Homework 2

This homework contains some questions and problems marked with a star. These are intended for graduate students enrolled in the 598NSG section. Undergraduates enrolled in the 498NSU section are welcome to solve them for extra credit. The topics covered in this homework include: DNN complexity estimation in fixed-point, fixed-point inference in Python, back-propagation for non-linear systems, linear prediction in fixed-point, SGD-based linear prediction, SGD with delayed updates. This homework needs to be completed in two parts. Part 1: complete any 3 problems (2 problems for students in 498 section) by 5pm Friday 09/25 and Part 2: the rest (4 problems) by 5pm Friday 10/02. No extension will be provided, so make sure to get started as soon as possible

Problem 1: SQNR of a Dot Product due to Output Quantization

A N-dimensional dot product given by

$$y = \sum_{j=1}^{N} w_j x_j \tag{1}$$

where $x_j \in [-x_m, x_m]$ and $w_j \in [-w_m, w_m]$ are uncorrelated RVs with zero mean and variances of σ_x^2 and σ_w^2, respectively. Derive the analytical expression for the SQNR due to quantization of y given below:

$$\text{SQNR}_{q_y(\text{dB})} = 6B_y + 4.8 - [\zeta_{x(\text{dB})} + \zeta_{w(\text{dB})}] - 10\log_{10}(N) \tag{2}$$

where $\zeta_x = \frac{x_m}{\sigma_x}$ and $\zeta_w = \frac{w_m}{\sigma_w}$.

Solution:

$$\begin{aligned}
\text{SQNR}_{q_y(\text{dB})} &= 6B_y + 4.8 - \zeta_{y(\text{dB})} && (3)\\
&= 6B_y + 4.8 - 20\log_{10}\frac{y_m}{\sigma_y} && (4)\\
&= 6B_y + 4.8 - 20\log_{10}\frac{N x_m w_m}{\sqrt{N}\sigma_x \sigma_w} && (5)\\
&= 6B_y + 4.8 - 20\log_{10}\frac{x_m}{\sigma_x} - 20\log_{10}\frac{w_m}{\sigma_w} - 10\log_{10} N && (6)\\
&= 6B_y + 4.8 - [\zeta_{x(\text{dB})} + \zeta_{w(\text{dB})}] - 10\log_{10}(N) && (7)
\end{aligned}$$

Problem 2: Neural Network Complexity Estimation in Fixed-Point

In this problem, you are asked to once again consider the ResNet-18 and VGG-11 networks discussed in HWK 1 Problem 3. You are always encouraged to write scripts to solve the problem. What are the representational and computational costs of both networks assuming a uniform precision assignment of 10 bits?

Solution: In order to solve this problem, it is important to remember the definition of the costs. The total representational cost is given by:

$$\sum_{l=1}^{L} (R_{W_l} B_{W_l} + R_{A_l} B_{A_l})$$

where R_{W_l}, R_{A_l}, B_{W_l}, B_{A_l} are the number of weights, number of activations, precision of weights, precision of activations, at layer l, respectively. Similarly, the total computational cost is given by:

$$\sum_{l=1}^{L} N_l \left(D_l B_{A_l} B_{W_l} + (D_l - 1)(B_{A_l} + B_{W_l} + \log_2 D_l - 1)\right)$$

where N_l is the number of dot products and D_l is the dot product length at layer l, respectively. Applying these formulas to the cases of ResNet-18 and VGG-11 (using results from HWK1), we obtain:
ResNet-18 Representational Cost: 1.386×10^8 bits.
ResNet-18 Computational Cost 2.377×10^{11} FAs.
VGG-11 Representational Cost: 1.359×10^9 bits.
VGG-11 Computational Cost 1.004×10^{12} FAs.

Problem 3: Fixed-Point Inference with Python

In this problem, we will perform fixed-point inference using the CIFAR-10 network we had prepared in Homework 1 Problem 4. In the newly provided zip file, you will find source code from the last homework along with additional files for dynamic ranges and precisions. These new files contain data that may be helpful to complete this problem.

1. Determine the minimum uniform precision requirement needed to maintain a test accuracy within 1% of the floating-point baseline. Note: in order to quantize weights and activations, you need to set their dynamic ranges, these can be found in the file provided on the course website. (HINT: activations are unsigned and weights are signed.)
   Solution: Running a quantized inference yields a minimum precision of 7 bits needed to maintain the accuracy to within 1% compared to the full precision baseline. The corresponding test accuracy obtained is 85.87%. (original FL baseline: 86.24%)

2. Using the *precision offsets* from the provided file on the course website, determine the minimum per-layer precision requirements needed to maintain a test accuracy within 1% of the floating-point baseline.
   Solution: Running a quantized inference with the provided precision offsets yields a test accuracy with 1% of the full precision baseline when the base precision is at least $B_{\min} = 5$. The corresponding accuracy obtained is 85.96%.

   Note: the per-layer precision assignments were obtained via the noise equalization method we have discussed in class.

3. Compare the corresponding representational and computational costs of precision assignments in Part 1 and 2.
   Solution:
   Uniform Quantization Representational Cost: 2.517×10^8bits
   Uniform Quantization Computational Cost: 1.85×10^{10} FAs
   Noise-gain Quantization Representational Cost: 1.916×10^8 bits
   Noise-gain Quantization Computational Cost: 2.84×10^{10} FAs

The noise-gain method has lower representational cost than that of uniform precision assignment but it also has a higher computational cost.

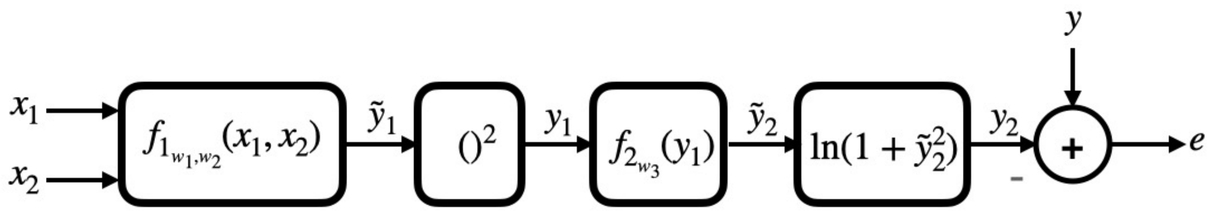


Figure 1: The non-linear system for Problem 4

Problem 4: Back-propagation for Non-linear System

Consider the non-linear system in Figure 1 where the goal is to predict the label y given an input (x_1, x_2). The system is parametrized by three weights (w_1, w_2, w_3). The following are the equations describing the feedforward computations of the systems:

$$\tilde{y}_1 = w_1 x_1 + w_2 x_2$$

$$y_1 = \tilde{y}_1^2$$

$$\tilde{y}_2 = w_3 y_1$$

$$y_2 = \ln(1 + \tilde{y_2}^2)$$

It is desired to minimize the mean-squared error

$$C = \mathbb{E}[e^2] = \mathbb{E}[(y - y_2)^2]$$

Assuming (x_1, x_2) are coordinates uniformly distributed over the unit circle, i.e. $x_1^2 + x_2^2 = 1$. The system is trying to predict its angle $y \in [-\pi, \pi]$ where

$$y = \begin{cases} \arctan(\frac{x_2}{x_1}), & x_1 > 0 \\ \pi + \arctan(\frac{x_2}{x_1}) & x_1 < 0 \text{ and } x_2 > 0 \\ -\pi + \arctan(\frac{x_2}{x_1}) & x_1 < 0 \text{ and } x_2 < 0 \\ \pi/2 & x_1 = 0 \text{ and } x_2 > 0 \\ -\pi/2 & x_1 = 0 \text{ and } x_2 < 0 \end{cases}$$

1. Derive expressions for the following three gradients: $\frac{\partial C}{\partial w_1}$, $\frac{\partial C}{\partial w_2}$ and $\frac{\partial C}{\partial w_3}$. Simplify as much as you can. (HINT: use the chain rule and LOTUS).

Solution:

$$\begin{aligned}\frac{\partial}{\partial w_1}\mathbb{E}[e^2] &= \mathbb{E}[\frac{\partial}{\partial w_1}(y-y_2)^2] \\ &= \mathbb{E}[-2(y-y_2)\frac{\partial y_2}{\partial \tilde{y_2}}\frac{\partial \tilde{y_2}}{\partial y_1}\frac{\partial y_1}{\partial \tilde{y_1}}\frac{\partial y_1}{\partial \tilde{w_1}}] \\ &= \mathbb{E}[-8(y-y_2)\frac{\tilde{y_2}w_3\tilde{y_1}x_1}{1+\tilde{y_2}^2}]\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial w_2}\mathbb{E}[e^2] &= \mathbb{E}[\frac{\partial}{\partial w_2}(y-y_2)^2] \\ &= \mathbb{E}[-2(y-y_2)\frac{\partial y_2}{\partial \tilde{y_2}}\frac{\partial \tilde{y_2}}{\partial y_1}\frac{\partial y_1}{\partial \tilde{y_1}}\frac{\partial y_1}{\partial \tilde{w_1}}] \\ &= \mathbb{E}[-8(y-y_2)\frac{\tilde{y_2}w_3\tilde{y_1}x_2}{1+\tilde{y_2}^2}]\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial w_3}\mathbb{E}[e^2] &= \mathbb{E}[\frac{\partial}{\partial w_3}(y-y_2)^2] \\ &= \mathbb{E}[-2(y-y_2)\frac{\partial y_2}{\partial \tilde{y_2}}\frac{\partial \tilde{y_2}}{\partial w_3}] \\ &= \mathbb{E}[-4(y-y_2)\frac{\tilde{y_2}y_1}{1+\tilde{y_2}^2}]\end{aligned}$$

Since (x_1, x_2) are points uniformly distributed on the unit circle, we can define a uniform random variable Θ over $[-\pi, \pi]$. Then we have $(x_1, x_2) = (\cos\Theta, \sin\Theta)$ and $y = \Theta$. Moreover we can also make the substitutions for other variables in the equation: $\tilde{y_1} = (w_1\cos\Theta + w_2\sin\Theta)$, $y_1 = (w_1\cos\Theta + w_2\sin\Theta)^2$, $\tilde{y_2} = w_3(w_1\cos\Theta + w_2\sin\Theta)^2$. By LOTUS, we have:

$$\mathbb{E}[f(\Theta)] = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(\theta)d\theta$$

Thus:

$$\frac{\partial}{\partial w_1}\mathbb{E}[e^2] = \mathbb{E}[-8(y-y_2)\frac{\tilde{y_2}w_3\tilde{y_1}x_1}{1+\tilde{y_2}^2}] = \frac{-4}{\pi}\int_{-\pi}^{\pi}(y-y_2)\frac{\tilde{y_2}w_3\tilde{y_1}x_1}{1+\tilde{y_2}^2}d\theta \tag{8}$$

$$\frac{\partial}{\partial w_2}\mathbb{E}[e^2] = \mathbb{E}[-8(y-y_2)\frac{\tilde{y_2}w_3\tilde{y_1}x_2}{1+\tilde{y_2}^2}] = \frac{-4}{\pi}\int_{-\pi}^{\pi}(y-y_2)\frac{\tilde{y_2}w_3\tilde{y_1}x_2}{1+\tilde{y_2}^2}d\theta \tag{9}$$

$$\frac{\partial}{\partial w_3}\mathbb{E}[e^2] = \mathbb{E}[-4(y-y_2)\frac{\tilde{y_2}y_1}{1+\tilde{y_2}^2}] = \frac{-2}{\pi}\int_{-\pi}^{\pi}(y-y_2)\frac{\tilde{y_2}y_1}{1+\tilde{y_2}^2}d\theta \tag{10}$$

2. Derive closed form expressions for the following three instantaneous gradients: $\frac{\partial e^2}{\partial w_1}$, $\frac{\partial e^2}{\partial w_2}$ and $\frac{\partial e^2}{\partial w_3}$. **Solution:**

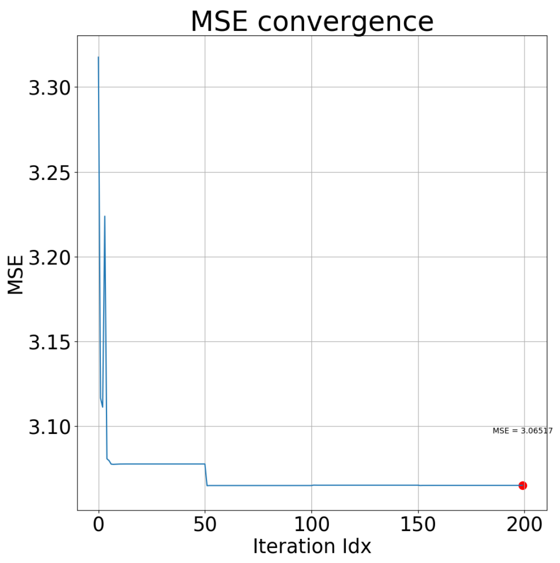
$$\frac{\partial}{\partial w_1} e^2 = -8(y - y_2)\frac{\tilde{y_2} w_3 \tilde{y_1} x_1}{1 + \tilde{y_2}^2}$$

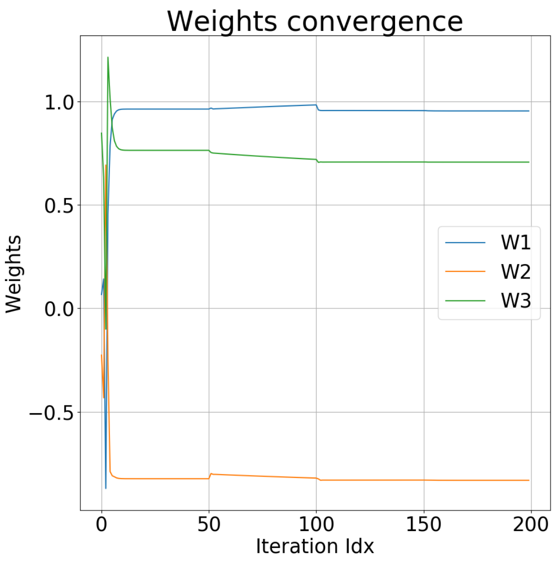
$$\frac{\partial}{\partial w_2} e^2 = -8(y - y_2)\frac{\tilde{y_2} w_3 \tilde{y_1} x_2}{1 + \tilde{y_2}^2}$$

$$\frac{\partial}{\partial w_3} e^2 = -4(y - y_2)\frac{\tilde{y_2} y_1}{1 + \tilde{y_2}^2}$$

3. Using derivation in part 2, implement the SGD algorithm applied to this setup. Plot the convergence curves and the evolutions of the weights (w_1, w_2, w_3).

   Solution: We implement the SGD algorithm to this system. We use the expressions for the derivatives obtained above for the weight updates. Here are the convergence curves:





   Final testset error settles at 3.065. Evaluating C yields 3.063. To match the average MSE with the sample MSE, 10000 test samples and 50000 training samples are used.
   The weights settle at $(0.9547247, -0.8316406, 0.7071791)$.

4. * Analytically determine the number of bits B_{w1} needed to quantize w_1 such that the MSE C increases by no more than 1% over its floating-point value reached in Part 3. Assume that the quantization noise appears in an additive form at the output. [Hint: use the expression for the gradient from Part 1 as the quantization noise gain for w_1.]

   Solution: Evaluating (8) with the final converged weights yielding 0.0163. By linear approximation, quantization noise caused on B_{w_1} will cause its noise on MSE to be amplified by $0.0163\times$. Our noise budget is $3.065 \times 0.01 = 0.03065$ and we need to find the minimum B such that the difference between quantized w_1 and the original w_1 is below the noise budget. Here we are adopting the perturbation model. Thus Δw_1 (parametrizable by B_{w_1} added to w_1 will cause a ΔC at the output with $\Delta C = \frac{\partial C}{\partial w_1}\Delta w_1$. Evaluating this criteria gives $B = 4$

Note: depending on the final learning result, Δw_1 could also decrease the final MSE. If the new MSE does not differ from the old MSE by 1%, the response will be marked as correct.

5. * Validate your answer in Part 4 by running the nonlinear system with w_1 quantized to B_{w1} bits.
   Solution: Re-evaluating C yields 3.068 with $B = 4$ after quantization of w_1, which is under the 1% threshold line. If enough test samples are used, the sample MSE will also be under the 1% threshold line.

6. This system is a non-convex system and therefore has many local minima. To see this, can you find a solution that is different from the one in Part 3?
   Solution: By observing the nonlinear system, we can see that the flipping the sign of w_3 will not change the predictor output. Thus $(w_1, w_2, -w_3)$ of the one found in part 3 is also a solution.

Problem 5: Fixed-Point Linear Predictor *

Calculate the minimum precision of the input and coefficient for the 3-tap predictor designed in Homework 1 Problem 5 such that the fixed-point model achieves an SNR that is within $0.1dB$ of the floating-point SNR (=$10\log_{10}\frac{\sigma_x^2}{\sigma_e^2}$ where σ_x^2 are the variances of the input and error, respectively). Estimate the computational cost of this precision assignment. Validate your answer by simulating the fixed-point and floating-point predictors.

Solution:

$$10\log_{10}\frac{\sigma_y^2}{\sigma_e^2} - 10\log_{10}\frac{\sigma_y^2}{\sigma_{qy}^2 + \sigma_e^2} = 0.1 \tag{11}$$

$$10\log_{10}\frac{\sigma_{qy}^2 + \sigma_e^2}{\sigma_e^2} = 0.1 \tag{12}$$

$$10\log_{10}\frac{\sigma_{qy}^2 + \sigma_e^2}{\sigma_e^2} = 0.1 \tag{13}$$

$$\frac{\sigma_{qy}^2}{\sigma_e^2} = 10^{0.01} - 1 \tag{14}$$

$$\sigma_{qy}^2 = (10^{0.01} - 1) \times \sigma_e^2 = 0.352 \times (10^{0.01} - 1) = 0.008199 \tag{15}$$

where σ_{qy}^2 is the power of the quantization noise and σ_e^2 is the MSE we found in the previous homework. From (15), we can see that the total quantization noise can be as much as 0.008119 such that the new total SNR is within 0.1 dB of the original one.

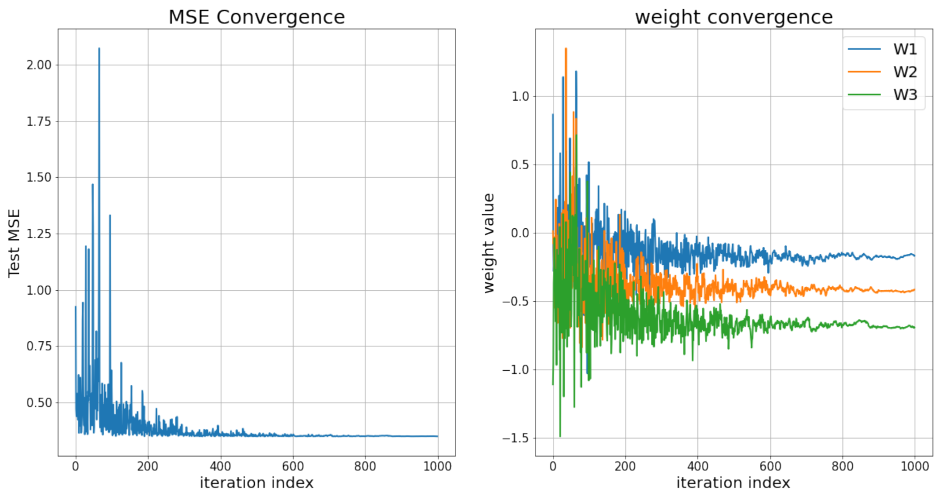
The total quantization noise power is equal to $\|\mathbf{w}\|^2 \frac{\Delta_x^2}{12} + N\mathbb{E}\left[x^2\right] \frac{\Delta_w^2}{12} = 0.0552\Delta_x^2 + 0.13\Delta_w^2$
There are multiple ways to answer the question now, but for simplicity, let us assume that input and weight precision are equal. We equate the quantization noise to our budget and obtain $(0.0552 + 0.13)\Delta^2 < 0.008199 \Rightarrow B > 3.24$. Thus the precision required is of 4 bits.
As this is a three tap filter, we get a 1-bit bit growth during the accumulation. Thus, the computational cost is of $N \times B^2 + (N-1) \times (2B+1) = 62$ FAs. One can simulate the above results using the quantization Notebook provided in class.

Problem 6: SGD-based Linear Predictor for Time Series

Code a 3-tap SGD-based linear predictor for predicting x_n from Homework 1 Problem 5. The coefficients of this online predictor should approach the optimal ones previously determined.

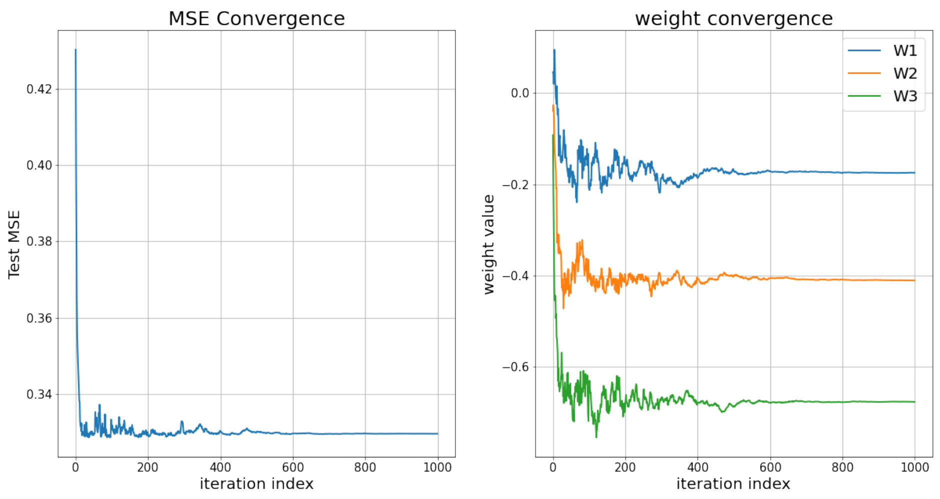
1. Calculate the maximum value of the learning rate μ_m (stability bounds) for which convergence is guaranteed.
Solution: The stability bound is
$$\mu_m < \frac{2}{N \times \sigma_x^2} = \frac{2}{3 \times 0.52} = 1.28$$

2. Plot the convergence behavior of the MSE and coefficients for $\mu = 0.5\mu_m$. Remember to use ensemble average plots. Compares these with the MSE and coefficients obtained in Homework 1 Problem 5. Repeat this part for $\mu = 0.01\mu_m$ and $\mu = 50\mu_m$.
Solution: We first run with $\mu = 0.5\mu_m$. Here are the convergence curves:



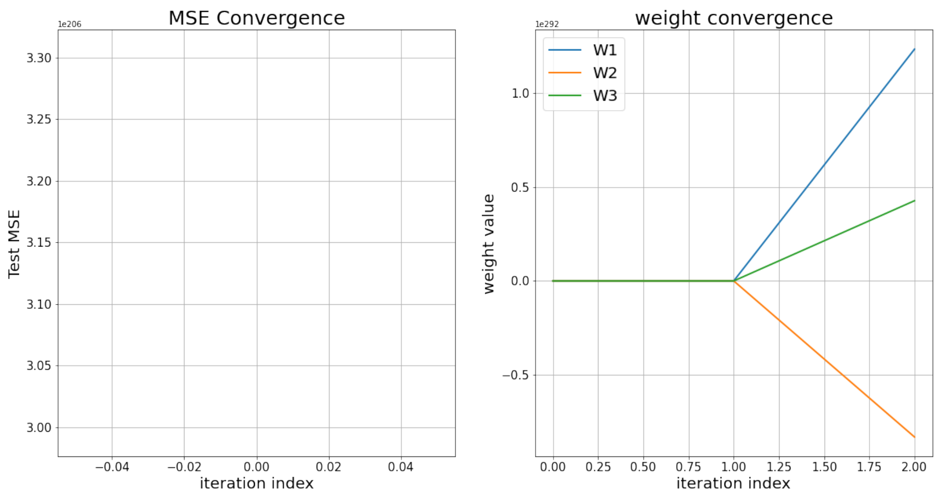
As can be seen, the weights converge towards the theoretical solution that was obtained in HWK1, but unstable at the beginning; the MSE increases dramatically and decreases at the end.

Next, we use $\mu = 0.01\mu_m$. Here are the convergence curves:



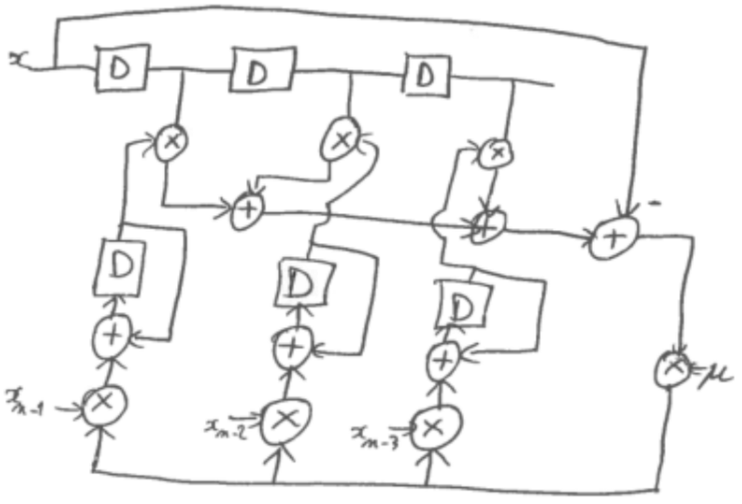
Again, the weights converge towards the theoretical solution that was obtained in HWK1. The convergence is very stable at all

Finally, we use $\mu = 50\mu_m$. Here are the convergence curves:



This learning rate leads to instability and the error is off the chart.

3. Sketch an architecture and calculate its critical path delay, latency, and throughput in terms of the adder delay (T_A) and the multiplier delay (T_M).
Solution: Here is the sketch (pardon my lack of artistic skills):



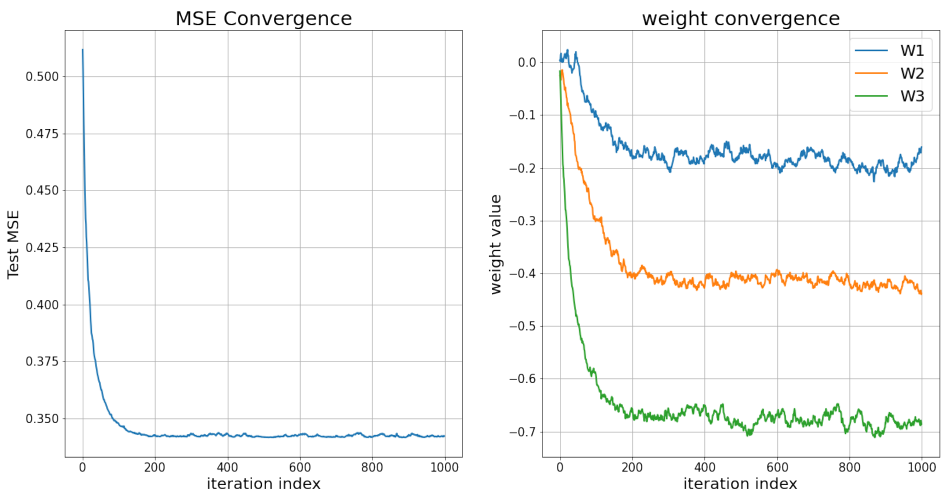
In what follows, we do account for the cost of multiplication with the learning rate. It is also correct to assume it to be negligible (shift operation).
Critical path: $3T_M + 4T_A$
Latency: $T_M + 2T_A$
Throughput: $\frac{1}{3T_M+4T_A}$

Problem 7: SGD with Delayed Updates *

The conventional SGD algorithm can be modified by delaying the weight vector update by M_1 iterations as follows:

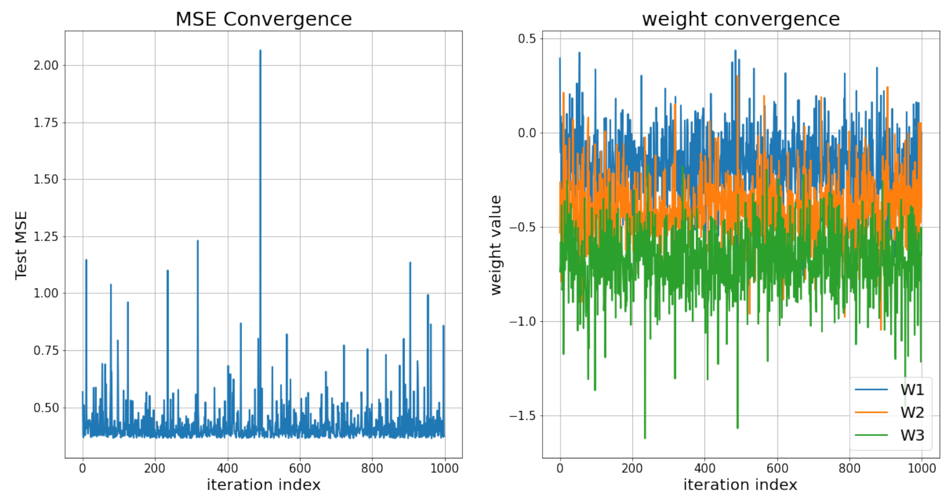
$$\mathbf{w}_n \leftarrow \mathbf{w}_{n-1} + \mu \nabla_{\mathbf{w}} \left(e^2_{n-M_1} \right)$$

1. Code a 3-tap SGD-based linear predictor for predicting x_n as in Homework 1 Problem 5. Assume $M_1 = 5$. Plot its convergence behavior of the coefficients.
Solution: Since $\mu = 0.001$ worked quite well in the case of no delays (previous problem), we use this learning rate and code the LMS with delayed updates using $M1 = 5$. Here are the convergence curves:



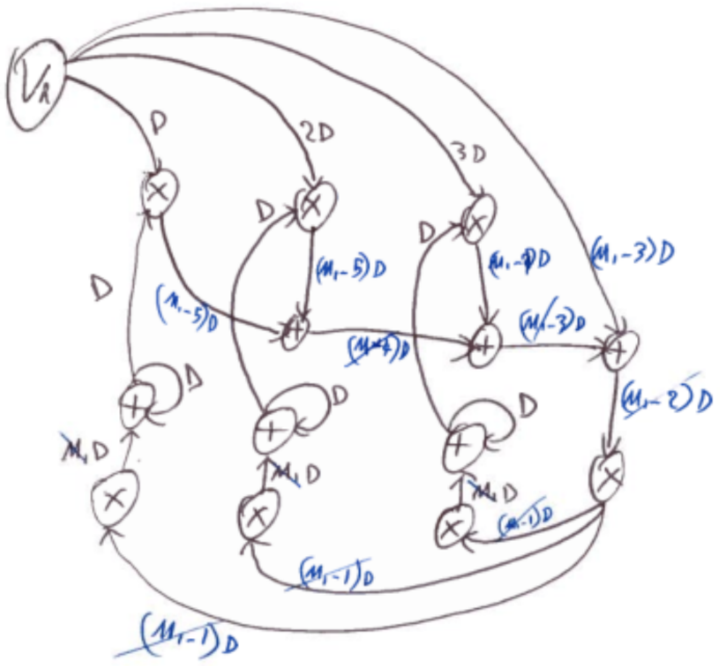
As can be seen, the LMS algorithm still converges well in spite of the delayed updates.

2. Sweep the learning rate μ to see if you can find a case where the original SGD algorithm converges but the delayed one does not.
Solution: The delayed LMS algorithm fails to converge when $\mu = 0.1$ as shown in the below convergence curves. This was well within the stability bound of the conventional LMS (no delays).



3. Determine the minimum value of M_1 such that the critical path delay of the pipelined architecture is equal to one multiplier delay, i.e. $T_{cp} = T_M$. Assume $T_M > T_A$. Draw the pipelined architecture.
Solution: We can use the data flow graph instead of the architecture for a change (note this will probably be useful for you guys in HWK3 so I am using it instead). We assign M_1

delays on the update and redistribute those across all computations. As can be seen, we achieve full pipelining when $M_1 - 5 = 1 \Rightarrow M_1 = 6$.


V_k
D
2D
3D
D
D
D
$(M_1-5)D$
$(M_1-5)D$
$(M_1-4)D$
$(M_1-4)D$
$(M_1-3)D$
$(M_1-3)D$
$(M_1-2)D$
D
D
D
M_1D
M_1D
M_1D
$(M_1-1)D$
$(M_1-1)D$
$(M_1-1)D$