Q-function aproximation in Q-learning algortihms using neural network

Ing. Michal Chovanec*
Department of technical cybernetics
Faculty of Management of science and Informatics
Prof. Ing. Juraj Miček, PhD.
michal.chovanec@yandex.com

Abstract

This paper is focused on results in Q-learning value function approximation using basis function neural network. For large state space is function approximation necessary, but common feed forward neural network can't be learned. We present neural network of basis functions, which can be learned using backpropagation.

Categories and Subject Descriptors

H.4 [**Machine learning**]: Artificial inteligence; H.4 [**Machine learning**]: Reinforcement learning; H.4 [**Machine learning**]: Neural networks

Keywords

Q-learning, reinforcement learning, neural network

1. Introduction

In learning systems based on supervised learning - set of inputs and required outputs can be error estimated and minimalized using relevant methods. There are many problems, where this can't be done - required value is unknown for most of cases. One of possible solutions is reinforcement learning [1], [2]. In reinforcement learning system is output from control unit (agent) sequence of actions. For each action is obtained reward from environment defined with Markovov decision process [3], [4], which is equal to zero in most cases. In few cases are rewards non zero, and can be used in learning process.

In general, we can define agent position in system as state vector $s(n) \in \mathbb{S}$, in step n, and action vector as $a(n)$. Goal is to find sequence π of actions to maximize final reward defined as

$$\Lambda(\pi) = \sum_{n=0}^{L(\pi)} \gamma^n P_{\pi(n)}(s(n), s(n-1)) \qquad (1)$$

*Recommended by thesis supervisor: Prof. Ing. Juraj Miček, PhD.
Defended at Faculty of Management of science and Informatics, University of Zilina

where $\gamma \in \langle 0, 1 \rangle$ is discount factor $P_{\pi(n)}(s(n), s(n-1)$ is reward function when transfer from state $s(n-1)$ into $s(n)$ obatained using strategy $\pi(n)$ $L(\pi)$ is sequence π length.

For huge state spaces it is hard to compute $P(s, s')$. One of posible way how to find maximum $\Lambda(\pi)$ is Q-learning algorithm.

2. Q-learning algorithm

Q-learning algorithm autor is Christopher J.C.H. Watkins, published in 1992 [5] and few other can be found in [6] or [7]. Convergence into optimal strategy (acccording to equation 1) was proven in in [8], [9], [10] and [11].

Let $\mathbb{S}$ is set of states a $\mathbb{A}$ is set of actions where $\mathbb{S} \in \mathbb{R}^{n_s}$, $\mathbb{A} \in \mathbb{R}^{n_a}$, n_s and n_a are dimensions of state space and dimensions of actions space.

For environment is given reward funcion as $R(s(n), a(n))$, which is reward for agent action $a(n)$ done in state $s(n)$. Usually equal to zero. For convergence is just one non zero value required.

Value function is defined as

$$Q_n(s(n), a(n)) = R(s(n), a(n)) + \gamma \max_{a(n-1) \in \mathbb{A}} Q_{n-1}(s(n-1), a(n-1)) \qquad (2)$$

- $R(s(n), a(n))$ is reward funcion
- $Q_{n-1}(s(n-1), a(n-1))$ is value function in state $s(n-1)$ for action $a(n-1)$
- γ is discount factor, $\gamma \in (0, 1)$.

Function 2 computes action values in all states : agent is located in state $s(n)$ using $a(n)$ with imediate reward $R(s(n), a(n))$, from state $s(n-1)$ and fraction of best possible Q value in state $s(n-1)$. This ilustrates fig. 1.

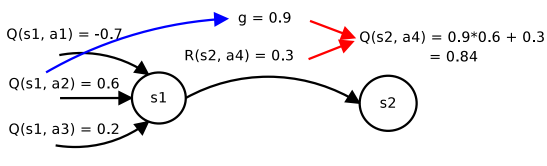

Figure 1: Q value function for $\gamma = 0.9$

The term $\max_{a(n-1) \in \mathbb{A}} Q_{n-1}(s(n-1), a(n-1)$ is responsible for convergence into optional strategy independly on action selection Another point of view is SARSA algorithm [12]

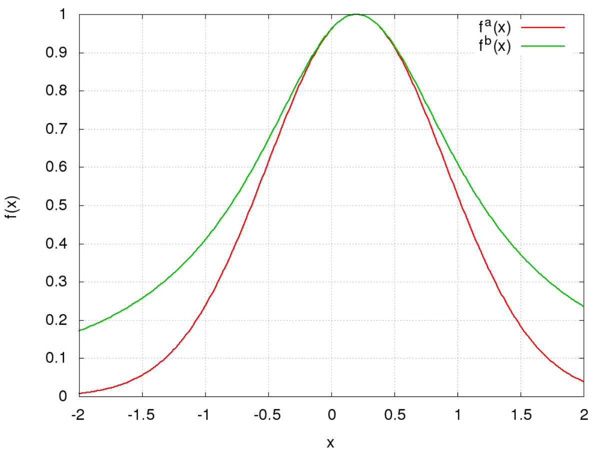

Figure 2: Basis function for one dimension

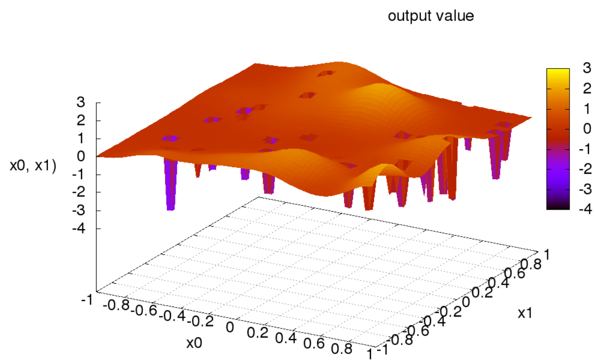

Figure 3: Q value function example

$$Q_n(s(n),a(n)) = (1-\alpha)Q_{n-1}(s(n),a(n)) + \alpha(R(s(n),a(n)) + \gamma Q_{n-1}(s(n-1),a(n-1))) \tag{3}$$

where $\alpha \in (0,1)$, and $Q_n(s(n),a(n))$ set on mean value and depends on action's selection strategy.

3. Solution design

For small state spaces can be $Q_n(s(n),a(n))$ stored as table. In huge spaces, or continuous spaces it is impossible, from two main reasons

1. huge memory requirements
2. all state transactions have to be visited

Function can be approximated, using neural networks. Common feed forward percepton network can't be learned [17]. From neural network is required to compute correct $Q_{n-1}(s(n),a(n))$ values, and learn value in $Q_n(s(n),a(n))$. Learning neural network in $Q_n(s(n),a(n))$ changes values in all other points s and a. One of solution is to define basis function neural network [13], [14], [15], [16]. For finite and discrete count of actions can be for each action defined own Q function.

Few examples of tested basis functions

$$f_j^1(s(n),a(n)) = e^{-\sum_{i=1}^{n_s}\beta_{aji}(n)(s_i(n)-\alpha_{aji}(n))^2} \tag{4}$$

$$f_j^2(s(n),a(n)) = \frac{1}{1+\sum_{i=1}^{n_s}\beta_{aji}(n)(s_i(n)-\alpha_{aji}(n))^2} \tag{5}$$

$$f_j^3(s(n),a(n)) = e^{-\sum_{i=1}^{n_s}\beta_{aji}(n)|s_i(n)-\alpha_{aji}(n)|} \tag{6}$$

where
$\alpha_{aji}(n) \in \langle -1,1 \rangle$ is maxima position
$\beta_{aji}(n) \in (0,\infty)$ function shape.

First two are plotted on fig. 2

For symetrical states transactions we can write more simple form

$$f_j^1(s(n),a(n)) = e^{-\beta_{aj}\sum_{i=1}^{n_s}(s_i(n)-\alpha_{aji})^2} \tag{7}$$

$$f_j^2(s(n),a(n)) = \frac{1}{1+\beta_{aj}\sum_{i=1}^{n_s}(s_i(n)-\alpha_{aji})^2} \tag{8}$$

$$f_j^3((n)s,a(n)) = e^{-\beta_{aj}\sum_{i=1}^{n_s}|s_i(n)-\alpha_{aji}(n)|} \tag{9}$$

Approximate Q-value function can be written as linear combination of l basis functions

$$Q^x(s(n),a(n)) = \sum_{j=1}^{l} w(n)_j^x f_j^x(s(n),a(n)) \tag{10}$$

where $w(n)_j^x$ are weights.

These functions have independent parameters, and approximate independent area of state space. Parameters can be learned separately.

3.1 Hybrid basis functions

In Q function can be spotted two basic shapes 3 - peaks, and hills. For peaks are reponsible negative $R(s(n),a(n))$ values and for hills positive $R(s(n),a(n))$ values.

We can combine two basic ideas an define new basis function, combining Gaussian curve with some sharp function

$$P_i(s(n),a(n)) = \begin{cases} r_{ai} & if\ s(n) = \alpha_i^1 \\ 0 & inak \end{cases} \tag{11}$$

$$H_j(s(n),a(n)) = w_{aj}e^{-\beta_{aj}\sum_{i=1}^{n_s}(s_i(n)-\alpha_{aji}^2)^2} \tag{12}$$

$$Q(s(n),a(n)) = \sum_{i=1}^{I} P_i(s(n),a(n)) + \sum_{j=1}^{J} H_j(s(n),a(n)) \tag{13}$$

where
α_j^1 are states where $H_j(s(n))$ have non zero values
α_j^2 are states where $f_j(s(n),a(n))$ have maximum

r_{ai} is negative reward value $R(s(n),a(n))$
w_{aj} is weight
β_{aj} is shape of Gaussian part, and $\beta > 0$
I a J are numbers of functions

Names P a H represent peak and hills.

4. Experiment results

For approximation hundreds of experiments were done and statistically computed error values. Goal is to navigate robot in 2 dimensional space into target. There were 8 actions of movement, which changes state as $s(n) = s(n-1) + a(n)dt$. Where set of actions

$$\mathbb{A} = [\; [0,1],[0,-1] \; [1,0],[-1,0], \; [1,-1],[1,1], \; [-1,-1],[-1,1] \;]$$

There was 64x64 discrete positions - optional solution can be computed, and used for approximation quality comparsion. In environment only is one positive reward and few zero rewards, example of one of four testing environments on fig 4.

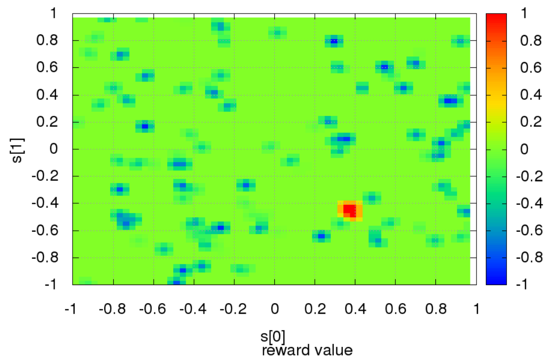

Figure 4: Reward function R(s(n), a(n)), map 2

For approximation these functions has been tested

1. sparse table
2. Gauss curve $f_j^1(s(n),a(n))$, equation 9
3. Gauss curve $f_j^1(s(n),a(n))$ combined with sparse table
4. Kohonen neural network modification siete $f_j^2(s(n),a(n))$
5. Kohonen neural network modification siete $f_j^2(s(n),a(n))$ combined with sparse table
6. Gauss curve combined with adaptive table (peak and hill), equation 13

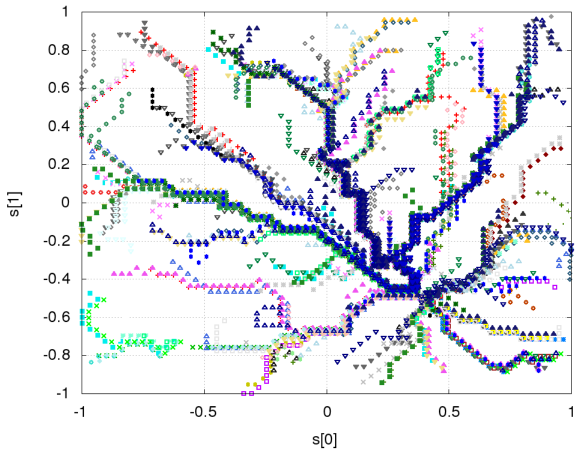

Figure 5: Agents path, optional solution

Agents trajectory for optional solution can be seen on fig 5.

Summary error results compraing with optional solution for 20 trials runs, in all four envinments are on figures 6, 7, 8, 9.

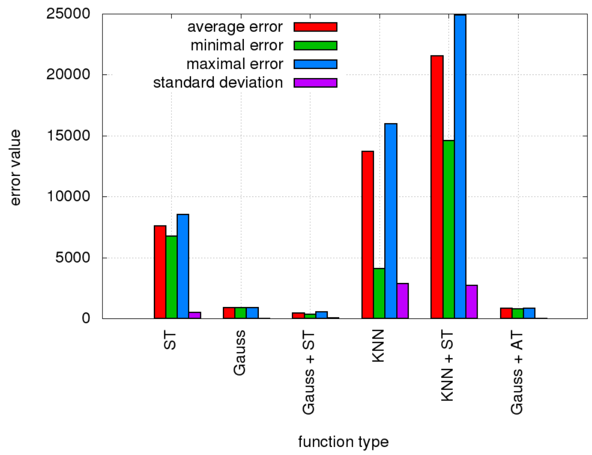

Figure 6: Summary results for map (environment) 0

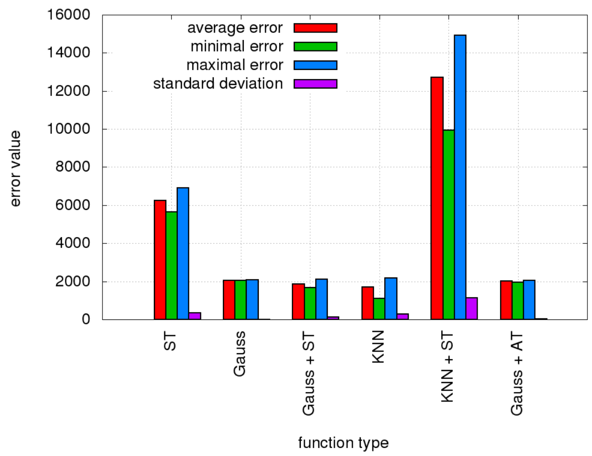

Figure 7: Summary results for map (environment) 1

And trajectories for three best results 10, 11, 12

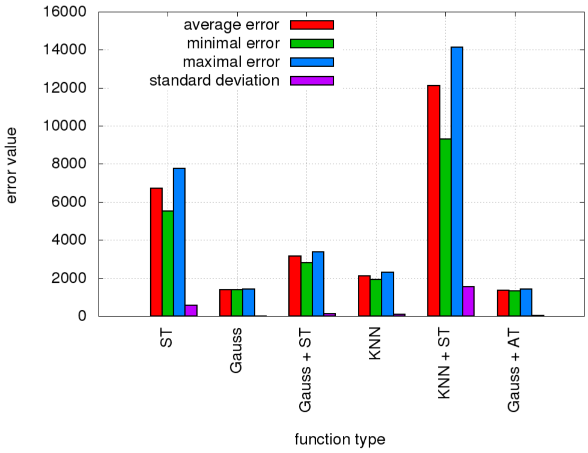

Figure 8: Summary results for map (environment) 2

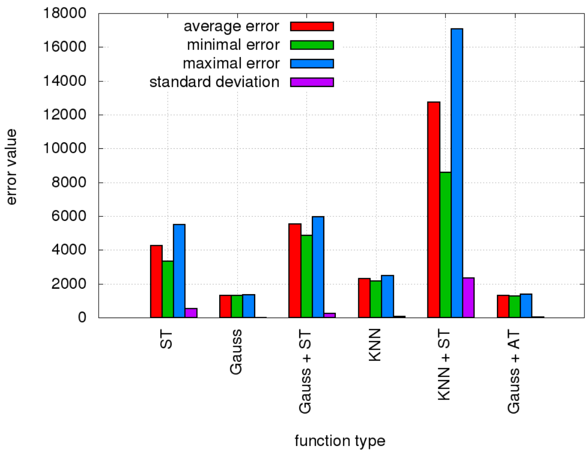

Figure 9: Summary results for map (environment) 3

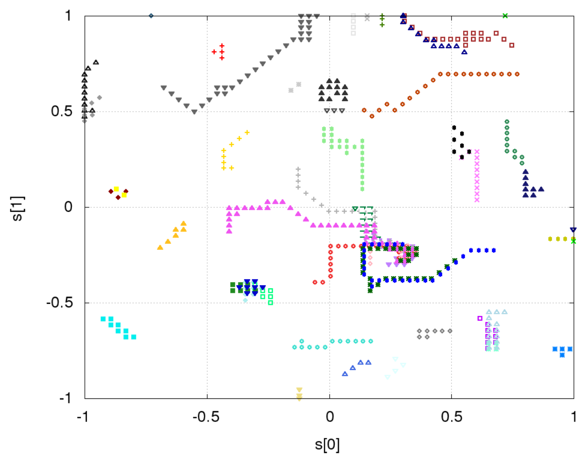

Figure 10: Agents path using Gaussian curve

5. Conclusions

From tested function, minimal error occures Gauss curve, Gauss curve combined with sparse table, and Gauss curve combined with adaptive table (peak and hill function). As we can seen, only acceptalbe agents trajectory results are for Gauss curve combined with adaptive table. All experiments results can be visited on [19] (more than 10000 results files), with full access

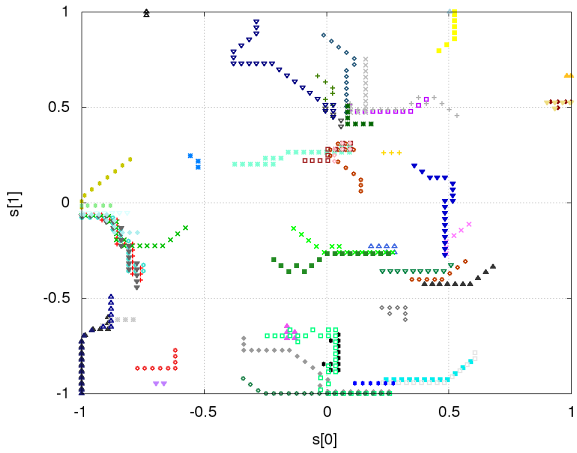

Figure 11: Agents path using Gaussian curve combined with sparse table

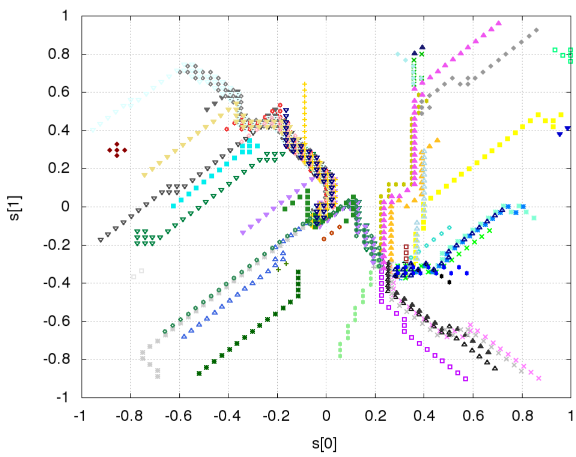

Figure 12: Agents path using Gaussian curve combined with adaptive table

to source codes for further research.

References

[1] Peter Dayan, Christopher J.C.H. Watkins, Reinforcement Learning `http://www.gatsby.ucl.ac.uk/~dayan/papers/dw01.pdf`

[2] Daniel Dewey, Oxford Martin Programme on the Impacts of Future Technology, Future of Humanity Institute : Reinforcement Learning and the Reward Engineering Principle `http://www.danieldewey.net/reward-engineering-principle.pdf`

[3] Markovove rozhodovacie procesy, stručne : Pieter Abbeel UC Berkeley EECS : Markov Decision Processes and Exact Solution Methods `http://www.cs.berkeley.edu/~pabbeel/cs287-fa12/slides/mdps-exact-methods.pdf`

[4] Martin L. Puterman : Markov Decision Processes: Discrete Stochastic Dynamic Programming , isbn 9781118625873, rok 2014, `https://books.google.sk/books?id=VvBjBAAAQBAJ`

[5] CHRISTOPHER J.C.H. WATKINS, PETER DAYAN : Technical Note Q-Learning, Machine Learning, 8,279-292 (1992) `http://www.gatsby.ucl.ac.uk/~dayan/papers/cjch.pdf`

[6] Q-learning 1 https://www-s.acm.illinois.edu/sigart/docs/QLearning.pdf

[7] Q-learning 2 http://mnemstudio.org/path-finding-q-learning-tutorial.htm

[8] Francisco S. Melo Institute for Systems and Robotics, Instituto Superior Técnico, Lisboa, PORTUGAL : Convergence of Q-learning: a simple proof http://users.isr.ist.utl.pt/~mtjspaan/readingGroup/ProofQlearning.pdf

[9] Eyal Even-Dar, Yishay Mansour : Convergence of optimistic and incremental Q-learning, http://web.cs.iastate.edu/~honavar/rl-optimistic.pdf

[10] Carden, Stephen, "Convergence of a Reinforcement Learning Algorithm in Continuous Domains" (2014). All Dissertations. Paper 1325. http://tigerprints.clemson.edu/cgi/viewcontent.cgi?article=2326&context=all_dissertations

[11] Francisco S. Melo and M. Isabel Ribeiro, Convergence of Q-learning with linear function approximation, Proceedings of the European Control Conference 2007 Kos, Greece, July 2-5, 2007, http://gaips.inesc-id.pt/~fmelo/pub/melo07ecc.pdf

[12] Karan M. Gupta Department of Computer Science Texas TechUniversity Lubbock, TX 79409-3104 : Performance Comparison of Sarsa(λ) and Watkin's Q(λ) Algorithms, http://www.karanmg.net/Computers/reinforcementLearning/finalProject/KaranComparisonOfSarsaWatkins.pdf

[13] Francisco S. Melo, Sean P. Meyn, M. Isabel Ribeiro An Analysis of Reinforcement Learning with Function Approximation, Appearing in Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 2008 http://www.machinelearning.org/archive/icml2008/papers/652.pdf

[14] David Silver : Lecture 6: Value Function Approximation http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/FA.pdf

[15] Francisco S. Melo M. Isabel Ribeiro : Q-learning with linear function approximation http://gaips.inesc-id.pt/~fmelo/pub/melo07tr-b.pdf

[16] Marina Irodova and Robert H. Sloan : Reinforcement Learning and Function Approximation, 2005, American Association for Artificial Intelli-gence (www.aaai.org) http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.81.7833&rep=rep1&type=pdf

[17] Mae L. Seto Springer Science & Business Media, 9. 12. 2012, Marine Robot Autonomy, ISBN 1461456592, chap 7.3.3.2

[18] video robota Motoko Aftermath Michal Chovanec, youtube https://www.youtube.com/watch?v=8sskJN_zuko

[19] Michal Chovanec, Q-learning zdrojové súbory https://github.com/michalnand/q_learning

[20] Michal Chovanec, Motoko robot zdrojové súbory https://github.com/michalnand/motoko_after_math_linefollower