Reinforcement learning - Lab 2

Dimitri Diomaiuta - 30598109

University of Southampton

1 Radial basis function

In this paper we analyze how to use radial basis function (RBF) to solve a linear regression and a reinforcement learning problem. A RBF is a real-valued function ϕ whose value depends on the distance from a point called origin such that $\phi(x) = \phi(||x||)$. RBF models are particularly useful to generate basis functions from feature vectors in order to solve function approximation problems via linear combination. The output of the linear combination is described by equation 1.

$$f(x) = \sum_{j=1}^{J} w_j \phi(||x - c_j||) \tag{1}$$

The radial basis function is usually assumed to be Gaussian and, hence, to be calculated as shown in equation 2 [1].

$$\phi(||x - c_j||) = \exp\left(-\frac{||s - c_j||^2}{2\sigma_j^2}\right) \tag{2}$$

As we can observe from equation 1, J defines the number of centers and, hence, the number of radial basis functions used in the linear combination and w_j the weights for each function.

2 Linear regression with RBF

In this section we solve a linear regression problem by exploiting the RBF model described in section 1. We solve the problem in a standard manner as follows:

- Load a raw dataset
- Divide into train and test set
- Given a J compute the design matrix of features using a RBF model
- Learn the weight vector by stochastic gradient descent on the train set
- Compute the root mean squared error of the model tested on the test set
- Compare the model with one deriving the weight vector by using the pseudoinverse.

We performed the steps for different number of radial basis function (J) to analyze which one is the model that generalizes better the pattern of the input data. The input dataset we used for the task is the UCI *red wine quality* dataset

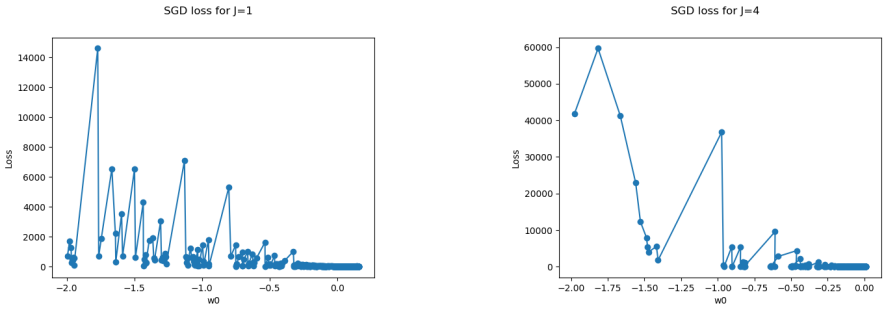

Fig. 1. Loss convergence for different J

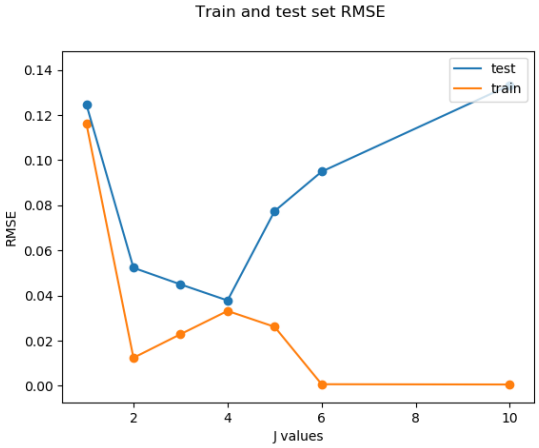

Fig. 2. Root mean squared error for train and test set

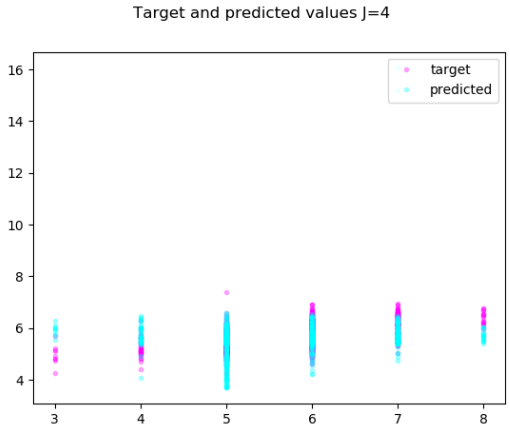

Fig. 3. Best model divergence from target data

[2]. We tested the program for $J = \{1, 2, 3, 4, 5, 6, 10\}$. We can observe the loss function convergence of SGD, the divergence of the best obtained model from the target data and the train and test set RMSE respectively in figures 1, 2, 3.

We can see, for different J values, that the error is minimized and that the models converge (Figure 1). Having more radial basis functions, hence a higher J, makes the model predictions more accurate and closer to the target. Increasing the J, however, is not always resulting in better and more accurate models that capture the general pattern of the input data. This problem is well known as the overfitting problem and results in a model being influenced not only by the pattern but also by the noise of the data. An overfitted model performs well on the training data and poorly on the testing data. Figure 2 describes this scenario. We also deduce that the RBF model with $J = 4$ produces the best results. This is confirmed by figure 3, we can in fact see that the predictions of this model fall close to the target ones.

3 Reinforcement learning with RBF

In a reinforcement learning setting RBF models can be exploited to derive a value function approximation. Tabular methods are impractical when dealing with problems that have big state spaces (e.g go has 10^{170} possible states) or that have continous valued features (tabular methods need preprocessing such tile coding). The first step is to derive a feature vector representation of a state via RBF ($x(s)$). The second one is to introduce a weight vector. The value function approximation is derived by a linear combination of the weight and feature vector, as shown in equation 3 [1].

$$\hat{v}(s, w) = w^T x(s) = \sum_{i=1}^{d} w_i x_i(s) \tag{3}$$

The problem now resembles a supervised learning setting and, thus, we can solve it by updating the weight vector exploiting gradient descent. The weight update using SGD is calculated in a similar manner to linear regression, as equation 4 shows [1].

$$w_{t+1} = w_t + \alpha \left[U_t - \hat{v}(S_t, w_t)\right] x(S_t) \tag{4}$$

Using this knowledge we solved the mountain car problem by deriving a value function approximation. The steps we took are the following:

- Obtain the value function used by tabular discretization in *Lab one*.
- Construct feature vector state representation using RBF
- Use tabular discretization value function as an unbiased estimate (U_t)
- Compute the weight vector using the pseudoinverse.

We performed the steps for different numbers of radial basis functions. In this case we could compute the weight vector using the pseudoinverse since we used as unbiased estimate of the target value function (U_t) the values obtain by tabular

discretization stored in *q_table*. When deriving the value function approximation on-line the unbiased estimator can be, for example, the return G_t in Monte Carlo methods or the target $R_{t+1}+\gamma\hat{v}(S_{t+1}, w)$ in temporal difference methods. We can observe the centers of the radial basis functions for $J = 10$, the value function derived via tabular discretization and the value functions approximation derived with different J respectively in figures 4, 5 and 6. As we can see from figure

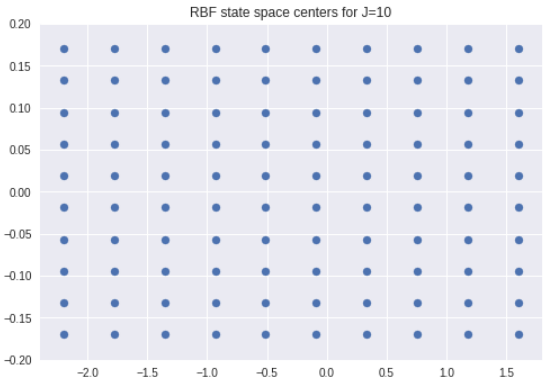

Fig. 4. RBF centers used to create feature vector state

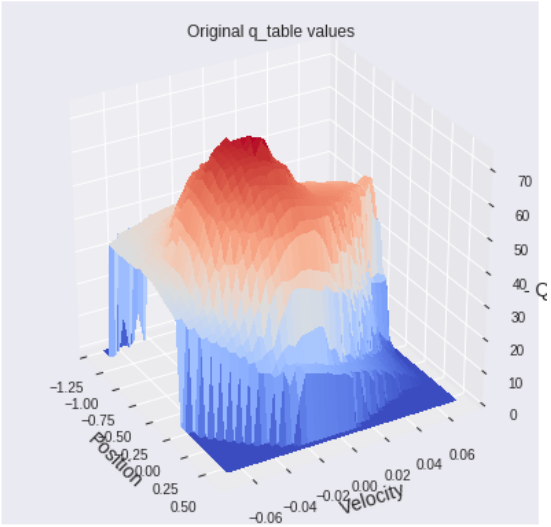

Fig. 5. q_table derived value function

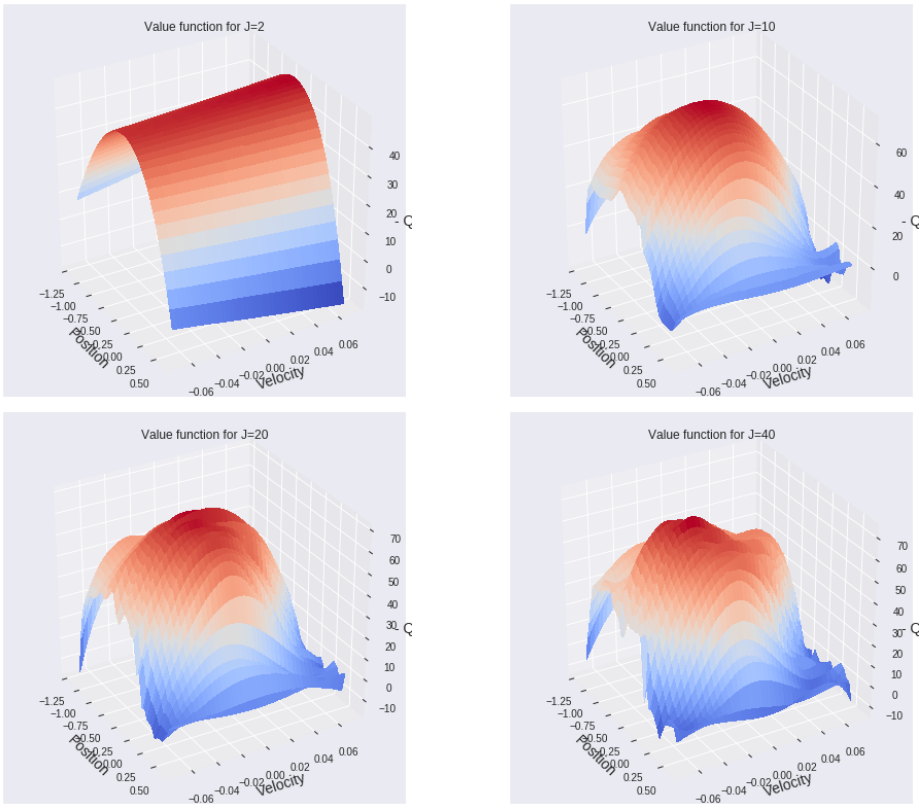

Fig. 6. Value functions approximation for different J

6, the data indicate that increasing the number of radial basis functions results into a more precise approximation. The figure with $J = 40$ is the one that is closer to the figure plotting the value function derived by the table discretization method, see figure 5. Increasing J, though, results in drastically incrementing the computational steps of the algorithm making this infeasible for very high J values. In this case a J value of 10 is enough to drive the car on top of the mountain.

References

1. Sutton, R.S. and Barto, A.G., 2011. Reinforcement learning: An introduction.
2. Archive.ics.uci.edu. (2019). UCI Machine Learning Repository: Wine Quality Data Set. [online] Available at: https://archive.ics.uci.edu/ml/datasets/Wine+Quality [Accessed 5 Mar. 2019].