CS 181 Machine Learning
Practical 4 Report, Team *la Dernière Dame M*

(Jeremiah) Zhe Liu[1], (Vivian) Wenwan Yang[2], and Jing Wen[1]

[1]Department of Biostatistics, Harvard School of Public Health
[2]Department of Computational Science and Engineering, SEAS

May 6, 2015

1 Exploratory Analysis

2 Method

2.1 Rationale on Model Choice

2.2 Estimation

2.2.1 SVM

The SVM maps the input vectors into high-dimensional feature space and returns the maximum margin hyper plane. SVM algorithm with a linear kernal could be implemented by constructing the following problem:

$$\min_{r,w,b} \frac{1}{2}\|w\|^2$$
$$s.t. y^{(i)T}\left(w^T x^{(i)} + b\right) \geqslant 1, i = 1...n$$

Data:
The SVM is implemented in a static tree trunk setting, and only the first tree ahead of the monkey is considered. Given the command to jump, the current state is labeled as 1, otherwise current state is labeled as -1. Hence at each step, the training set includes a current state $x^{(i)} = ['vel', 'top', 'bot'] = [\Delta x, \Delta y, \Delta z]$ and related label $y^{(i)} \in \{-1, 1\}$.

We can then choose a high-dimension feature space. For the below example, nine features from fundamental analysis are selected by adding second and third order of the feature.

$$\mathbf{\Phi} = \left[\Delta x, \Delta y, \Delta z, \Delta x^2, \Delta y^2, \Delta z^2, \Delta x^3, \Delta y^3, \Delta z^3\right]$$
$$k(x, z) = \mathbf{\Phi}^T \mathbf{\Phi}$$

2.2.2 Model-based estimation

Optimization Algorithm
Value iteration could be implemented in order to find the value function for small MDPs. The value iteration is illustrated as follows:

For each state, initialize $V(s) := 0$.
Repeat until converge
{For each state:

$$V(s) = R(s) + \max_{a \in A} \gamma \sum_{s'} P(s'|s, a) V(s')$$

}

After $V(s)$ and P_{sa} were learnt by the above algorithm, the optimal a is given by:

$$a^* = \max_{a \in A} P(s'|s, a) V(s)$$

2.2.3 Q-learning

2.2.4 exploitation vs. exploration

$e = 0.001$
$\epsilon = \frac{e}{k}$
k = # of actions taken from state s

2.3 **Numerical Challenges & Further Modification**

2.3.1 Parameter Selection

why $\gamma = 0.95$ $\alpha = 1/k$

3 Discussion & Possible Directions

3.1 Locally Weighted Regression

One method has effectively dealt with practical situations with large state spaces is locally weighted regression (LWR). It improves the performance of a discrete-state algorithm by applying a function approximator to a continuous-state problem. The method consists of two steps. The first one involves learning the value function over a small number of discrete states. The second step consists focusing the function approximator to generalize from the discrete states to a continuous state space, thereby effectively using previously discarded state information.

LWR creates a local model of a function around a query point. Training points are weighted according to a function of their distance from the query point, and this function is typically a kernel function. The below algorithm describes how LWR techniques construct a value function approximation. First, we obtain approximations for the current Q-value, and the maximum value from the resulting state, s'. We keep track of the points used in the LWR prediction of q for later and their associated weights. Then the new approximation of $Q(s, a)$ is calculated using the standard Q-learning update rule. This value is then used as a normal supervised training point for the LWR subsystem. We then update each of the points in K, bringing this value closer to $Q(s, a)$ depending its proximity to (s, a), as measured its kernel weights, ξ_i.

Input:
Experience, (s, a, r, s')
Learning rate, α
Discount factor, γ
LWR bandwith, h
1. $q \leftarrow Q_{predict}(s, a)$
2. $q_{next} \leftarrow max'_a Q_{predict}(s', a')$
3. $K \leftarrow$ set used in calculation of q
4. $\xi_i \leftarrow exp(-(q - k_i)^2/h^2)$
5. $q_{new} \leftarrow q + \alpha(r + \gamma q_{next} - q)$
6. Lean $Q(s, a) \leftarrow q_{new}$
7. **for** each point, (s_i, a_i) in K do
8. $Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \xi_i(q_{new} - Q(s_i, a_i))$

LWR is an aggressive learning algorithm that is fast to train. It can make reasonable predictions based on very little training data. Only simple linear functions are required to approximate the global function in the neighborhood of the query point, which avoids the difficult task of generating a global model of a nonlinear function. In addition, since LWR retains all of the training data, the approximation could be reevaluated on the original training points.

3.2 Neural Network

To deal with the large state space, besides discretizing the position space into bins and locally weighted regression, another strategy is implementing function approximation, such as a neural network. $\hat{Q}_\theta(s, a)$ could be represented by a non-linear approximator. The steps are illustrated as below:

a. Start with initial parameter values: θ.
b. Take action according to an exploit/explore policy, so that the results should converge to greedy policy.

c. Perform TD update for each parameter:

$$\theta_i = \theta_i + \alpha(R(s) + \beta \max_{a'} \hat{Q}_\theta(s', a') - \hat{Q}_\theta(s, a)) \frac{\partial \hat{Q}_\theta(s, a)}{\partial \theta_i}$$

d. Repeat step b and c until convergence.

Q-learning with nonlinear approximators often works well in practice. Typically the space has many local minima and convergence is not guaranteed.

Reference

1. Moerman W, Bakker B, Wiering M. (2009) **Hierarchical Assignment of Behaviours to Subpolicies**. *Cognitive Artificial Intelligence, Utrecht University*.

2. Aljibury H. (2001) **Improving the Performance of Q-learning with Locally Weighted Regression**. *Electrical Engineering, University of Florida*.

3. Srebro N, Jaakkola T.(2003) **Weighted low-rank approximations**. *Proceedings of the Twentieth International Conference* 720727.

4. R Salakhutdinov, A Mnih. (2008) **Probabilistic Matrix Factorization**. *Advances in Neural Information Processing Systems* Vol. 20

5. Koren, Y. (2008) **Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model**, *Proc. 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.