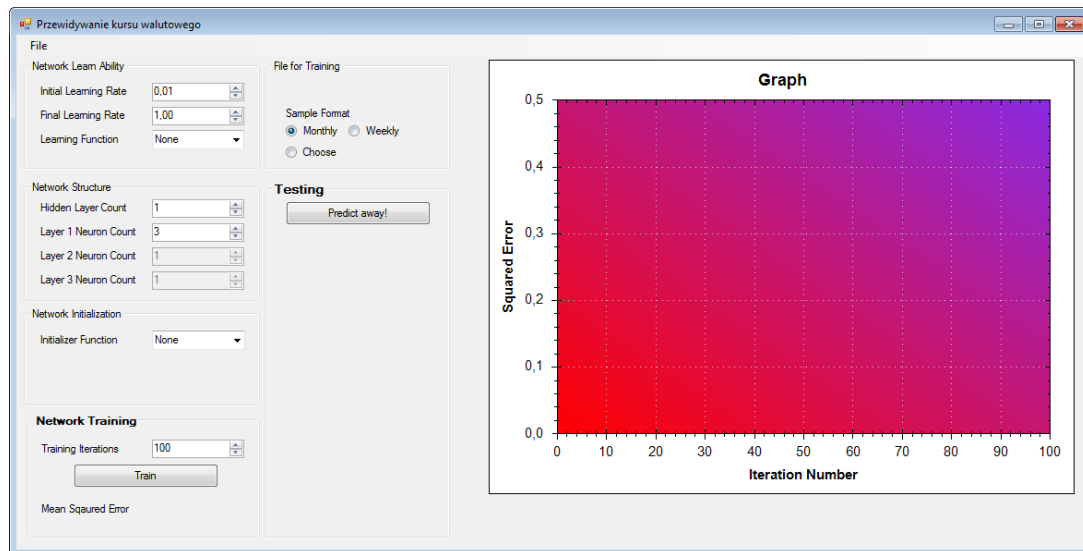
Nasza sieć decyduje czy wartość kursu zamknięcia następnego dnia będzie większa lub mniejsza od kursu zamknięcia dnia dzisiejszego. Zwrócona przez sieć wartość interpretowana jest następująco:

- 0 - wartość kursu będzie mniejsza
- 1 - wartość kursu będzie większa

Część II

Specyfikacja zewnętrzna

1 Interfejs graficzny



Rysunek 1: Interfejs po uruchomieniu programu

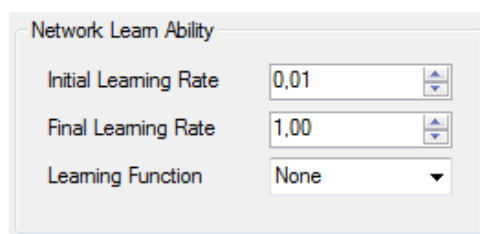
Przed przystąpieniem do trenowania sieci można ją skonfigurować:

1.1 Struktura sieci

Rysunek 2: Panel do konfiguracji struktury sieci z domyślnymi wartościami

W tym panelu definiuje się budowę wewnętrzną sieci neuronowej - liczbę warstw pośrednich oraz liczbę neuronów w każdej z warstw. Maksymalnie można utworzyć 3 warstwy z minimum 1 neuronem. W zależności od wartości w polu *Hidden Layer Count*, odpowiednie pola *Layer Neuron Count* są dostępne lub nie.

1.2 Zdolność nauki

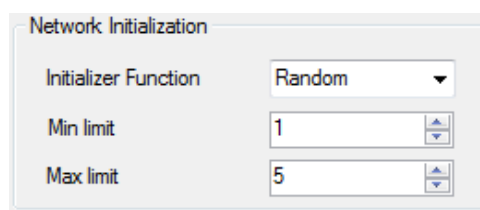


Rysunek 3: Panel do konfiguracji zdolności nauki z domyślnymi wartościami

Ten panel znajduje się w górnym lewym rogu interfejsu. Umożliwia konfigurację trzech parametrów:

- *Initial Learning Rate* - wartość startowa zdolności nauki, z nią uruchamiany jest trening sieci.
- *Final Learning Rate* - wartość zdolności nauki, którą sieć będzie miała, gdy trening się skończy.
- *Learning Function* - specjalizowana funkcja, jakiej sieć backpropagacji będzie wykorzystywać do nauki. Są cztery możliwości:
 - *None* - standardowa funkcja
 - *Exponential* - funkcja eksponencjalna
 - *Hyperbolic* - funkcja hiperboliczna
 - *Linear* - funkcja liniowa

1.3 Inicjalizacja funkcji

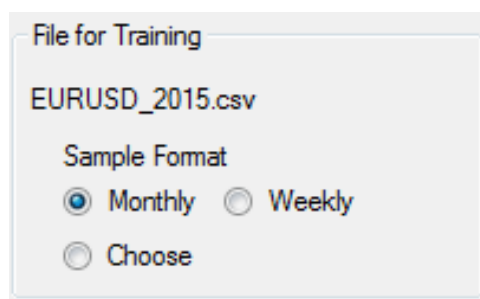


Rysunek 4: Panel do konfiguracji inicjalizacji sieci z przykładowymi ustawieniami

W tej części (lewa strona interfejsu) można ustawić inicjalizację sieci, czyli początkową wartość wag neuronów we wszystkich warstwach. Istnieje sześć możliwości:

- *None* - wartości domyślne.
- *Constant* - wskazana wartość stała.
- *NguyenWidrow* - sparametryzowana funkcja *NGuyen Widrow*.
- *NormalizedRandom* - znormalizowana wartość losowa.
- *Random* - wartość losowa ze wskazanego zakresu.
- *Zero* - funkcja typu Zero (wszystkie wartości są zbliżone do zera).

1.4 Analiza plików

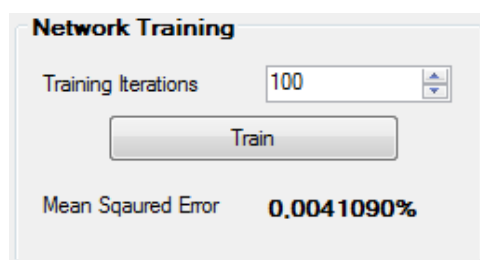


Rysunek 5: Panel z wczytanym plikiem i sposobem podziału

Ten panel znajduje się w górnej środkowej części interfejsu. Umożliwia pogląd załadowanego pliku do treningu sieci (tutaj *EURUSD_2015.csv*) oraz sposób w jaki dane będą grupowane:

- *Monthly* - co miesiąc.
- *Weekly* - co tydzień.
- *Choose* - wg wskazanej przez użytkownika liczby dni.

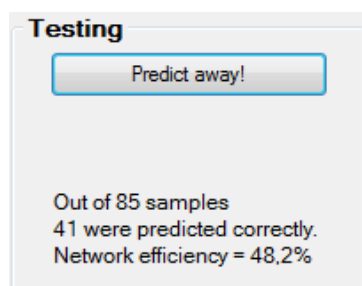
1.5 Trening sieci



Rysunek 6: Przykładowy wynik trenowania sieci po 100 iteracjach

Ten panel znajduje się w dolnym prawym rogu. Wymaga od użytkownika wprowadzania liczby iteracji treningowych, jakie wykona sieć (domyślnie jest to wartość 100). Po wciśnięciu przycisku *Train* sieć rozpocznie swój trening. Gdy zostanie on zakończony, na dole pojawia się *Mean Squared Error* (średni kwadratowy błąd) sieci w procentach. Oznacza on jakie wg obliczeń jest prawdopodobieństwo błędu sieci, czyli złego przewidzenia linii trendu kursu waluty. Program poświęca 70% próbek (zaokrąglonych w górę) do testów.

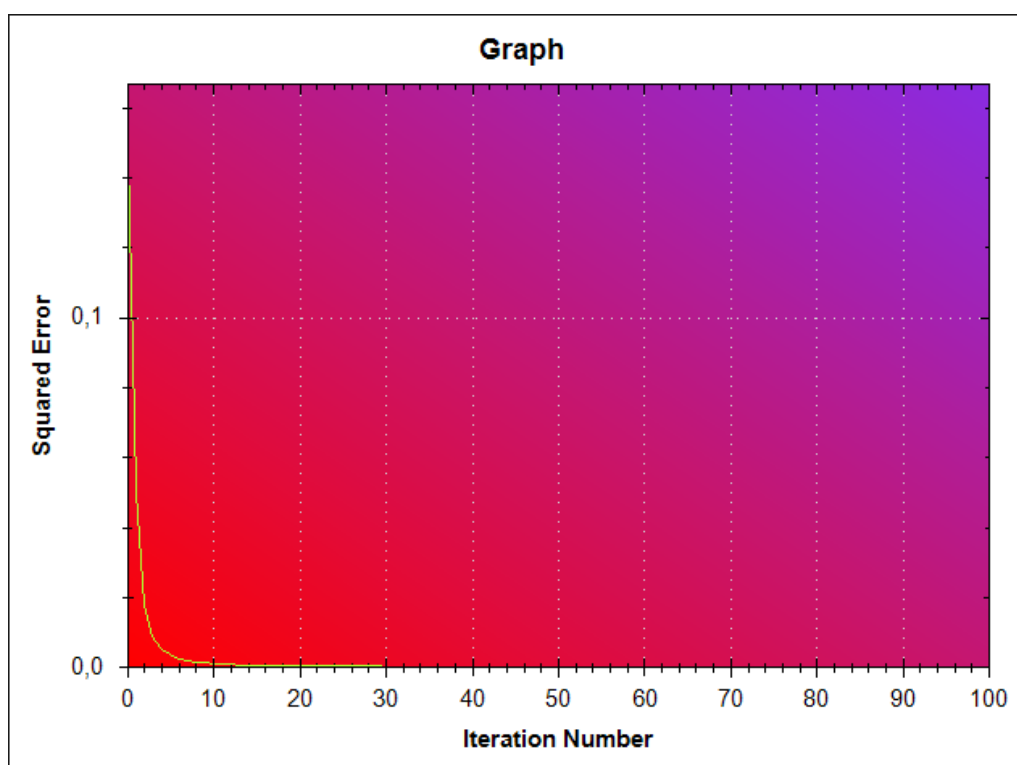
1.6 Trening sieci



Rysunek 7: Przykładowy wynik testowania sieci

Na środku interfejsu znajduje się przycisk *Predict away!*. Gdy sieć została wytrenowana i skonfigurowana, jego wciśnięcie powoduje uruchomienie testowania sieci i zmierzenie jej efektywności. Po wykonaniu testów informacje zostaną wyświetlone w tym samym panelu. Program poświęca 30% próbek w celu zbadania jakości działania sieci.

1.7 Graf błędu



Rysunek 8: Przykładowy przebieg nauki sieci

W czasie treningu sieci jest generowany graf zależności jakości sieci od liczby iteracji. Przedstawia on jak prawdopodobieństwo błędu zmieniało się wraz z kolejnymi iteracjami. Można powiedzieć, że obrazuje on, w jaki sposób sieć się uczy.

Maksymalną wartością na osi poziomej jest liczba iteracji, z kolei na osi pionowej jest to maksymalny średni błąd kwadratowy (wyliczany dynamicznie w trakcie działania programu).

2 Korzystanie z programu

Aby móc wytrenować sieć i badać jej efekty, wystarczy wykonać kilka prostych kroków:

1. Wczytać plik z danymi próbkowymi. Należy wybrać z górnego menu *File*→*Open* i wskazać odpowiedni plik CSV.
2. Ustawić wszystkie parametry zgodnie z instrukcjami w poprzednim podrozdziale.
3. Wybrać *Train* by wytrenować sieć.
4. Wybrać *Predict away!* by testować sieć.

Część III

Wykonanie i specyfikacja wewnętrzna

1 Dane testowe

1.1 Zbieranie

By móc przystąpić do działania, potrzebowaliśmy danych historycznych o kursie EUR/USD na rynku Forex. Istnieje dużo stron, które udostępniają te dane za odpowiednią opłatą, ale nie chcąc ani wydawać dziesiątek dolarów za nie, ani zakładać własnej firmy, szukaliśmy źródła, które udostępniłoby dane za darmo i przyjaznym formacie. Ostatecznie znaleźliśmy dwie strony internetowe, które nam pomogły:

- <http://www.global-view.com/forex-trading-tools/forex-history/> - strona za darmo udostępnia dane historyczne na temat kursu EUR/USD w formacie CSV. W generowanych przez nią plikach, z dokładnością do dnia, znajdują się kursy zamknięcia, najwyższy oraz najniższy w ciągu dnia. Prawie idealnie spełniało to nasze potrzeby.
- <http://pl.investing.com/currencies/eur-usd-historical-data> - strona za darmo wyświetla dane historyczne w postaci tabeli, której dane można kopiować. Mając dane z poprzedniej strony, wszystkie potrzebne nam wartości dziennych kursów otwarcia kopiowaliśmy ręcznie do pliku CSV.

1.2 Format pliku

Postanowiliśmy, żeby format pliku wejściowego był jak najprostszy, by dane mogły być szybko przetworzone, a jednocześnie wygodnie uzupełniane w miarę rozrastania się pliku. Zachowaliśmy plik CSV, jaki dostarczyła jedna ze stron, jest on wygodny oraz łatwy do sparsowania.

Przykładowy kawałek pliku:

```
2014-01-01,1.3745,1.3754,1.3744,1.38004
2014-01-02,1.3655,1.3775,1.3627,1.37629
2014-01-03,1.3587,1.3672,1.3583,1.36664
2014-01-06,1.3634,1.3653,1.3569,1.35908
2014-01-07,1.3614,1.366,1.3594,1.36289
2014-01-08,1.3581,1.3635,1.3551,1.36153
```

Data stanowi unikalny klucz dziennego kursu i nie może się powtarzać. W tym rodzaju pliku nagłówki są zbędne, więc aby móc łatwo zapamiętać, która wartość jest która, ułożyliśmy je *alfabetycznie*. Plik kolejno przedstawia:

Data, Close (zamknięcie), High (maksymalny), Low (minimalny), Open (otwarcie)

1.3 Parsowanie pliku

Ponieważ istnieje spora liczba gotowych bibliotek do parsowania plików CSV, zdecydowaliśmy się nie pisać własnej funkcji, tylko skorzystać z jednej z już istniejących. Nie szukaliśmy długo, nasz wybór padł na małą, ale bardzo wysoko ocenianą bibliotekę "CSV Parser" autorstwa Ideafixxxer (prawdziwe imię i nazwisko nieznane) ze strony Code Project¹.

Biblioteka przyjmuje na wejście plik CSV oraz zwraca strukturę typu `string[][]` z informacjami z pliku CSV w postaci tekstu. Format ten nie jest dogodny dla naszych potrzeb, dlatego napisaliśmy własny kod, który zmienia ową tablicę dwuwymiarową na mapę: kluczem jest data, a wartością tablica jednoelementowa wartości typu `double`. Całość została zakapsułkowana do klasy *ExchangeData*. Co prawda po formacie próbek wyraźnie widać, że wystarczyłby typ danych `float`, jednak nasza biblioteka przyjmuje jako parametry wartości `double`, więc postanowiliśmy poświęcić tych kilka kilobajtów by uniknąć konieczności wielokrotnego rzutowania.

```
class ExchangeData
{
    private const int RATE_NUMBER = 4;
    private readonly Dictionary<DateTime, Double[]> data;

    public ExchangeData(string[][] inputData)
    {
        data = new Dictionary<DateTime, Double[]>();
        for (int i = 0; i < inputData.GetLength(0); i++)
        {
            DateTime dt = ToDateTime(inputData[i][0]);
            Double[] rate = new Double[RATE_NUMBER];
            // Close
            rate[0] = Double.Parse(inputData[i][1],
                                   CultureInfo.InvariantCulture);
            // High
            rate[1] = Double.Parse(inputData[i][2],
                                   CultureInfo.InvariantCulture);
            // Low
            rate[2] = Double.Parse(inputData[i][3],
                                   CultureInfo.InvariantCulture);
            // Open
            rate[3] = Double.Parse(inputData[i][4],
                                   CultureInfo.InvariantCulture);
            this.data[dt] = rate;
        }
    }

    public static DateTime ToDateTime(string datetime)
    {
        string year = datetime.Substring(0, 4);
        string month = datetime.Substring(5, 2);
        string day = datetime.Substring(8, 2);
        return new DateTime(Convert.ToInt32(year), Convert.ToInt32(month),
                             Convert.ToInt32(day));
    }
};
```

¹Parser CSV: <http://www.codeproject.com/Tips/741941/CSV-Parser-Csharp>

2 Obliczanie współczynnika RSI

```
private double calculateRSI(List<Double[]> input, int start = 0)
{
    int periodNumber = input.Count - start;
    double alpha = 2.0d / (periodNumber + 1);
    double denominator = 0.0d;
    double avg_growth = 0.0d, avg_fall = 0.0d;
    List<double> growth = new List<double>();
    List<double> fall = new List<double>();
    // porównanie wartości Close pomiędzy kolejnymi dniami oraz obliczenie
    // spadku/wzrostu
    for (int i = start; i < periodNumber - 1; ++i)
    {
        double weight = Math.Pow(1 - alpha, periodNumber - i);
        double value = input[i][0];
        double value_next = input[i + 1][0];
        if ((value - value_next) >= 0)
            fall.Add((value - value_next) * weight);
        else
            fall.Add(0);
        if ((value_next - value) >= 0)
            growth.Add((value_next - value) * weight);
        else
            growth.Add(0);
        denominator += weight;
    }
    //obliczanie sredniego spadku/wzrostu
    avg_fall = fall.Sum() / denominator;
    avg_growth = growth.Sum() / denominator;
    //obliczenie RSI
    double RSI = 100 - (100 / (1 + (avg_growth / avg_fall)));
    return RSI;
}
```

Metoda zwracająca współczynnik RSI (patrz: Analiza Problemu 2). W celu obliczenia średniego wzrostu i spadku kursu wykorzystano pętlę, która iteruje po podanej liście zawierającej dane historyczne kursów (parametr *input*). Zależnie od tego czy różnica między wartością Close aktualnego dnia, a dnia następnego jest dodatnia czy ujemna, następuje dodanie tej różnicy do odpowiedniej listy (spadków / wzrostów). W przypadku wartości spadku różnica obliczana jest między dniem następnym, a aktualnym, po to aby wartości w obu listach były dodatnie.

W ciele funkcji odnosimy się do elementu zerowego tablicy, ponieważ właśnie tam znajdują się kursy zamknięcia (patrz: format pliku 1.2).

Po wyjściu z pętli następuje obliczenie średnich z wartości znajdujących się w listach, a następnie podstawienie tych wartości do wzoru (patrz: Analiza Problemu 2) i zwrócenie jej przez metodę.