

Deep reinforcement learning

Michal CHOVANEC, PhD.

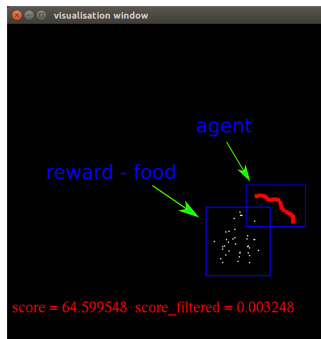
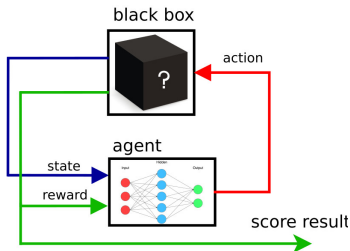
May 2018

Faculty of Management Science and Informatics

Problem definition

- learn to play game with unknown rules
- input : state and reward
- output : action and total score
- $Q(s, a)$: learn Q function

agent never sees required value (required action)



Q-learning algorithm

$$Q'(s, a) = R(s, a) + \gamma \max_{a' \in A} Q(s', a')$$

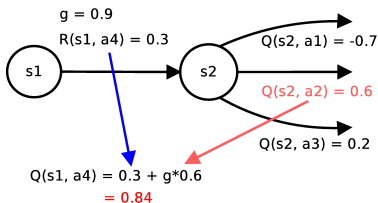
where

$Q(s, a)$ is previous state

$Q(s', a')$ is actual state

$R(s, a)$ is reward obtained in state s after executing action a

γ is discount factor $\gamma \in \langle 0, 1 \rangle$



SARSA algorithm

State Action Reward State Action

$$Q'(s, a) = (1 - \alpha)Q(s, a) + \alpha(R(s, a) + \gamma Q(s', a'))$$

where

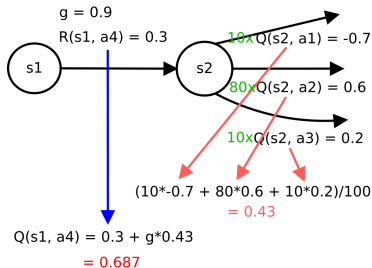
$Q(s, a)$ is previous state

$Q(s', a')$ is actual state

$R(s, a)$ is reward obtained in state s after executing action a

γ is discount factor $\gamma \in \langle 0, 1 \rangle$

α is learning rate $\alpha \in (0, 1)$



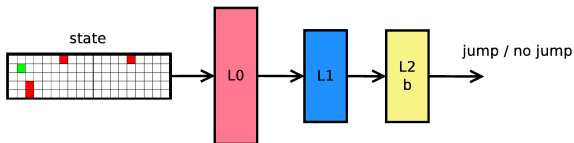
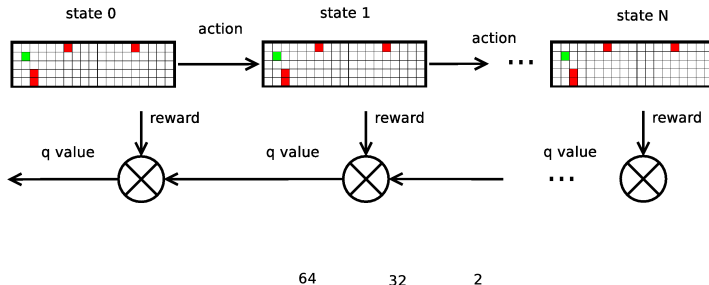
Storing Q values

- table
- linear combination of basis function (handmade features)
- Kenerva's sparse encoding
- neural network

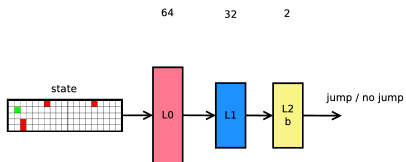
problems

- state correlations
- nonstationary Q values
- convergence to optimal strategy

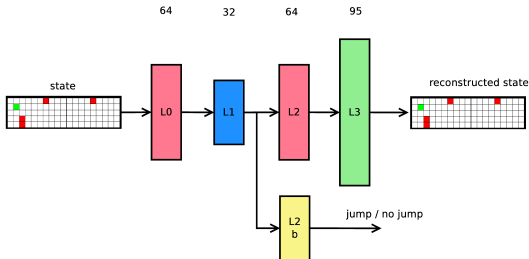
Neural network approximator - deep reinforcement learning



Speed up learning



common feed forward neural network



stacked autoencoder + feed forward neural network

Sparse weights - less computing

$$\Delta w = \eta E_x \frac{df(y)}{dw} - \lambda \text{sgn}(w)$$

where

E is error,

x is input,

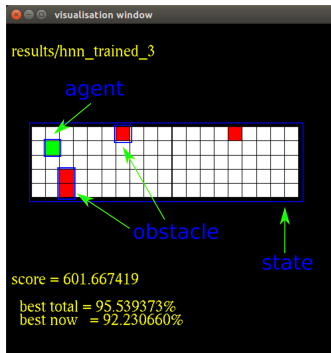
y is output,

f is activation function (ReLU, tanh, softmax ...),

η is learning rate ,

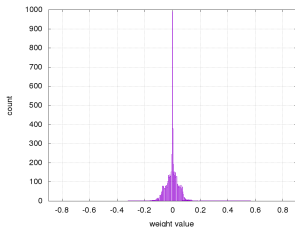
λ is sparsity parameter

Arcade game experiment

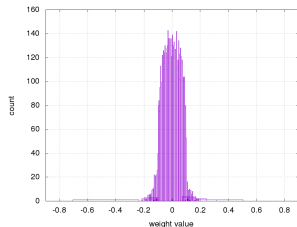


	FNN sparse	FNN no sparse	AE+FNN sparse	AE+FNN no sparse
unsupervised iterations	0	0	100000	100000
supervised iterations	200000	200000	200000	200000
iterations per slice	0	0	50000	50000
learning rate	0.0005	0.0005	0.0005	0.0005
init weight range	0.1	0.1	0.1	0.1
dropout	0	0	0	0
lambda	0.00000001	0	0.00000001	0

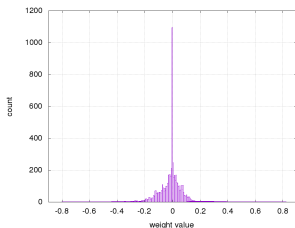
Sparsity results



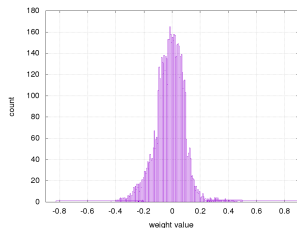
FNN sparse weights histogram



FNN no sparse weights histogram

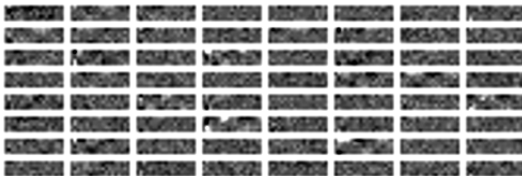


AE+FNN sparse weights histogram

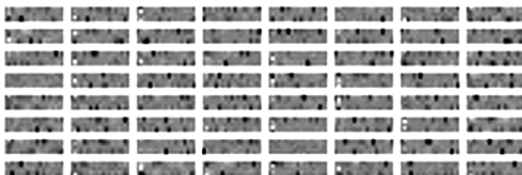


AE+FNN no sparse weights histogram

Sparsity results

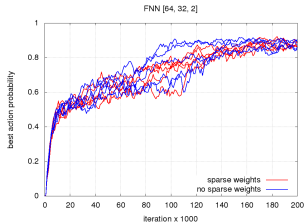


FNN sparse weights visualisation

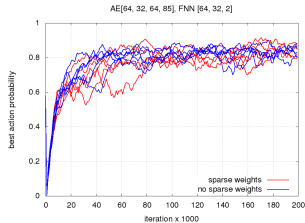


AE+FNN sparse weights visualisation

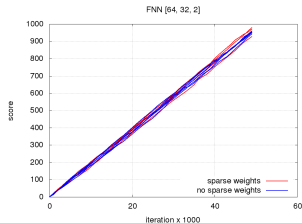
Score results



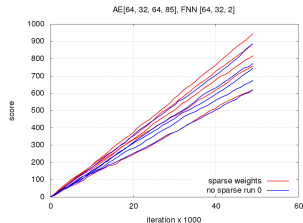
FNN progress comparison



AE+FNN progress comparison

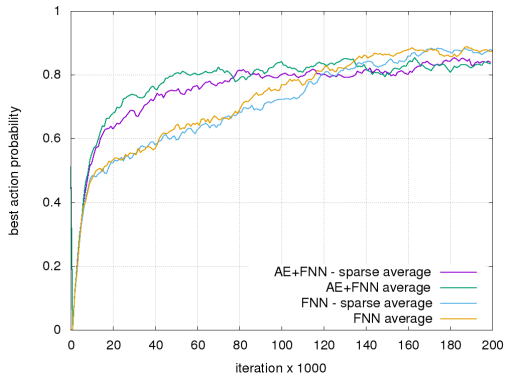


FNN score



AE+FNN score

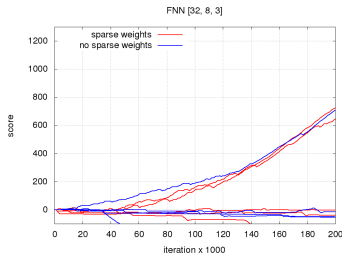
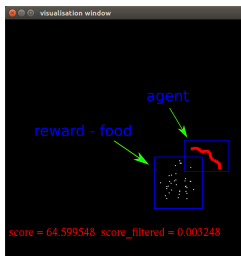
Results



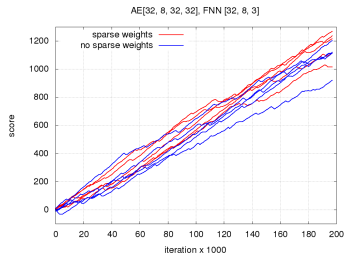
Average training progress comparison

	average score	best score	worst score	average best action probability [%]
FNN sparse weights	957.31	978.3	927.31	94.04
FNN nospase weights	951.5	959.3	942.644	95.95
AE+FNN sparse weights	763.58	942.97	618.66	88.16
AE+FNN no sparse weights	737.78	884.98	618.99	87.19

Snake game experiment

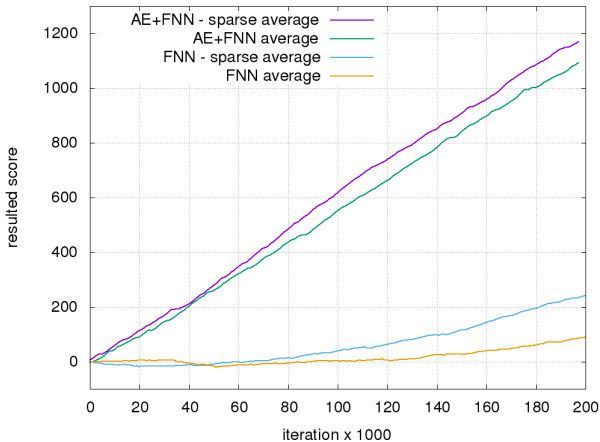


FNN score progress comparison



AE+FNN score progress comparison

Snake game experiment



Training worms score progress for best networks



<https://github.com/michalnand/robotics>

https://github.com/michalnand/machine_learning

michal.nand@gmail.com