# CMPUT 609 Assignment 2

Shaurya Seth

January 20, 2023

## Question 1

The tabular method is a special case of linear function approximation where the weight vector is the same number of dimensions as the number of states, $d = |S|$ and the feature vectors are a one-hot encoding of the state. This ensures that a single element in the weight vector corresponds to the value of that particular state.

For example, the feature vectors of an MDP with three states would be [1, 0, 0], [0, 1, 0], and [0, 0, 1].

## Question 2

We could use asymmetric rectangular tilings in this case where the tilings are longer (have a greater receptive field) in the dimension that requires more generalization.

## Question 3

Using (9.19) the step-size parameter could be:

$$\alpha = \frac{1}{20 \times 78}$$

This would be reasonable because in our tile coding, 78 elements of the feature vector would be active at any given time step and the $\frac{1}{78}$ would result in one-trial learning while the $\frac{1}{20}$ would ensure that it takes 20 steps on the same feature vector for the learning to asymptote.

## Question 4

Using the given information in (9.19) we get:

$$\alpha = (\mathbf{x}_t^\top \mathbf{x}_t)^{-1}$$

Using linear function approximation in (9.7) we get:

$$\begin{aligned}
\mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha[U_t - \mathbf{w}_t^\top \mathbf{x}_t]\mathbf{x}_t \\
&= \mathbf{w}_t + [U_t - \mathbf{w}_t^\top \mathbf{x}_t]\mathbf{x}_t(\mathbf{x}_t^\top \mathbf{x}_t)^{-1} \\
&= \mathbf{w}_t + [U_t - \mathbf{x}_t^\top \mathbf{w}_t](\mathbf{x}_t^\top)^{-1} \\
&= \mathbf{w}_t + \frac{U_t}{\mathbf{x}_t^\top} - \mathbf{w}_t \\
&= \frac{U_t}{\mathbf{x}_t^\top}
\end{aligned}$$

We see that the error goes to zero in one update:

$$\begin{aligned} error &= U_t - \mathbf{x}_t^\top \mathbf{w}_{t+1} \\ &= U_t - \frac{\mathbf{x}_t^\top U_t}{\mathbf{x}_t^\top} \\ &= U_t - U_t \\ &= 0 \end{aligned}$$

## Question 5

The logistic function and its derivative are given by:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

The approximate value function and its gradient becomes:

$$\hat{v}(s, \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x})$$

$$\nabla \hat{v}(s, \mathbf{w}) = \sigma(\mathbf{w}^\top \mathbf{x})(1-\sigma(\mathbf{w}^\top \mathbf{x}))\mathbf{x}$$

Using $z = \mathbf{w}_t^\top \mathbf{x}_t$ and (9.7) we get the learning algorithm:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha[U_t - \sigma(z)]\sigma(z)(1-\sigma(z))\mathbf{x}_t$$

## Question 6

Using $a = \sigma(z) = \sigma(\mathbf{w}_t^\top \mathbf{x}_t)$ the cross-entropy loss is given by:

$$C = -[U_t ln(a) + (1-U_t)ln(1-a)]$$

Taking the gradient of C we get:

$$\nabla C = (\sigma(z) - U_t)\mathbf{x}_t$$

Using (9.4) we get the learning algorithm:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha[\sigma(z) - U_t]\mathbf{x}_t$$

We find that this is much simpler than the update rule in the previous exercise.

## Question 7

This function can be learned exactly as long as there is a bias input because the output is simply the sum of the first four inputs thresholded by the bias unit to be positive or negative. The weight vector at convergence could be [-1, 1, 1, 1, 1, 0, 0, ...] with zeros for all the distractor inputs. Note that -1 works here as a bias because the problem has a strict inequality.

The error rate on the first example is the probability that the target will be zero because $\mathbf{w}_1 = \mathbf{0}$. This can be calculated as $P(Y_t = 0) = 1 - P(\sum_{i=2}^{5} x_1(i) = 0) - P(\sum_{i=2}^{5} x_1(i) = 1) = 0.5^4 + 4 \times 0.5^4 = 0.6875$. The error rate should asymptotically fall to zero because an exact solution can be learned. Both of these match the experimental results as shown in Figure 1.

The curve for 100 runs was fairly noisy while binning got rid of the initial point. So 1000 runs were chosen for the experiment to obtain a smoother curve. In general, we can be more confident in the error rate as we increase the number of runs (law of large numbers).

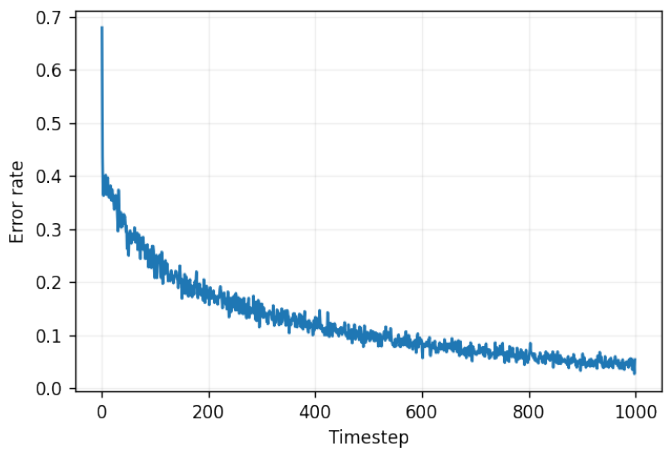

Figure 1: Learning curve averaged over 1000 runs.

## Question 9

With 10% noise the learning curve should have a higher overall error rate (it should move upwards). More specifically, the curve should start at the same point as Question 7 (random guessing with zero weights) and it should asymptote somewhere above 0.1. It is reasonable to think of 0.1 as the "minimum" error rate because that is what an exact solution without learning would achieve with the given noise.

We run the experiment with a 10% probability of a noisy target at every timestep as shown in Figure 2. It appears that the curve asymptotes somewhere around 0.25 which is more than twice our "minimum" error rate. This makes sense because every time a switched target is presented, the weights move in the wrong direction so not only do we incur error on the noisy target but also on some of the subsequent targets. This can be seen for the learning curve of a single run in Figure 7 (Appendix).

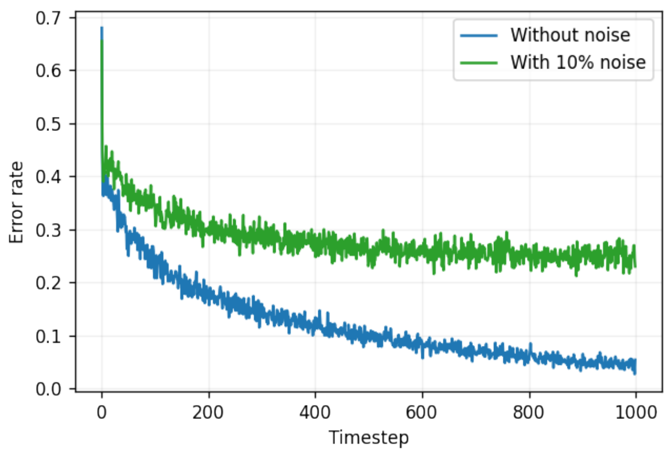

Figure 2: Learning curves averaged over 1000 runs with and without noise.

## Question 10

A reasonable guess would be that changing the step-size would effect the rate of learning but the way our perceptron is set up in Question 7, the step-size has no such effect as shown in Figure 3. Adding a constant step-size simply scales the weight vector by a constant. Since only the sign of the dot product is relevant, such a scaling has no effect on learning.

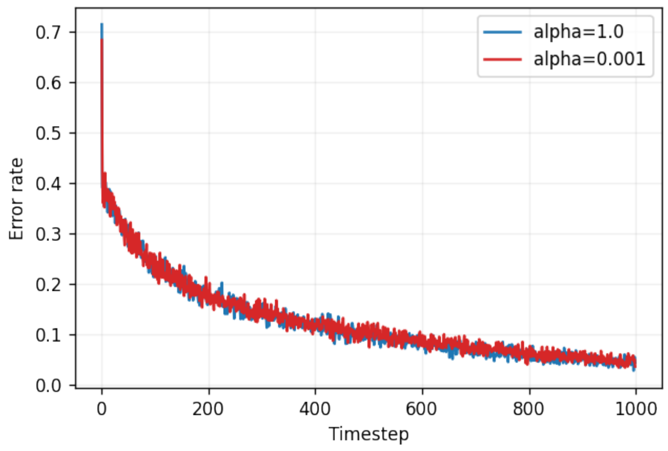

Figure 3: Learning curves averaged over 1000 runs for different step-sizes (classification).

For the regression problem, the expected value of $E_1$ for $\mathbf{w}_1 = \mathbf{0}$ can be calculated as:

$$\begin{aligned}\mathbb{E}[E_1] &= \mathbb{E}[(Y_1 - Z_1)^2] \\ &= \mathbb{E}[Z_1^2] \\ &= \mathrm{Var}(Z_1) + \mathbb{E}[Z_1]^2 \\ &= 6\end{aligned}$$

The best possible weight vector would sum the first four elements after the bias input:

$$\mathbf{w}^* = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & ... \end{bmatrix}$$

Assuming the learning rule finds this vector, the expected value of $E_1$ can be calculated as:

$$\begin{aligned}\mathbb{E}[E_t] &= \mathbb{E}[(Y_t - Z_t)^2] \\ &= \mathbb{E}[N_t^2] \\ &= \mathrm{Var}(N_t) + \mathbb{E}[N_t]^2 \\ &= 1\end{aligned}$$

We choose the step-size based on the rule of thumb in Section 9.6 (for $d = 20$) using $\tau = 10$ for convergence after 10 presentations:

$$\alpha = (\tau \mathbb{E}[\mathbf{x}_t^\top \mathbf{x}_t])^{-1} = 0.01$$

The learning curve for 10 individual runs ($n = 25$) is shown in Figure 8 (Appendix) while the learning curve for an average of 1000 runs in shown in Figure 4. We find that the curves do start around the expected value of $E_1$ we found earlier and approach $\mathbf{w}^*$:

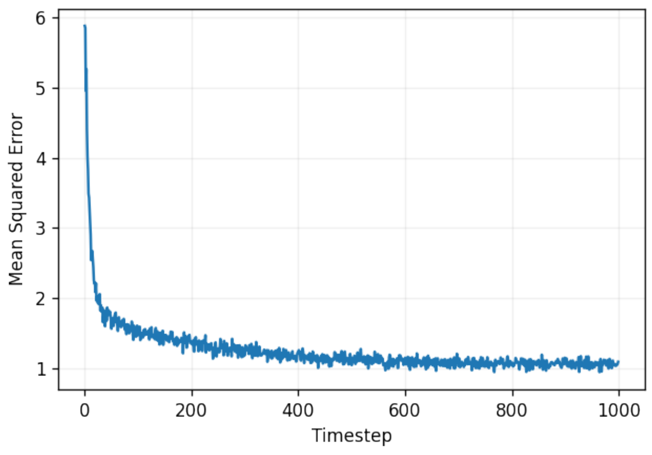

Figure 4: Learning curve (MSE) averaged over 1000 runs (regression).

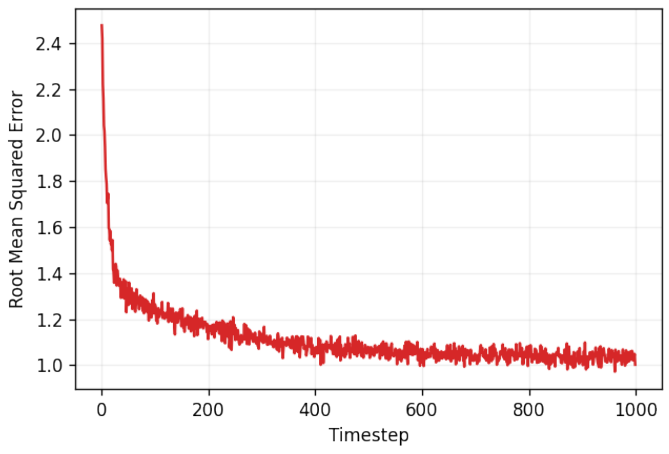

Figure 5: Learning curve (RMSE) averaged over 1000 runs (regression).

## Question 11

The value of around $\alpha = 2^{-6} \approx 0.01$ seems to be best for this setting as shown in Figure 6. The standard error is plotted for $r = 100$. For this, the step-sizes of $2^{-4}$ and $2^{-7}$ have significantly worse performance than $2^{-5}$.

We find that the rule of thumb in Section 9.6 was roughly correct.

The best value of $\alpha$ is inversely related to the size of the feature vector $d$.

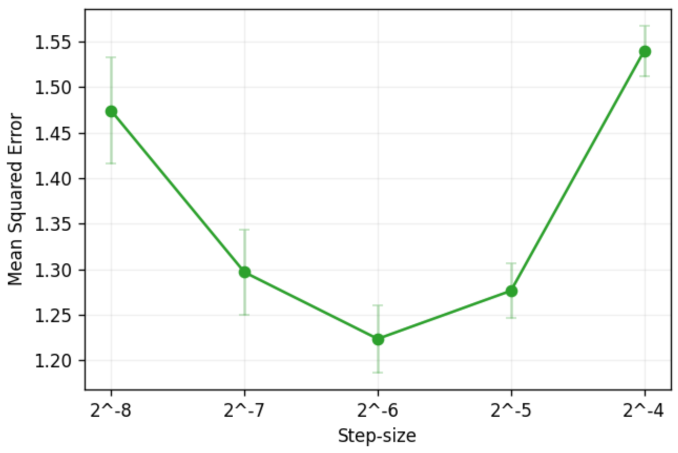

Figure 6: Parameter study for $d = 20$ (regression).

## Appendix

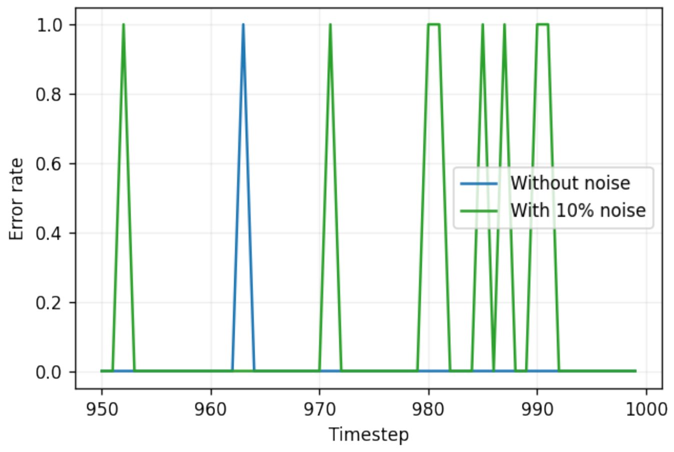

Figure 7: Learning curve for 1 run with and without noise (last 50 steps).

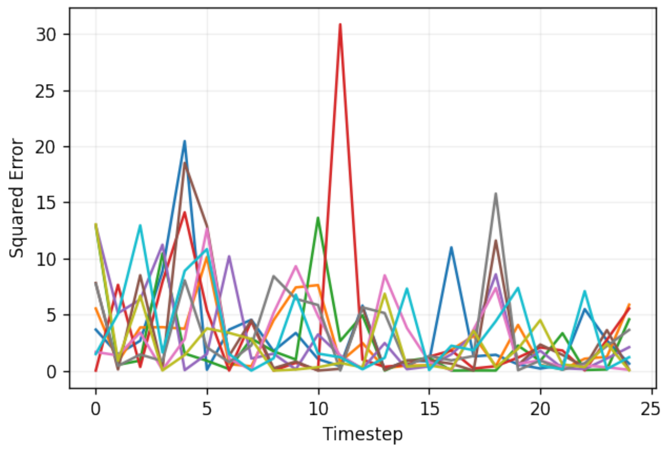

Figure 8: Learning curves for 10 individual runs (regression).