ME 469: Learning Comparison

MICHAEL RENCHECK

Question 1: The New Learning Aim

The new learning aim to be applied to the locally weighted linear regression (LWLR) algorithm is a different dataset formulation of replacing the motion model. Instead of trying to predict the change in state given the controls and some aspects of the current state (the original learning aim), the goal of this new formulation is to predict the next state given the current state and the controls. Both datasets were based on dataset 1 from the UTIAS study[1]. Table 1 compares the differences between the two learning aims.

Although, the learning aims are attempting to effectively achieve the same goal of estimating the true motion model, the implementation varies in a non-trivial way. In addition to predicting a different output, the overall input space is larger than the original learning aim. In addition to that, angle wrapping of the groundtruth heading in the new learning aim was handled by calculating the sine and cosine of each value. Whereas the original learning aim implemented angle wrapping logic in the dataset formulation.

Table 1. Comparison of the old and new learning aim

Learn the change in state, $\Delta\vec{x}$		**Learn the next state, $\vec{x}_{t+1}$**	
Input	**Output**	**Input**	**Output**
v, ω, θ_t	$\Delta x/\Delta t$	$v, \omega, x_t, y_t, \sin(\theta_t), \cos(\theta_t)$	x_{t+1}
v, ω, θ_t	$\Delta y/\Delta t$	$v, \omega, x_t, y_t, \sin(\theta_t), \cos(\theta_t)$	y_{t+1}
v, ω, x_t	$\Delta\theta/\Delta t$	$v, \omega, x_t, y_t, \sin(\theta_t), \cos(\theta_t)$	$\sin(\theta_{t+1}), \cos(\theta_{t+1})$

There is also some difference in the transferability between scenarios. The original learning aim is trying to predict a relative change in state, so this training data could be transferable if the robot was operated differently under the same environmental conditions, especially the change in x and y predictions. For the new learning aim, the training data is more specific to the measurement set up. Therefore, this training data might need to be modified to be reused. For example, if the world coordinates change, the training data would need to be modified to match the new world frame. These differences resulted in very different models of data given the different input spaces. Figure 1 shows a graphical comparison for the x component of the expected output for each learning aim. The new dataset formulation results in a highly non-linear data model with minimal noise across the input space which is very well suited to the LWLR algorithm.

Question 2: Learning Algorithm Comparison

Initially this dataset was used in conjunction with a neural network, which in theory can be very different or very similar to a locally weighted regression. This implementation was a network that only consisted of one hidden layer using the sigmoid activation function, so there are a few key differences compared to the LWLR.

Both algorithms require a set of data to be used as the training set and a set of data to test how well the training data represents the learning aim the algorithm is trying to predict. While both algorithms are a form of supervised learning, meaning they require a known/expected output for each set of inputs, the first major difference is that a neural network must be trained prior to making a prediction. Depending on the size of the training set and size of the network, this can lead to long training times required before any prediction can be made. However, once the

[1] (Halpern, Lui, Barfoot, & Leung, July 2011)

network is trained, each query is calculated very quickly even for a large test dataset since the network is essentially performing a series of multiplications and summation. LWLR is a form of lazy learning, meaning that prior to making a prediction, there is no training time required. Even though this method requires no training time, the actual time to make a prediction is directly correlated to the size of the training dataset and the number of inputs. In the final equation[2] to make a prediction for the LWLR, shown below, there is a matrix inversion. This portion of the computation will take the most time, especially for large training datasets. For the training dataset used for the new learning aim, a single prediction takes roughly half a second.

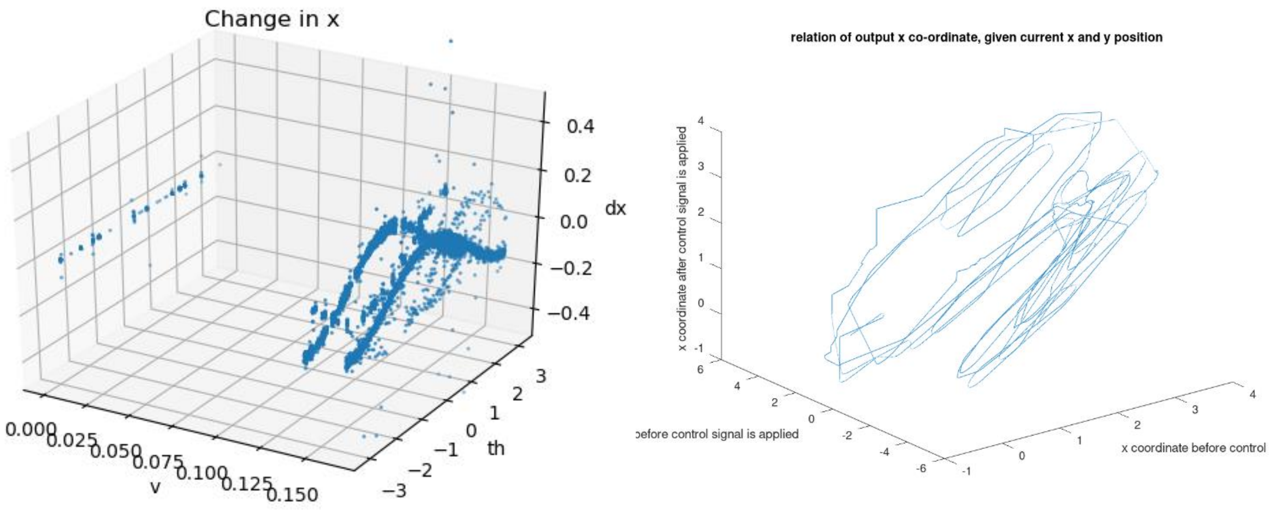


Figure 1. Comparison of the x output across some of the input space for each learning aim.

$$\hat{y} = \boldsymbol{q}^T(\boldsymbol{Z}^T\boldsymbol{Z})^{-1}\boldsymbol{Z}^T\boldsymbol{v}$$

q = query point
$\hat{y}$ = the predicted output
Z = weighted training data input matrix
v = weighted training data output vector

Another difference is how each algorithm is using weighting factors. In LWLR, the weighting factors are calculated directly based on a distance function, relating how far the query point is from each point in the training dataset. This is then put through a kernel function in order to create an inverse relationship between the distance and the weight; as distance increases, the corresponding weight should decrease. These weighting factors are completely dependent on the query point and are recalculated each time a new query is made. For a neural network the weighting factors are determined through trial and error during the training phase. A set of weights are randomly initialized for every connection to each unit for all layers of the network. The training data inputs are then propagated forward through the network and an error is computed between the predicted output and the known output. This error is then used to modify each weighting value by back propagating a stochastic gradient descent. This process is then repeated until the resulting prediction error is under a set threshold. These weights are then used to predict an output for each query in the test set.

Another major difference in the two algorithms, is the robustness to noise. Since the neural network is iteratively refining the weights across the whole training set, if there are any outliers in the training dataset, the resulting weights are minimally impacted by them. In contrast, these will greatly impact a prediction made using the LWLR

[2] (Atkeson, Moore, & Schaal, 1997)

algorithm. If the inputs corresponding to the outlier are close to the query point, the weighting factor attributed to the outlier will also be high, which will cause local linear fit (and subsequently the prediction) to be very inaccurate.

Lastly, a neural network has the advantage of being able to extrapolate and make accurate predictions even if the query was not directly inside the bounds of the training data input space. This is advantageous because there is a robustness when attempting to use the learned network in a general application. However LWLR is unable to make accurate predictions if the query is too far outside of the input space. As a given query get farther outside of the input space captured by the training data, the distance will grow resulting in small weighting values. This will produce a poor locally linear fit and return a prediction will be very far off from reality. This limitation may limit the usefulness of running a LWLR in a general application unless the training data sufficiently covers the input space.

Question 3: Algorithm Implementation on New Aim

Since the new learning aim was similar in format to the old learning aim (a regression problem with multiple inputs and corresponding outputs), very little effort was required to adopt the dataset to work with the algorithm. For this implementation, each output variable was assigned its own iteration of the LWLR algorithm to circumvent minimizing the error across four outputs. This allowed each output prediction to be computed directly allowing for a faster overall computation time. The only modifications that were made was adjusting the bandwidth parameter to dial in the best fit for this new dataset. For LWLR, the bandwidth is a design parameter to control the calculated weighting factor for each data point in the training set. It is represented by *h* in the following equations:

$$w_i = \sqrt{K\left(\frac{d(x_i, q)}{h}\right)}$$

$$K = exp\left(-\frac{d(x_i, q)^2}{h}\right)$$

K() = Kernel Function
d() = Distance function
h = fixed global bandwidth

Use a bandwidth of $h > 1$, the weighting factor increases for points farther away from the query. This creates a smoothing effect across the predicted output. However, smoothing too much will produce a poor fit of the predicted output. Conversely, using a bandwidth of $h < 1$, will increase weight of the data points very close to the query, making the local linear model more dependent on training data points very close to the query. Decreasing *h* too much will result in over fitting and the LWLR will fail at capturing the underlying function. The effect of the changing bandwidth on a generic exponential decay function can be visually observed in Figure 2.

The *h* value can only be tuned through testing and will very for each implementation. For the old learning aim and dataset, the best results were obtained with $h = 0.06$, but for the new learning aim and dataset, the best results came with $h = 10$. Figure 3 shows a graphical comparison of the variance across the x_{t+1} prediction for new learning aim testing dataset for $h = 1$ and $h = 10$. The $h = 10$ plot has a lower average variance across the entire dataset meaning that the predictions are a better overall fit across the whole test set. These same trend of a lower average variance for $h = 10$ was also observed for the other three output variables.

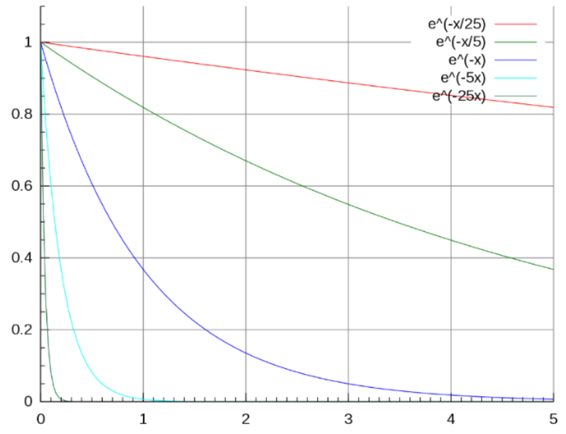


Figure 2. Graph of exponential decay functions similar to those used to calculate the weighting factors[3]

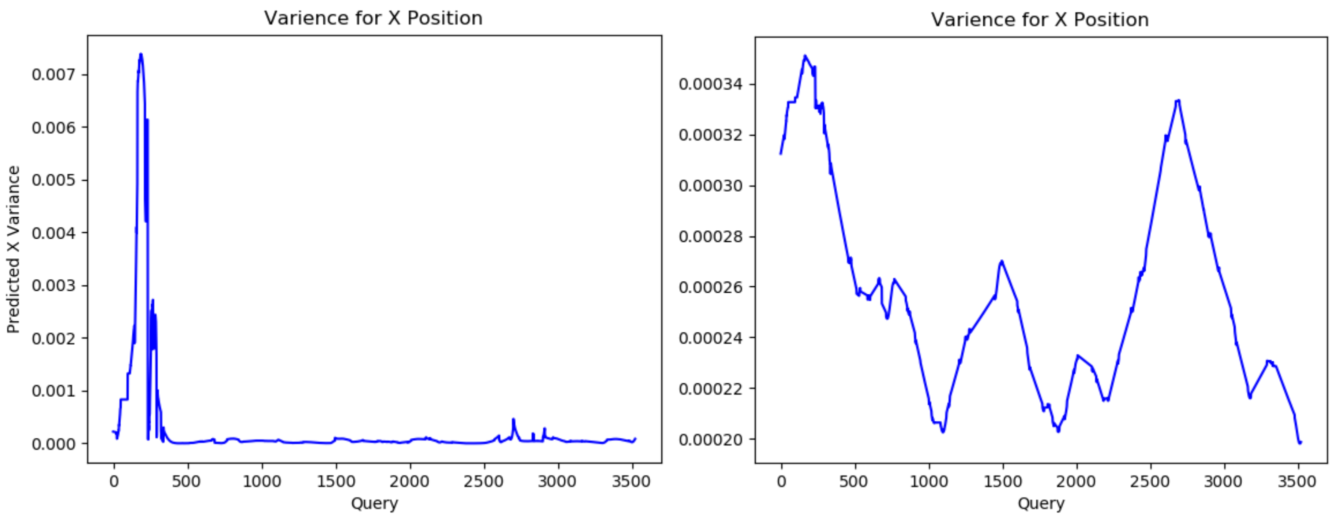


Figure 3. Variance comparison for *h*=1 (left) and *h*=10 (right)

Question 4: Assessment Comparison

The testing results in the figures below compare the performance of the LWLR algorithm to the neural network for the same training and testing dataset for the new learning aim. The total dataset had a 70/30 split between the amounts of training points to testing points. Also Table 2 shows a comparison of the calculated mean squared error across the testing set for each output variable. The MSE was calculated using the equation below:

$$MSE = \sum_{1=1}^{N} \frac{(\hat{y_i} - y_i)^2}{N}$$

Table 3 shows a comparison for the calculated coefficient of determination (R^2) across the testing set for each output variable. R^2 was calculated using the equations shown earlier.

The performance across both algorithms is good with the LWLR performing slightly better along some regions of the test set. Looking at the plot for the new dataset in Figure 1, it seems to imply that the output relative to the six dimensional input space is highly locally linear. Due to this fact, the local linear models fit to the dataset extremely

[3] (Autopilot, 2010)

well producing a highly accurate prediction. The fact that LWLR prediction is made by only considering training data points that are close to the query point, helps explain why there is a slight improvement in the final prediction when compared to the neural network output. The prediction from the network is using weights calculated based on the entire training dataset which may be causing the slight error on some of the sharp corners.

These results are further reinforced by comparing the calculations of MSE and R^2 for both algorithms. The results for LWLR show a slight improvement over the values resulting from the neural network comparisons. It should be noted that the LWLR results for $\sin(\theta_{t+1})$ and $\cos(\theta_{t+1})$ are unrealistic and due to a numerical error in the computation. The variance plots for both output variables are non-zero across the dataset so there is definitely some error, however when calculating the numerator of the SS_{res} equation, the squared difference must result in a very small number such that the computation is being rounded to zero.

Since the learning aims were similar, an approximate comparison was also made between the old and new dataset. Since the datasets vary in the actual points used, this is not a direct comparison. However, both datasets were created from the same trajectory so it was possible to evaluate roughly the same region. These results are shown in Figure 7. This comparison demonstrates the importance of data set formulation before trying to run it through an algorithm.

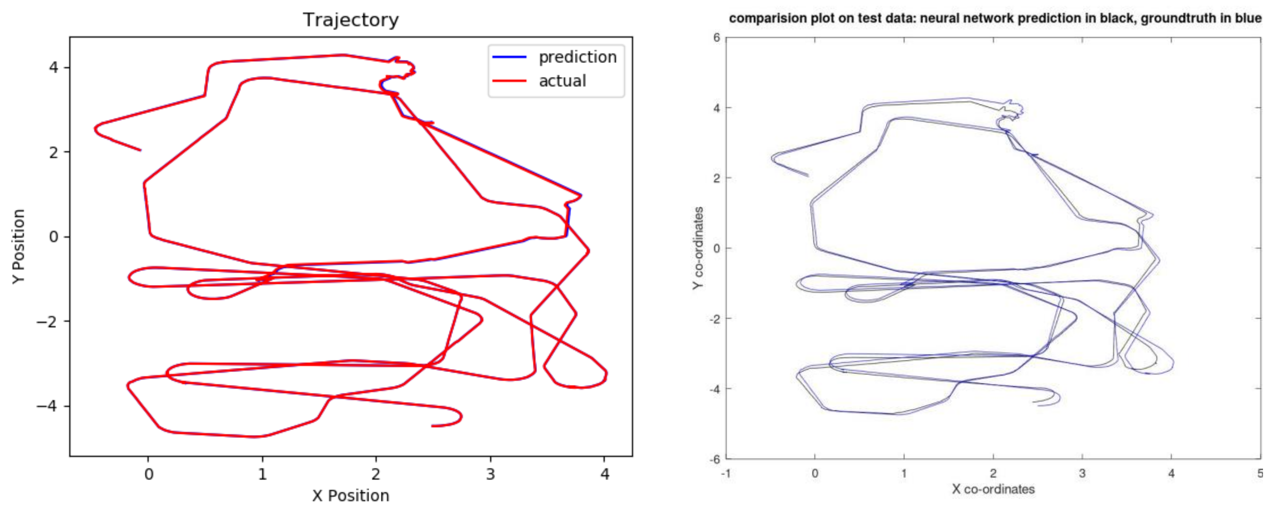


Figure 4. Predicted and expected trajectory comparison (left LWLR, right NN)

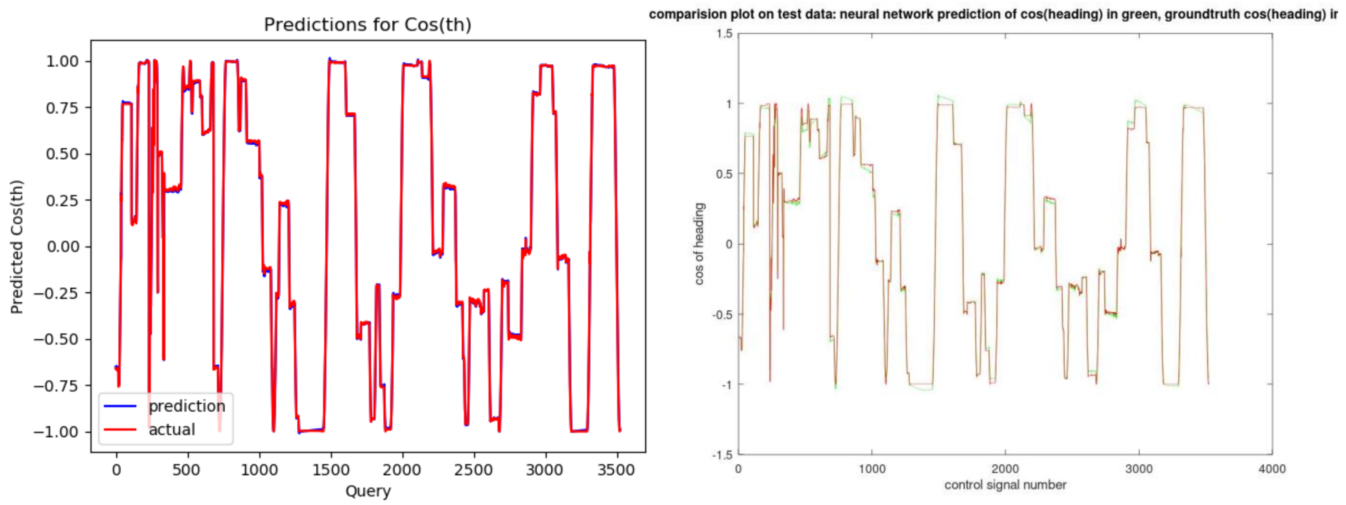
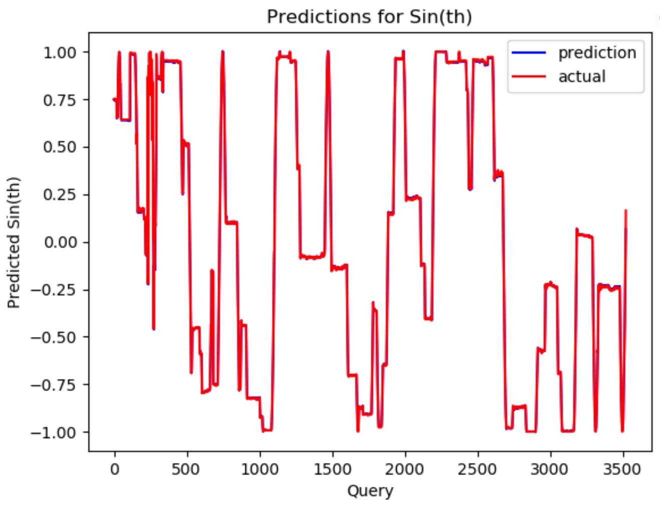


Figure 5. Predicted and expected *cos(θ)* comparison (left LWLR, right NN)



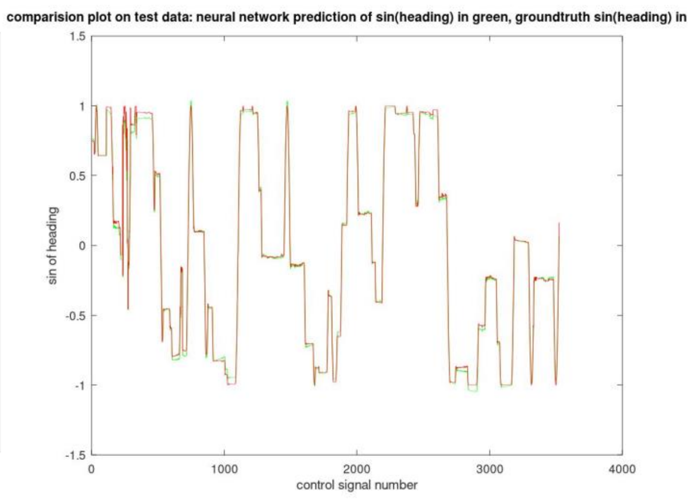


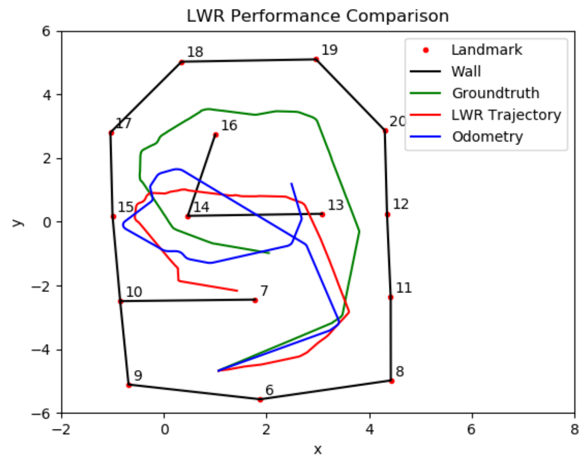
Figure 6. Predicted and expected *sin(θ)* comparison (left LWLR, right NN)

Table 2. MSE Comparison for testing data set on LWLR and neural network

	MSE x_{t+1}	MSE y_{t+1}	MSE $\sin(\theta_{t+1})$	MSE $\cos(\theta_{t+1})$
Neural Network	0.002545	0.0034131	0.0026013	0.0046155
LWLR	0.000337	0.0020741	0.0019773	0.0032508

Table 3. R^2 Comparison for testing data set on LWLR and neural network

	R^2 x_{t+1}	R^2 y_{t+1}	R^2 $\sin(\theta_{t+1})$	R^2 $\cos(\theta_{t+1})$
Neural Network	0.99840	0.99941	0.99462	0.99077
LWLR	0.99978	0.99996	0.99591	0.99350



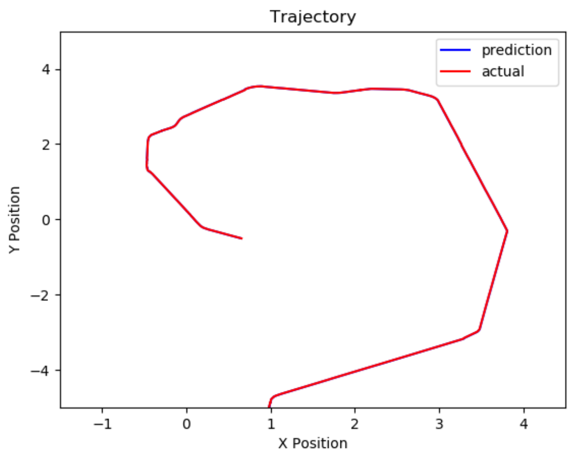


Figure 7. Comparison of the old and new dataset for the LWLR algorithm over roughly the same trajectory

Question 5: Conclusion

For this learning aim and dataset formulation both the LWLR and neural network algorithms perform very well with a slight edge given to LWLR. The LWLR performed slightly better to highly locally linear relationship between the inputs and various outputs, resulting in very accurate locally linear models of for each query point.

These results also demonstrate the variation by using a different dataset formulation. Even though the new and old learning aims were trying to achieve the same goal of replacing the motion model. The new dataset formulation preformed much better than the old dataset. This can be attributed to how the new dataset did a better job of handling angle wrapping as well as creating an input space more suitable to the LWLR algorithm.

The end goal of this experiment would be to use the learning algorithm in place of a filter in order to predict the true state of the robot. While the performance of the LWLR excels for the given training and test set, it may perform poorly if the robot travels into a new area of the input space in a future run. Since the algorithm cannot make a accurate predictions if the query is not captured by the training data, there is a risk that the prediction error will increase greatly and cause unpredictable behavior by the controller. Since the robot is operating in a relatively small area for dataset 1, this may not be an issue and it is also feasible to harvest more data to better cover the input space. However, for a larger or continuous environment the neural network would be the preferred option of the two.

Bibliography

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally Weighted Learning. Kluwer Academic Publishers.

Autopilot. (2010, July 30). *Exponetial Decay*. Retrieved from Wikipedia: https://commons.wikimedia.org/wiki/File:Plot-exponential-decay.svg

Halpern, Y., Lui, H., Barfoot, T., & Leung, K. (July 2011). The UTIAS Multi-Robot Cooperative Localization and Mapping Dataset. *International Journal of Robotics Research*, 30(8):969–974.