Homework 3 - Solutions

October 1, 2017

Solutions

In this homework, you have learned how to implement the analytical solution of the least squares regression algorithm for **prediction**, also known as **Wiener solution.**

In general terms, you are optimizing an adaptive filter (AF) – searching for the optimal set of weights W^* – such that the error is minimized.

There are several different ways to optimize this filter to predict the next point in the time series. You may need to establish a few assumptions before you start computing, namely stationarity. Is the signal you are trying to predict stationary? i.e., are the statistics of the signal not changing over time?

There are different, related (and more technical) definitions of stationarity:

- **Strict Stationarity (SS)** is the strongest form of stationarity. It means that the joint statistical distribution of any collection of the time series variates never depends on time. So, the mean, variance and any moment of any variate is the same whichever variate you choose. However, a time series that occurs in nature are almost always not strict stationarity.
- **Wide-Sense Stationary (WSS).** In general terms, this weaker type of stationary is satisfied when a time series has: i) a constant mean, ii) a constant variance, and iii) an auto-covariance that does not depend on time.
- **Cointegration.** It is possible to consider a weaker form of stationarity still: a series that is first-order stationary which means that the mean is a constant function of time. For example, economists are keen on this kind of stationarity, particularly in how to combine time series with time-varying means to obtain one which is first-order stationary.

In general, a natural phenomenon is usually not stationary. For this homework, however, we are to assume that the time series provided is stationary. This means that we compute only **one** set of filter weights W for the entire training set, per filter order M and regularization term λ.

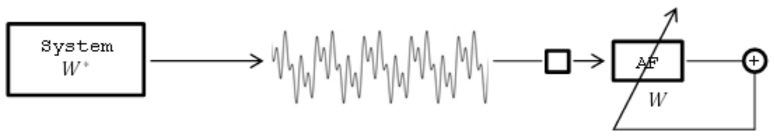


Figure 1: Adaptive filter to model the system class.

If you did not assume stationarity, you would have to divide the time series in windows and compute *local* auto-correlation matrices enough times so that the average filter weights obtained were representative of the time series.

Again, from this point forward, we are **assuming** that the statistics of the signal remain the same across time, that is, the **time series is stationary**.

The steps to filter optimization, validation and testing are:

1. Compute a set of filter weights W for every filter order M and regularization term λ using the training set.

2. Apply every W previously compute to predict the validation set and calculate the mean square error (MSE). This will let us plot a 3-D surface error and locate the set of parameters (M and λ) that minimize the error – let's call it W^*.

3. Predict both not-noisy and noisy test sets using the filter weights W and calculate the error.

To solve step 1, consider filter order $M = 3$, you may find it simpler to organize the data in this way:

$$X = \begin{bmatrix} x(3) & x(4) & x(5) & x(6) & \cdots \\ x(2) & x(3) & x(4) & x(5) & \cdots \\ x(1) & x(2) & x(3) & x(4) & \cdots \end{bmatrix} \tag{1}$$

The first column of X is the set of past points $[x(3), x(2), x(1)]^T$ used to predict the first point in Y, $x(4)$, so that $y(1) = x(4) = \sum_{k=1}^{M} w(k)x(4-k) = w(1) * x(3) + w(2) * x(2) + w(3) * x(1)$.

Observe that:

$$\begin{aligned}
\hat{y}(n) &= W^T X \\
\hat{y}(n) &= \sum_k w(k)x(n-k) \\
\hat{y}(n) &= [w(1)w(2)w(3)] \begin{bmatrix} x(3) & x(4) & x(5) & x(6) & \cdots \\ x(2) & x(3) & x(4) & x(5) & \cdots \\ x(1) & x(2) & x(3) & x(4) & \cdots \end{bmatrix} \\
\hat{y}(n) &= \begin{bmatrix} w(1)x(3) + w(2)x(2) + w(3)x(1) \\ w(1)x(4) + w(2)x(3) + w(3)x(2) \\ w(1)x(5) + w(2)x(4) + w(3)x(3) \\ \cdots \end{bmatrix}^T
\end{aligned} \tag{2}$$

where $\hat{y}$ is the prediction of the **desire response** $Y = [x(4), x(5), x(6), x(7), \ldots]$.

Problem 1

The least squares regression algorithm is a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired and the actual signal).

Optimize an adaptive filter to predict the next point in the time series, using **least square regression** (analytic solution). Since this data is experimental, you may need to use the regularized solution

$$\mathbf{w}^* = (\mathbf{R} + \lambda\mathbf{I})^{-1}\,\mathbf{p}$$

Use the training data set to find your hyper-parameters; the validation data set to validate the hyper parameters (filter order M and regularization term λ) you have introduced in the solution; and the test data set to evaluate how good of approximation your algorithm gives you.

Problem 1.1

(5 points) Select the hyper-parameters for this adaptive filter, namely, filter order and regularization parameter (trained with "training.txt" and validated using "validate.txt"). The idea is to experiment several different values in training and select the one that provides the best mean square error in the validation set. Present a plot of the validation set in the 2-d space of filter order and regularization parameter.

An example pseudo-code to solve this question can be seen in algorithm 2. (Keep in mind that the implementation can be achieved in different ways.) In step 2, we predict the validation set with every set of weights W and calculate its prediction error in the MSE-sense (in the pseudo-code 2, it is denoted as MSE).

For $2 \le M \le 30$ and $0 \le \lambda \le 0.5$ (incrementing by 0.01), the surface error is seen in Figure 2. The MSE is minimized when filter order $M = 30$ and regularization term $\lambda = 0$. The correspondent (trained) weights can be seen in Figure 3 and the prediction error of the validation set is shown in Figure 4 and 5.

```
**Data:** Training set, validation set, set of filter orders $M$, set of regularization terms $\lambda$
$N = length(training)$
$N_v = length(validate)$
for $M \in 1$ to $30$ do
    for $\lambda \in 0$ to $0.5$ do
        $D \leftarrow training(M+1:N)$
        $X \leftarrow zeros(M, N-M)$
        for $i \in 1$ to $N-M$ do
            $X(:,i) \leftarrow training(i+M-1:-1:i)$
        end
        $R \leftarrow XX^T/(N-M)$
        $R \leftarrow R + \lambda * \mathbf{I}$
        $P \leftarrow XD^T/(N-M)$
        $W \leftarrow R^{-1}P$
        for $i \in 1$ to $N_v - M$ do
            $sample = validate(i+M-1:-1:i)$
            $desire\_sample = validate(M+1)$
            $output = \text{sample}^T W$
            $error(i) = desire\_sample - output$
        end
        $MSE(M,\lambda) \leftarrow \frac{1}{N_v-M}\sum error^2$
    end
end
```

Algorithm 1: Wiener Solution.

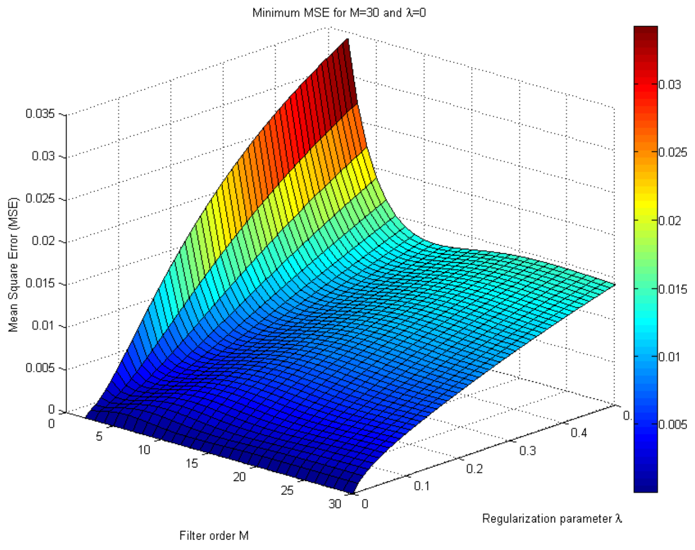


Figure 2: Surface error based on MSE for filter orders $2 \le M \le 30$ and regularization term $0 \le \lambda \le 0.5$ (incrementing by 0.01).

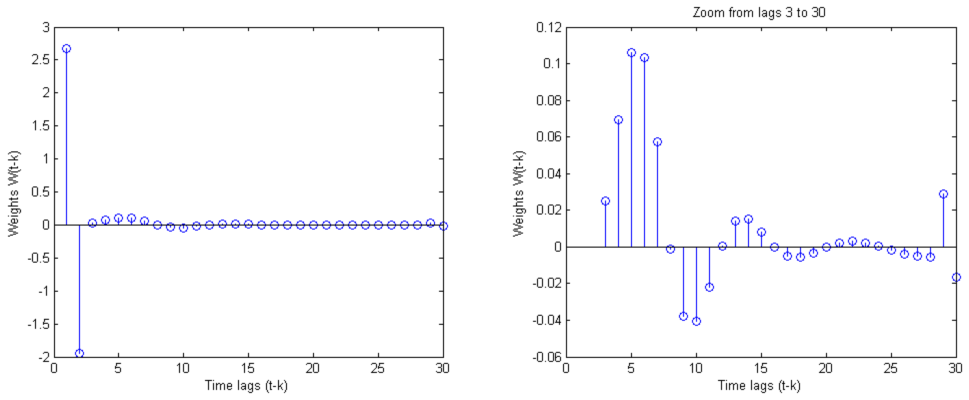


Figure 3: Left: optimal weights for filter order $M = 30$ and regularization term $\lambda = 0$. Right: zoom in to lags 3 to 30.

Problem 1.2

(5 points) Quantify the performance in the MSE-sense of your algorithm using filter orders 4, 8 and 30 in the noisy data ("testnoisy.txt"). Which filter order works better for this noisy data? Explain the results.

To solve this question, we can use the pseudo-code in 2 but apply every W to the noisy test set (instead of the validation set). The prediction error of the noisy set using filter order $M = \{4, 8, 30\}$

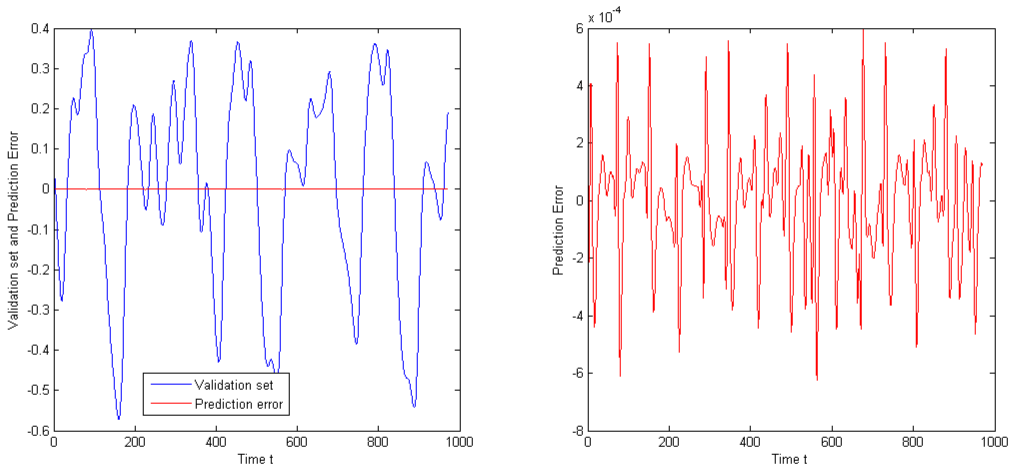


Figure 4: Validation set and its prediction error using the optimal set of weights (filter order $M = 30$ and regularization term $\lambda = 0$).

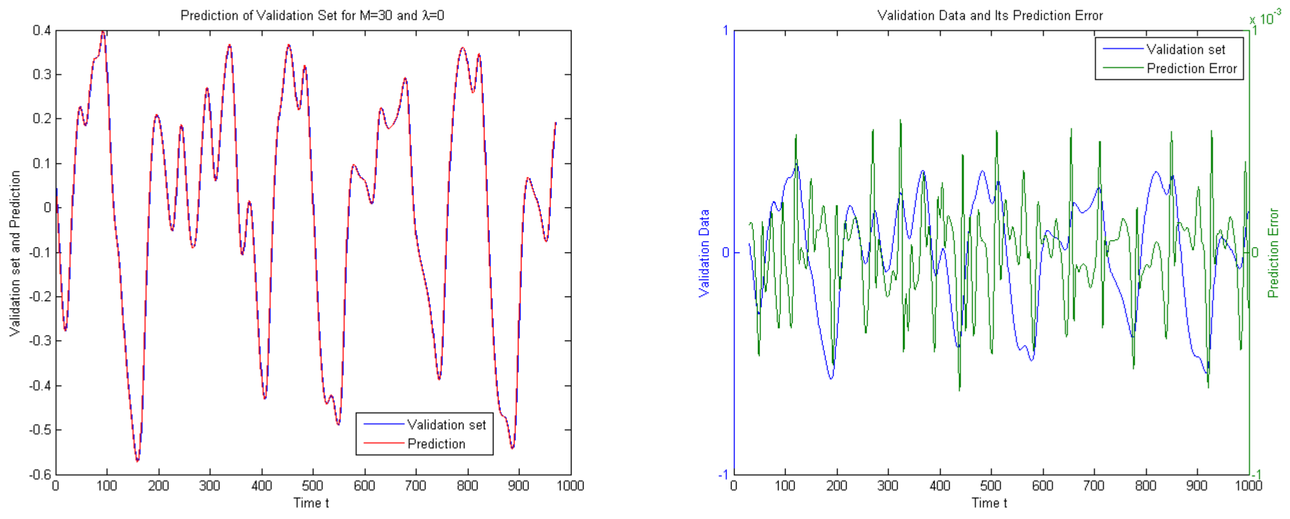


Figure 5: Prediction of the validation set using $M = 30$ and $\lambda = 0$ (left). Prediction error and validation set overlap with different y-axis (right). Main errors ocurr when the gradient changes in the validation data changes.).

are $MSE = \{15.14, 12.64, 12.66\}$ respectively. We can see that these values are much larger than the errors obtained with the validation data.

For the noisy test, the filter order that minimized the error was $M = 8$ which is smaller than the optimal filter order in the test set ($M = 30$). The reason is the filter weights above 8 lags bring in more noise than they help for the prediction. So the order and weights of the filter created without noise may not work very well here.

Problem 2

Optimize an adaptive filter to predict the next point in the time series, using least mean squares (LMS)

$$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta \nabla \mathbf{e}(n)$$

Use the training data set to find your hyper-parameters; the validation data set to validate the hyper parameters (filter order M and step-size η) you have introduced in the solution; and the test data set to evaluate how good of approximation your algorithm gives you.

For stationary signals, the LMS algorithm solution on the parameters (weights w) should converge to the analytic solution, Wiener solution. However, the signals provided in this homework are non-stationary, meaning that statistics of the data change across time. LMS is preferred for non-stationary signals because it can track statistical changes. In addition, the LMS algorithm requires estimation of a new parameter, step size μ. This parameter will dictate how fast we arrive at a solution (minima of the cost function) and how accurate this solution is (misadjustment).

```
Data: Training set, validation set, set of filter orders $M$, set of step sizes $\mu$, initial weight
      vector $W$
$N = length(training)$
$N_v = length(validate)$
for $M \in 2$ to $30$ do
    for $\mu \in 0.001$ to $0.5$ do
        $D \leftarrow training(M+1:N)$
        for $i \in 1$ to $N - M$ do
            $X(i) \leftarrow training(i+M-1:-1:i)$
            $Y(i) \leftarrow W^T X(i)$
            $e(i) \leftarrow D(i) - Y(i)$
            $W \leftarrow W + 2\mu e(i) X(i)$
            $\left(\text{or } W \leftarrow W + 2\mu e(i) \frac{X(i)}{X(i)^T X(i)} \text{ (normalized by the input power)}\right)$
        end
        for $i \in 1$ to $N_v - M$ do
            $sample = validate(i+M-1:-1:i)$
            $desire\_sample = validate(M+1)$
            $output = \text{sample}^T W$
            $error(i) = desire\_sample - output$
        end
        $MSE(M,\mu) \leftarrow \frac{1}{N_v - M} \sum error^2$
        $NMSE(M,\mu) \leftarrow \frac{MSE(M,\mu)}{\frac{1}{N_v} \sum validate^2}$
    end
end
```

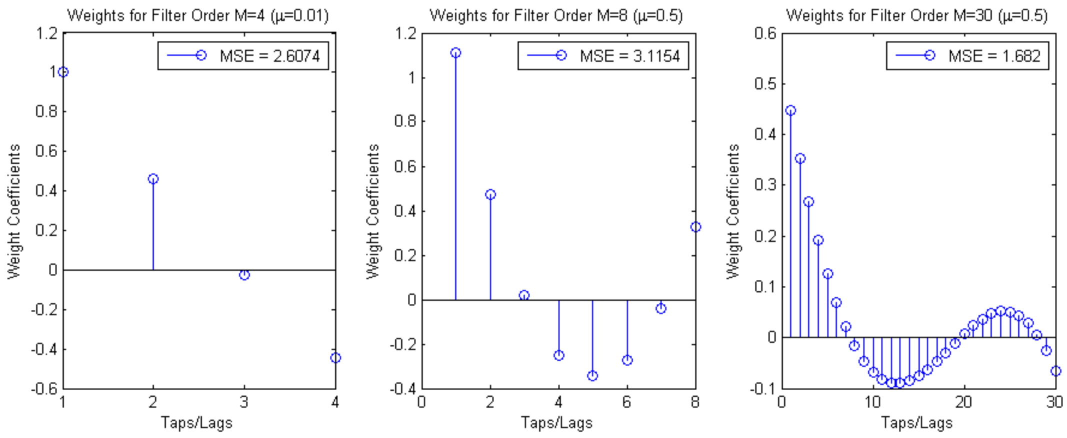
Algorithm 2: LMS Algorithm.

Problem 2.1

(5 points) Select the hyper-parameters for this adaptive filter, namely, filter order and step-size (trained with "training.txt" and validated using "validate.txt"). Quantify the performance in the MSE-sense of your algorithm using filter orders 4, 8 and 30 in the noisy data ("testnoisy.txt"). Which filter order works better for this noisy data? Explain the results.

For $M \in [1, 30]$ and $\mu \in \{0.001, 0.005, 0.01, 0.05, 0.1, 0.5\}$, the minimum MSE was found for filter order $M^* = 2$ and step size $\mu^* = 0.05$.

The performance in the MSE-sense of the LMS algorithm using filter order 4, 8, and 30 in the noisy data can be seen in Figure along with the respective weights.

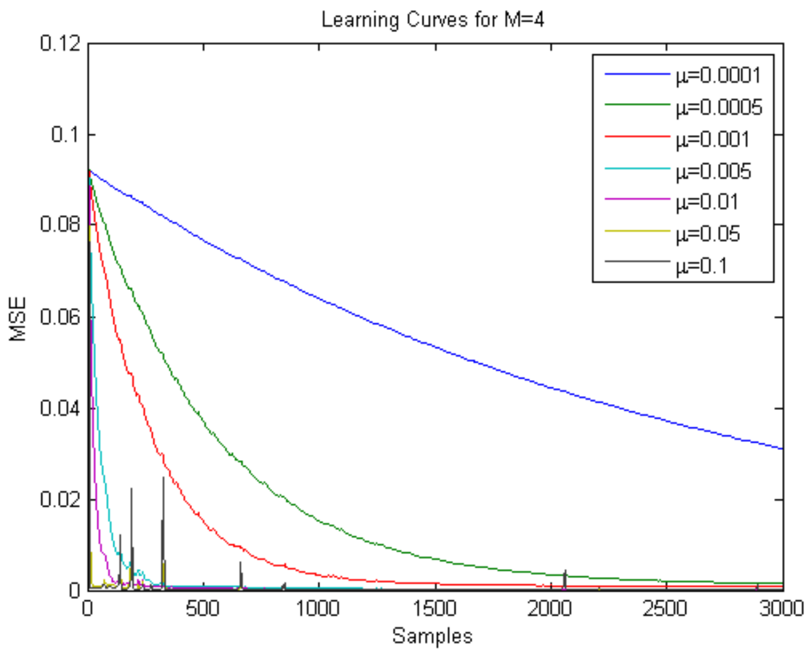


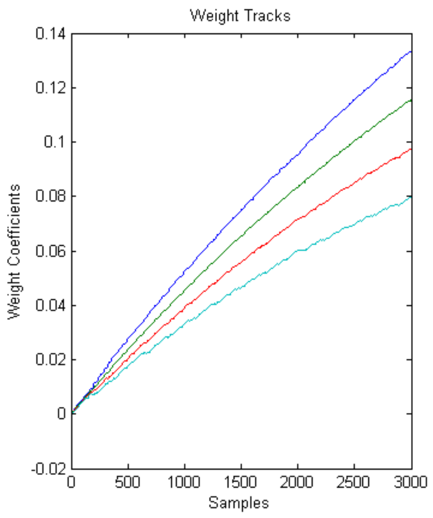
Problem 2.2

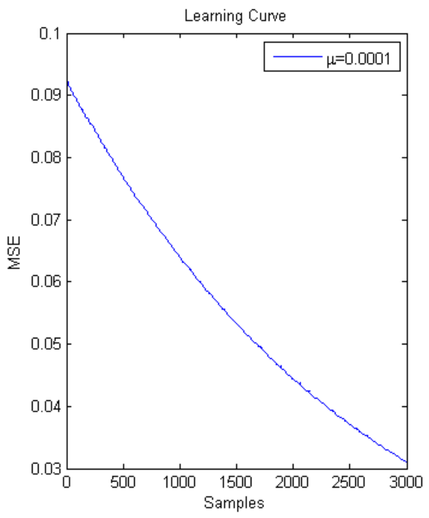
(5 points) Now, choose a fixed filter model order, e.g. $M = 4$, and change the step-size. Plot the learning curves and weight tracks for several step-sizes. What happens when the step-size is too large? What happens when the step-size is too small? What is the compromise that must be made in order to find an *optimal* solution? Explain the results.

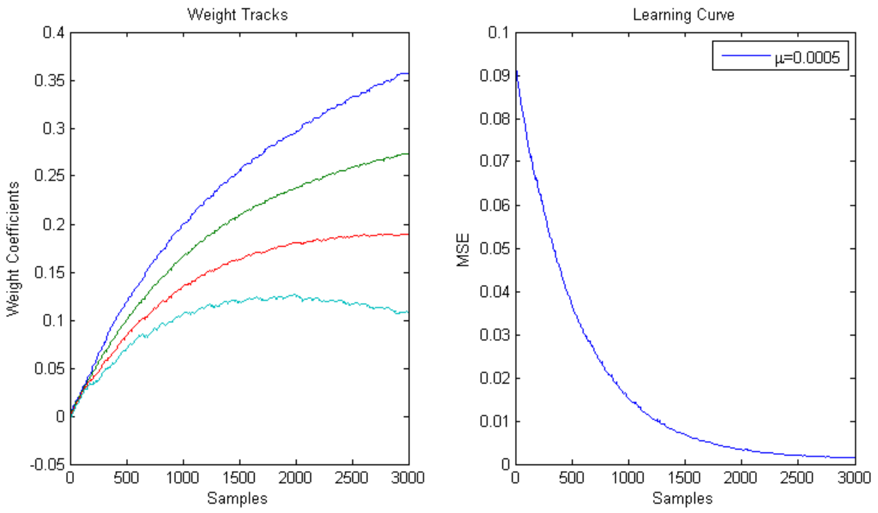
For filter order $M = 4$, and step sizes $\mu = \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$, the learning curves can be seen in Figure . For a small step size, $\mu = 0.0001$, the algorithm takes much longer to adapt the weights. This can be seen from the respectively learning curve because it does show convergence (value at which the MSE stabilizes). For a larger step size, $\mu = 0.01$, the LMS will converge, at first faster along the direction of the eigenvector of the auto-correlation function associated with the largest eigenvalue and then it takes only a few samples to reach the local minimum. For a too large step size, $\mu = 0.1$, the weights go too fast in the direction of the maximum eigenvalue and is bounced back for some samples, as seen in the large spikes of the learning curve.

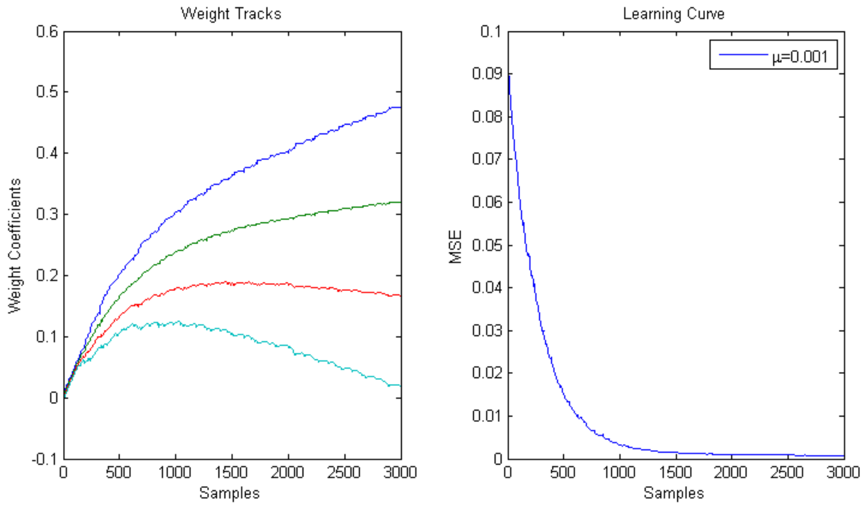
The weight tracks for these step sizes can be seen in Figure - . For this particular order, the preferred step size is $\mu = 0.01$.

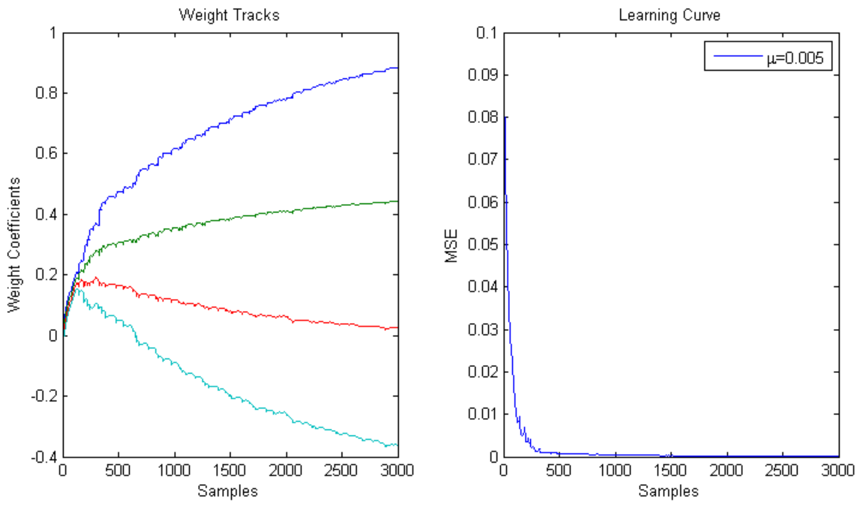

Learning Curves for M=4
MSE
Samples
μ=0.0001
μ=0.0005
μ=0.001
μ=0.005
μ=0.01
μ=0.05
μ=0.1

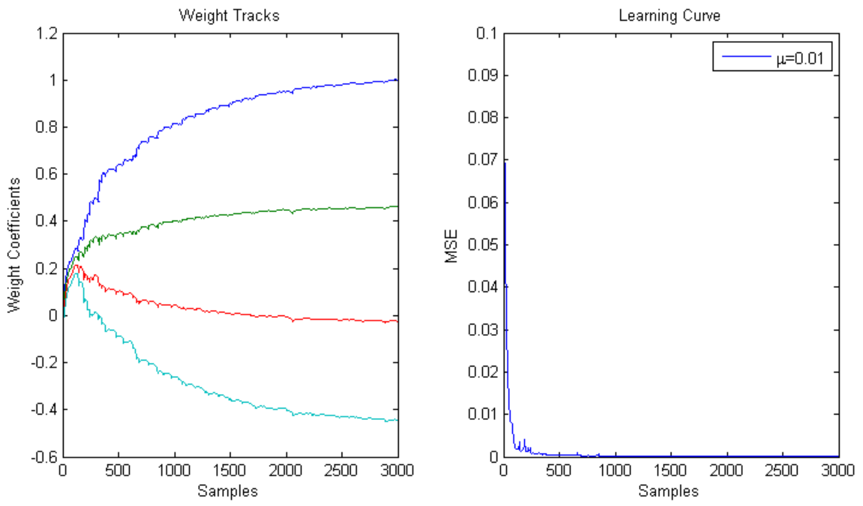

Weight Tracks
Weight Coefficients
Samples

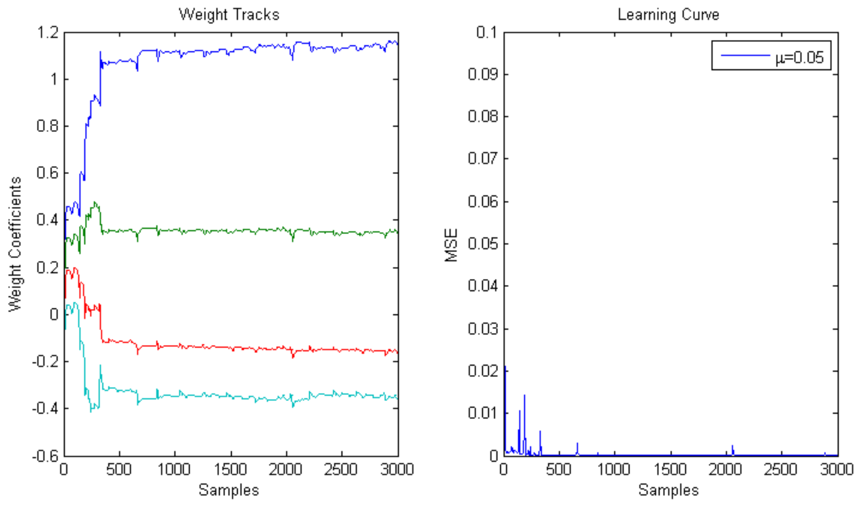

Learning Curve
MSE
Samples
μ=0.0001

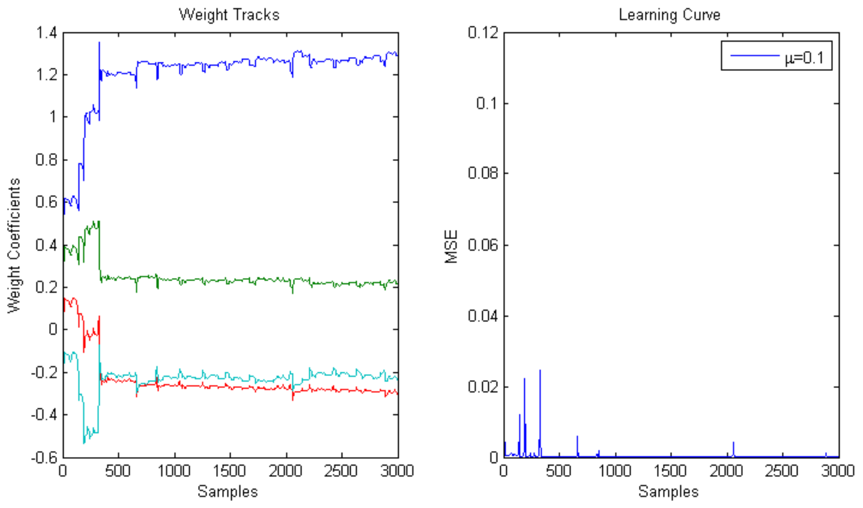

Weight Tracks
Weight Coefficients
Samples
Learning Curve
MSE
Samples
μ=0.0005


Weight Tracks
Weight Coefficients
Samples
Learning Curve
MSE
Samples
μ=0.001


Weight Tracks
Weight Coefficients
Samples
Learning Curve
MSE
Samples
μ=0.005


Weight Tracks
Weight Coefficients
Samples
Learning Curve
MSE
Samples
μ=0.01


Weight Tracks
Weight Coefficients
Samples
Learning Curve
MSE
Samples
μ=0.05


Weight Tracks
Weight Coefficients
Samples
Learning Curve
MSE
Samples
μ=0.1