Heurystyczne Metody Optymalizacji
Informatyka, sem V
Dokumentacja projektu Unity Neural Network Car

Artur Bednarczyk, Dawid Grajewski, grupa A

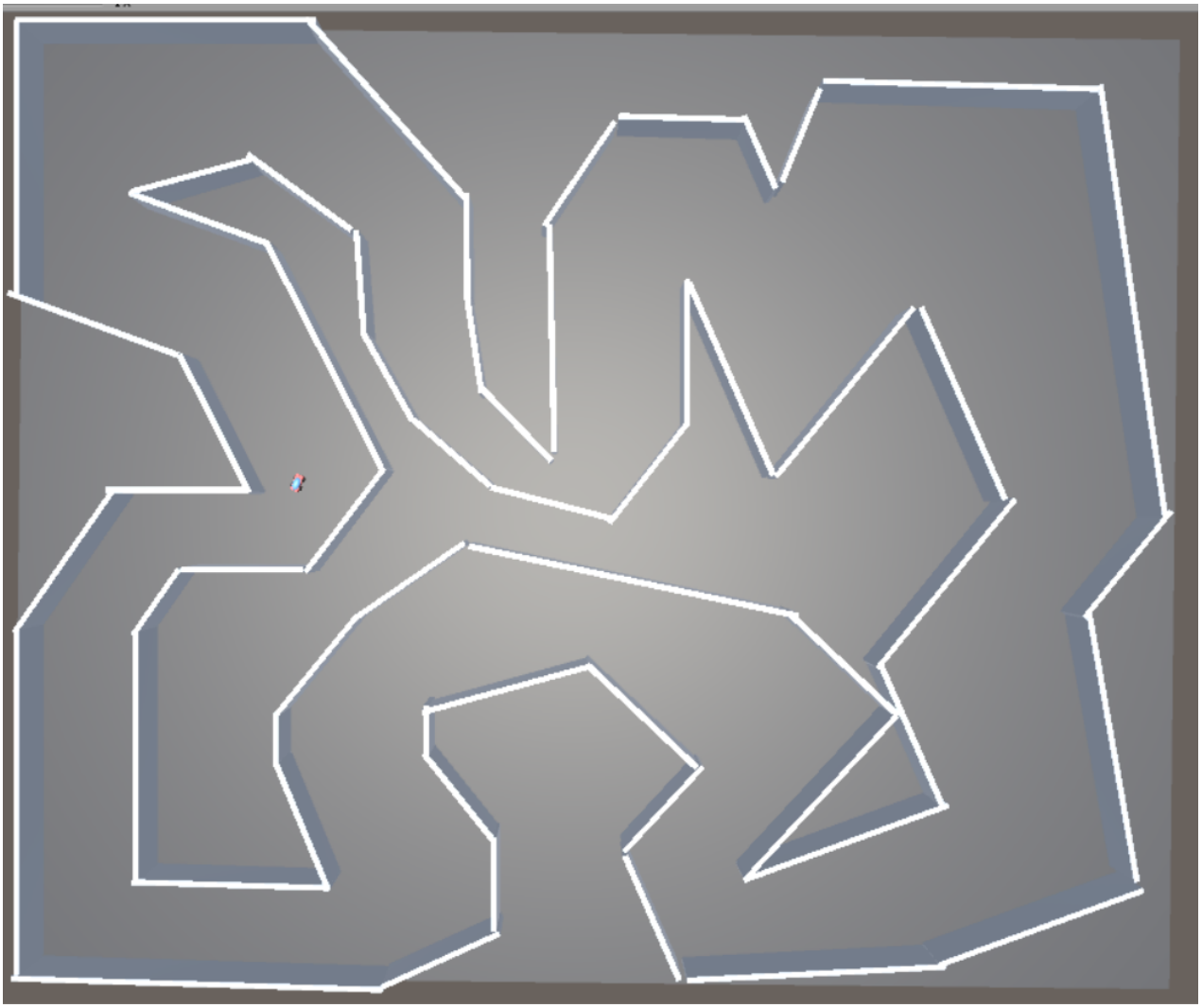
3 stycznia 2019

Spis treści

1 Część I

1.1 Opis programu

Unity Neural Network Car to symulacja samochodu poruszającego się po torze. Jednak nie tylko użytkownik może kontrolować pojazd, a również sztuczna inteligencja. Implementacja zaawansowanej technologii pozwoliła na „nauczenie" samochodu w jaki sposób przejechać przez cały tor i nie rozbić się na najbliższym zakręcie.



1.2 Instrukcja obsługi

Uruchamiamy aplikację i możemy obserwować ruch pojazdu.

1.3 Dodatkowe informacje

Projekt został wykonany z wykorzystaniem silnika Unity oraz języka C#.

2 Część II

2.1 Opis działania

2.1.1 Sieć neuronowa

Sieć neuronowa jest strukturą inspirowaną budową naturalnych neuronów, łączących je synaps oraz układów nerwowych. Wykorzystana w projekcie sieć jest wielowarstwową siecią jednokierunkową korzystającą z algorytmu propagacji wstecznej.

2.1.2 Budowa sieci

Wykorzystana sieć składa się z trzech głównych warstw.

- Warstwa wejścia.
- Warstwy ukryte.
- Warstwa wyjścia.

Warstwa wejścia Każdy neuron tej warstwy przekazuje do warstwy ukrytej początkowe wartości.

Warstwy ukryte Tutaj dzieje się wszystko co najważniejsze. Synapsy między neuronami mają przypisane wagi, które początkowo są losowe. W ramach uczenia się, wagi są dostosowywane tak, aby rezultat końcowy był jak najbliżej spodziewanego.

Warstwa wyjścia To tutaj nauczona już sieć daje nam wynik.

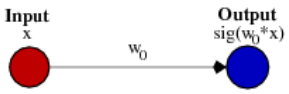
2.1.3 Uczenie sieci

Uczenie sieci jest realizowane poprzez podanie jej zestawu danych wejściowych oraz spodziewanych wyników. W tym projekcie wykorzystano metody takie jak:

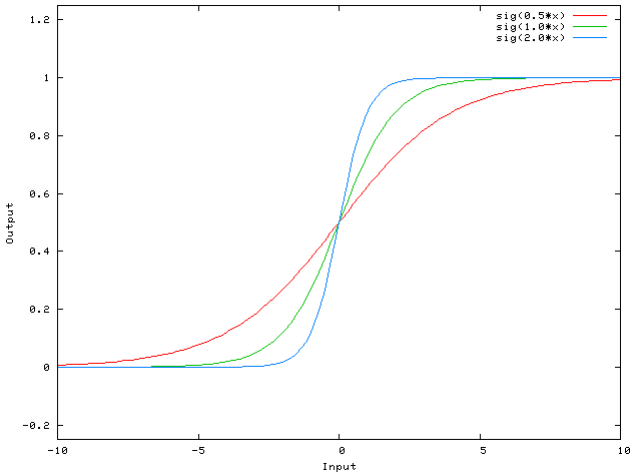
- Propagacja Wsteczna
- Biases
- Momentum
- Współczynnik uczenia

Propagacja wsteczna to podstawowy algorytm uczenia nadzorowanego wielowarstwowych, jednokierunkowych sieci neuronowych. Dzięki wstecznej propagacji błędu możemy dobrać odpowiednie wagi we wszystkich warstwach sieci. Początkowo dla losowych wag obliczane są błędy, czyli różnica między odpowiedzią obliczoną a spodziewaną, które następnie są propagowane do wcześniejszych warstw. Wagi zostają modyfikowane na podstawie błędu i obliczonych danych. Algorytm jest zostaje zatrzymany, gdy średnia wartość błędu przestanie maleć.

Biases Są to wartości, które pozwalają na przesunięcie funkcji aktywacji w sposób, na który zmiana wagi nie pozwala. Przykład funkcji bez bias:

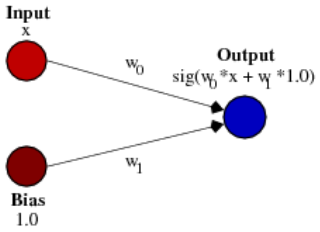


Rysunek 1: https://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks

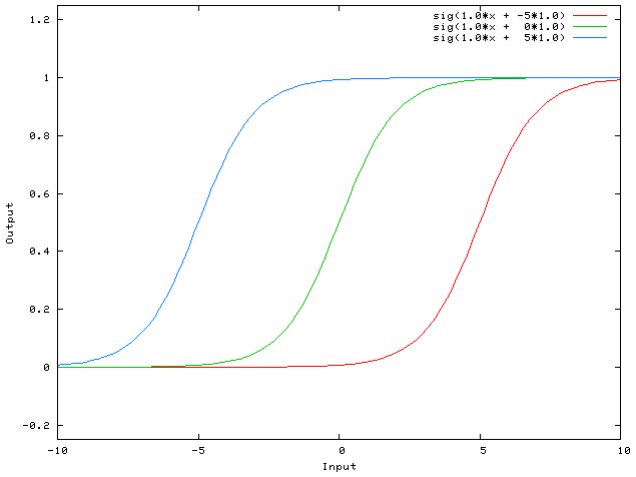


Rysunek 2: https://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks

Przykład funkcji z bias:



Rysunek 3: https://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks



Rysunek 4: https://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks

Wykorzystanie tego jest konieczne aby uczenie zostało zrealizowane z prawidłowymi wynikami.

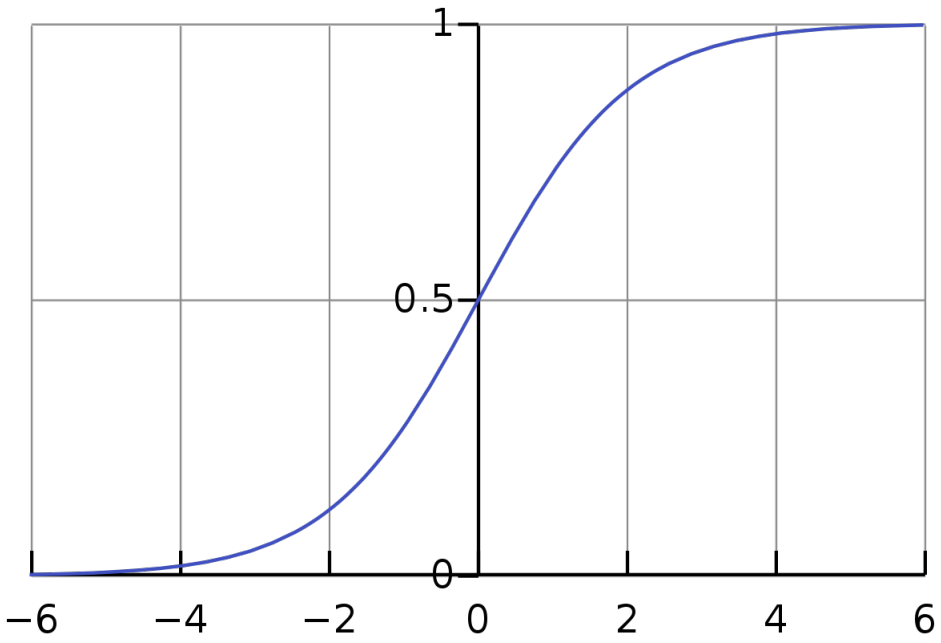
Momentum Jest to współczynnik odpowiadający za dodanie ułamka poprzedniej wagi do aktualnej. Używany jest aby zapobiec zbieżności do lokalnego minimum lub punktu siodłowego. Wysoka wartość tego współczynnika przyspiesza proces konwergencji, jednakże zbyt wysoka może spowodować, że cały system stanie się niestabilny, a zbyt niska spowolni proces uczenia.

Współczynnik uczenia to współczynnik odpowiadający za to jak duże są zmiany wag oraz biasów.

2.1.4 Funkcja Aktywacji

Wartość wyjścia neuronów jest obliczana za pomocą funkcji aktywacji. W tym projekcie została użyta funkcja sigmoidalna:

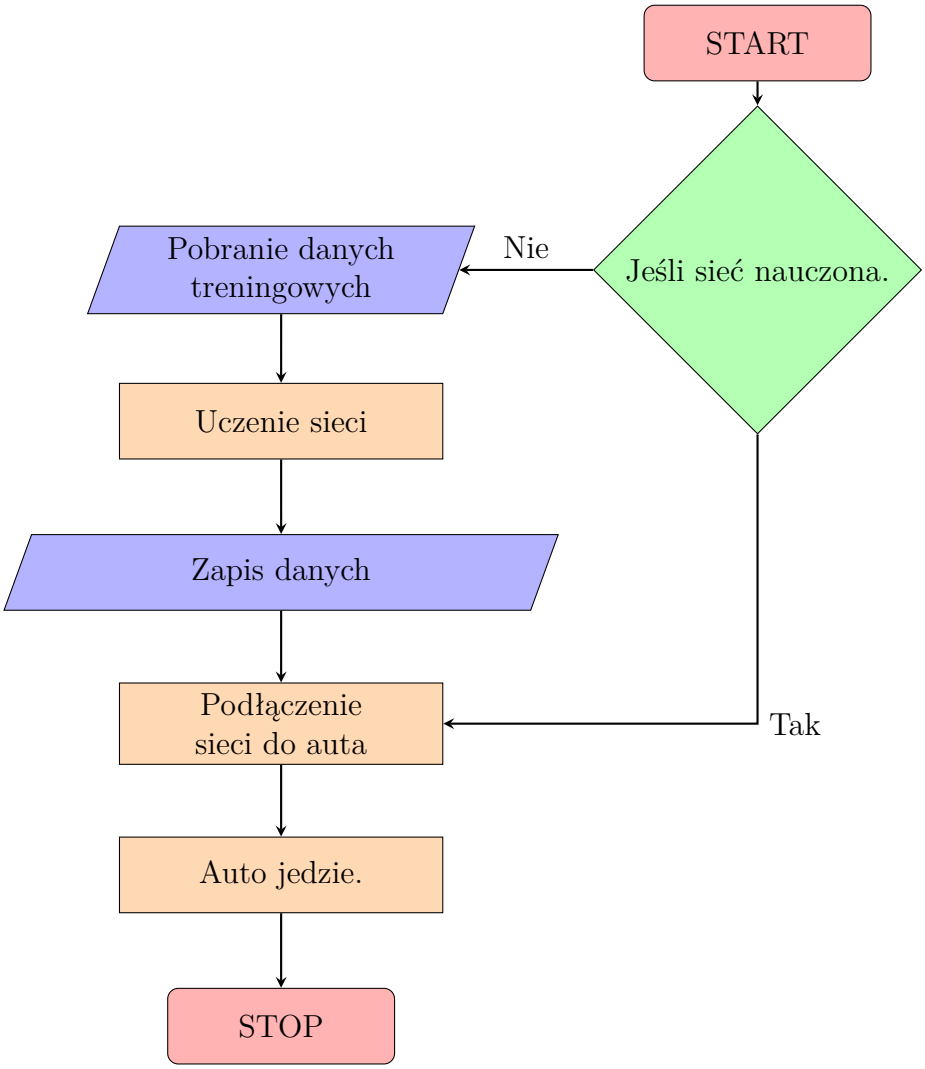
$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$



Rysunek 5: https://en.wikipedia.org/wiki/Sigmoid_function

Funkcja ta przyjmuje wartości od 0 do 1.

2.2 Algorytm


START
Jeśli sieć nauczona.
Nie
Pobranie danych treningowych
Uczenie sieci
Zapis danych
Podłączenie sieci do auta
Tak
Auto jedzie.
STOP

2.3 Bazy danych

Zestaw danych wykorzystany do uczenia sieci:

```
14.85;14.85;0
15.2;14.53;-0.58
13.32;17.02;1
14.13;15.67;0.91
15.17;14.55;-0.55
11.15;17.01;1
11.61;13.86;0.98
12;12.78;0.65
12.11;12.54;0.41
12.27;12.21;-0.06
9.19;10.16;0.75
11.7;11.57;-0.13
21.19;30.54;1
11.52;13.08;0.92
15.13;14.7;-0.41
13.91;5.75;-1
27.65;7.57;-1
25.97;28.01;0.97
2.45;34.22;1
6.81;25.24;1
16.07;9.18;-1
8.77;28.54;1
53.54;16.19;-1
14.85;14.85;0
11.91;20.92;1
15.48;13.44;-0.97
11.55;42.66;1
9.13;57.46;1
11.39;8.89;-0.99
8.77;17.61;1
6.76;11.94;1
9.29;17.88;1
11.99;14.92;0.99
14.62;23.18;1
9.26;43.26;1
11.42;11.28;-0.14
27.01;15.2;-1
46.28;12.89;-1
9.78;42.4;1
9.7;34.88;1
16.85;29.69;1
13.93;32.74;1
11.93;22.02;1
11.85;12.9;0.78
11.65;13.09;0.89
11.47;13.27;0.95
11.28;36.85;1
16.38;59.26;1
13.28;17.18;1
21.74;9.69;-1
14.12;15.83;0.94
8.96;46.62;1
10.84;25.92;1
10.43;14.23;1
11.96;12.46;0.46
8.41;56.14;1
9.51;9.86;0.34
9.58;9.8;0.22
8.58;34.3;1
11.4;28.29;1
12.04;18.61;1
9.9;12.58;0.99
11.4;12.29;0.71
21.91;14.5;-1
27.92;10.77;-1
10.56;41.72;1
12.83;34.31;1
22.76;12.84;-1
14.72;41.8;1
10.84;17.78;1
13.08;11.69;-0.88
11.38;13.75;0.98
12.49;44.46;1
18.94;62.05;1
14.82;15.15;0.32
15.99;13.69;-0.98
14.75;14.95;0.2
14.61;15.09;0.45
13.22;25.17;1
8.73;44.85;1
12.02;13.28;0.85
8.11;18.57;1
9.59;9.82;0.23
8;11.66;1
9.77;32.76;1
12.85;12.84;-0.01
17.03;20.62;1
10.1;13.55;1
29.72;11.01;-1
23.51;24.2;0.6
10.39;14.55;1
9.35;38.1;1
14.44;31.04;1
17.75;45.91;1
15.47;8.69;-1
10.89;21.47;1
11.22;13.82;0.99
11.57;13.18;0.92
13.29;40.47;1
16.3;27.81;1
14.79;14.91;0.12
14.79;14.91;0.12
17.33;13.24;-1
14.83;14.86;0.03
14.83;14.86;0.03
15.97;13.94;-0.97
15.97;13.94;-0.97
14.82;14.87;0.05
13.13;17.46;1
14.07;15.7;0.93
13.51;16.47;0.99
14.15;15.53;0.88
12.79;17.21;1
12.02;48.39;1
10.48;16.74;1
10.48;16.74;1
11.78;12.9;0.81
11.78;12.9;0.81
14.1;10.7;-1
14.1;10.7;-1
12.61;11.8;-0.67
12.61;11.8;-0.67
12.6;11.81;-0.66
10.3;53.25;1
9.32;56.21;1
11.54;7.91;-1
10.98;8.66;-0.98
17.35;6.73;-1
```

2.4 Implementacja

Projekt powstał z wykorzystaniem narzędzi Unity. Funkcjonalność została zawarta w folderze Assets/Scripts, którego zawartość wygląda następująco:

- Controllers
  - AutonomicCarController.cs
  - BrainController.cs
  - CarController.cs
  - ManualCarController.cs
- NeuralNetwork
  - Helpers
    - ExportHelper.cs
    - HelperNetwork.cs
    - ImportHelper.cs
  - NetworkModels
    - Dataset.cs
    - Network.cs
    - Neuron.cs
    - Sigmoid.cs
    - Synapse.cs
- Serializers
  - ISerializer.cs
  - XmlSerializer.cs

Klasy i opis ich metod:

- AutonomicCarController.cs
  - Update() - Wywołuje metodę sprawdzającą dystans do ścian, metodę ruchu i metodę sterowania.
- BrainController.cs
  - Compute(double,double) - Pobiera z sieci wynik na podstawie odległości od ścian.
  - Load() - Odtworzenie sieci neuronowej z wskazanego pliku.
  - Save() - Zapis instancji sieci neuronowej do wskazanego pliku.
  - TestNetwork() - Testuje sieć.
  - Train() - Uczy sieć na podstawie danych z wskazanego pliku.

- CarController.cs
  - MoveForward() - Ruch do przodu.
  - TurnLeft() - Skręt w lewo.
  - TurnRight() - Skręt w prawo.
  - Update() - Wywołuje metodę sprawdzającą dystans do ścian.
- ManualCarController.cs
  - Update() - Wywołuje metodę sprawdzającą dystans do ścian oraz jeśli zbiera dane to wywołuje metodę zbierającą dane do uczenia.
- ExportHelper.cs
  - ExportNetwork(Network, string, ISerializer) - wywołuje metodę pobierającą sieć i eksportuje ją do pliku.
- HelperNetwork.cs
  - HelperNetwork() - Konstruktor tworzący listy neuronów dla warstwy wejściowej, wyjściowej, listę list neuronów warstw ukrytych oraz listę synaps.
- ImportHelper.cs
  - ImportNetwork(string, ISerializer) - Importuje sieć z pliku.
- Dataset.cs
  - DataSet(double[], double[]) - Konstruktor klasy DataSet.
- Network.cs
  - Compute(double[]) - Wywołuje metodę przedniej propagacji i pobiera wyniki z warstwy wyjściowej.
  - GetRandom() - Zwraca losową liczbę.
  - Network() - Konstruktor.
  - Network(int,int[],int,double?,double?,bool) - Konstruktor.
  - Train(List<DataSet>,double) - Wywołuje metody propagacji aż średni błąd będzie mniejszy niż podano.
  - Train(List<DataSet>,int) - Wywołuje metody propagacji aż zostanie osiągnięta określona liczba powtórzeń.
- Neuron.cs
  - CalculateError(double) - Oblicza błąd
  - CalculateGradient(double?) - Oblicza gradient.
  - CalculateValue() - Oblicza wartość.

  - Neuron() - Konstruktor.
  - Neuron(IEnumerable<Neuron>) - Tworzy połączenia synaps między neuronami.
  - UpdateWeights(double,double) - Aktualizuje wagi.

- Sigmoid.cs
  - Output(double) - Zwraca wynik funkcji w zależności od podanej wartości.
  - Derivative(double) - zwraca pochodną funkcji.

- Synapse.cs
  - Synapse() - Konstruktor.
  - Synapse(Neuron,Neuron) - Konstruktor z wejściowym i wyjściowym neuronem.

- XmlSerializer.cs
  - Deserialize(string) - Odczytuje sieć z pliku.
  - Serialize(HelperNetwork, string) - Zapisuje sieć do pliku.

2.5 Testy

Sieć działa prawidłowo. Samochód porusza się po torze w sposób zadowalający.

Wykres błędów :