

Raport z projektu indywidualnego  
Vowpal Wabbit

*Dawid Wietrzych i Jacek Krakowski*

# Vowpal Wabbit

## 1. Implementacja

Do zaimplementowania uczenia maszynowego w naszej symulacji użyliśmy systemu wspieranego przez firmę Microsoft jakim jest Vowpal Wabbit. Na naszych maszynach zainstalowany został cały pakiet VW, a z algorytmu może korzystać każda inna maszyna, która ma zainstalowany wyżej wymieniony pakiet.

## 2. Model – Input - Output

Naszymi danymi wejściowymi do uczenia był plik tekstowy wygenerowany przez funkcję **get\_data\_for\_vowpal.py**, gdzie po każdym przejściu mapy przez śmieciarkę, do pliku zostały dopisywane kolejne przejścia w wymaganym przez VW formacie stringów. Do trenowania zostało użyte 5955 przykładów uczących. Każda linia składa się z tzw. „Labela” (liczby z zakresu od 0 do 4),

1   5 5 6 8 7 7 7 6	} Przykład pojedynczych predykcji
3   7 7 6 6 5 7 7 7	

określającego w naszym przypadku przewidywany ruch dla danej sytuacji w jakiej znajduje się agent. To znaczy odpowiednio przesunięcie śmieciarki w górę („0”), w lewo („1”), w dół („2”), lub prawo („3”), a także bez ruchu (wymuszone przez strukturę naszych algorytmów przechodzenia). Do uzyskania tych wyników posłużyliśmy się algorytmami przechodzenia w głąb i wszerz (*Depth First Search* i *Breadth First Search*). Dzięki temu algorytm, za każdym przesunięciem się agenta sprawdzał wszystkie elementy dookoła i odpowiednio przypisywał wartości obiektom (konwertował je na format VW) tj:

- 5 - droga
- 6 – domek
- 7 – wszystko inne (m.in. trawa)

Wersja naszego modelu została wytrenowana na 15 przejściach przez dane wejściowe. Dzięki temu otrzymaliśmy plik cache, który uwydatniał pracę algorytmowi uczenia.

Zostały też użyte dodatkowe flagi to trenowania jak:

- `-c` – używanie plików cache, do szybszego działania algorytmu
- `-passes 15` – uwzględnienie liczby trenowań

Jak i już w części tworzącej model:

- `-t` – nieskupianie się na labelach, lecz na części w której znajdują się poszczególne obiekty (co pozwala zmniejszyć zużycie pamięci)
- `-quiet` - wyłącza normalny wydruk diagnostyczny aktualizacji postępu.

Dance wyjściowe są swoistym przewidywaniem najlepszej drogi dla naszej śmieciarki (według algorytmów VW), tak aby odwiedziła wszystkie domy w możliwie jak najkrótszym czasie, bez konieczności powtarzania dróg (znalezienie optymalnej ścieżki Hamiltona).

### 3. Uruchomienie

Po uruchomieniu środowiska z modelem uczącym śmieciarka porusza się odpowiednio po wyznaczonych drogach i zatrzymuje się przy wybranych domkach. Dane wszystkich statystyk po uruchomieniu wyglądają następująco:

```
final_regressor = ../Data/vowpal.model
Num weight bits = 18
learning rate = 0.5
initial_t = 0
power_t = 0.5
decay_learning_rate = 1
using cache_file = ../Data/vowpal-data.txt.cache
ignoring text input in favor of cache input
num sources = 1
```

average loss	since last	example counter	example weight	current label	current predict	current features
1.000000	1.000000	1	1.0	1.0000	0.0000	9
3.655604	6.311208	2	2.0	4.0000	1.4878	9
3.466136	3.276669	4	4.0	1.0000	1.8501	9
3.823006	4.179875	8	8.0	4.0000	1.4175	9
3.103831	2.384656	16	16.0	2.0000	2.1663	9
2.376020	1.648209	32	32.0	1.0000	1.8924	9
2.433972	2.491925	64	64.0	3.0000	2.3930	9
1.991000	1.548027	128	128.0	0.0000	2.5055	9
1.797653	1.604307	256	256.0	2.0000	2.4506	9
1.663720	1.529787	512	512.0	4.0000	3.2069	9
1.566940	1.470159	1024	1024.0	1.0000	1.7464	9
1.543921	1.520902	2048	2048.0	4.0000	2.1040	9
1.493835	1.443749	4096	4096.0	4.0000	2.4284	9
1.459436	1.459436	8192	8192.0	3.0000	2.7216	9 h
1.474756	1.490076	16384	16384.0	1.0000	1.9235	9 h

```
finished run
number of examples per pass = 5360
passes used = 4
weighted example sum = 21440.000000
weighted label sum = 47164.000000
average loss = 1.455099 h
best constant = 2.199813
total feature number = 192960
```

