HW3 REPORT

SGD with momentum

Adding momentum to SGD is like adding inertia (mass) to the gradient descent process. The momentum carries history information and helps prevent the fluctuaion of stocastic gradient.

With $\mu < 1$, the old information of gradient will decay exponentially, so the weight of recent gradients is larger than old ones.

I use the following fomular:

$$\begin{aligned} m_t &= \mu * m_{t-1} + (1-\mu) * \eta * g_t \\ p_t &= p_{t-1} - m_t \end{aligned}$$

where η is the learning rate.

Notice that I apply an additional $(1-\mu)$ on the gradient g_t. To get the same effect with the usual version

$$\begin{aligned} m_t &= \mu * m_{t-1} + g_t \\ p_t &= p_{t-1} - \eta' * m_t \end{aligned}$$

we need to set $\eta = \eta'/(1-\mu)$

Adam

Adam algorithm uses the second moment of gradient to control the step size. In addition to the momentum, we also need to store the square of gradient for each learnable parameter.

I use the following fomular:

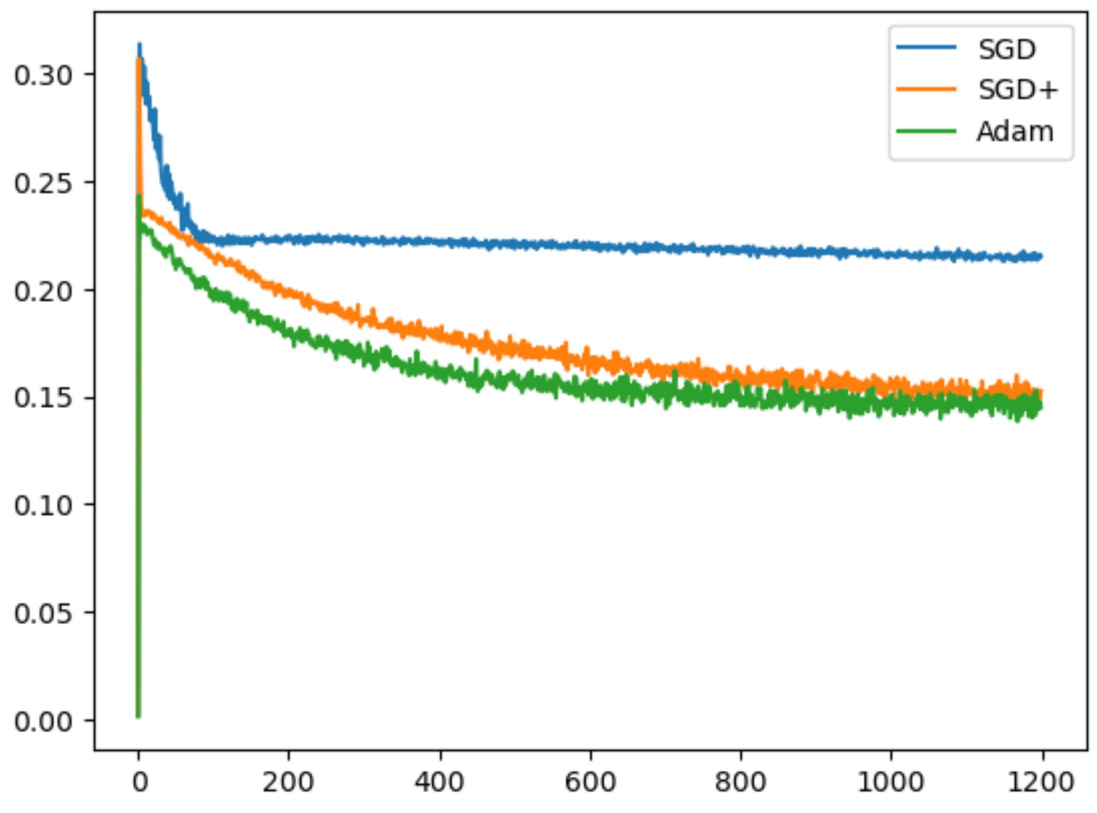
$$\begin{aligned} m_t &= \beta_1 * m_{t-1} + (1-\beta_1) * g_t \\ v_t &= \beta_2 * v_{t-1} + (1-\beta_2) * g_t^2 \\ p_t &= p_{t-1} - \eta * \frac{m_t}{\sqrt{v_t + \epsilon}} \end{aligned}$$

where η is the learning rate. All operations are element-wise for vector m_t, v_t and g_t.

Results

One-neuron model

Figure 1: comparison of three algorithms with $\eta = 1 \times 10^{-3}$

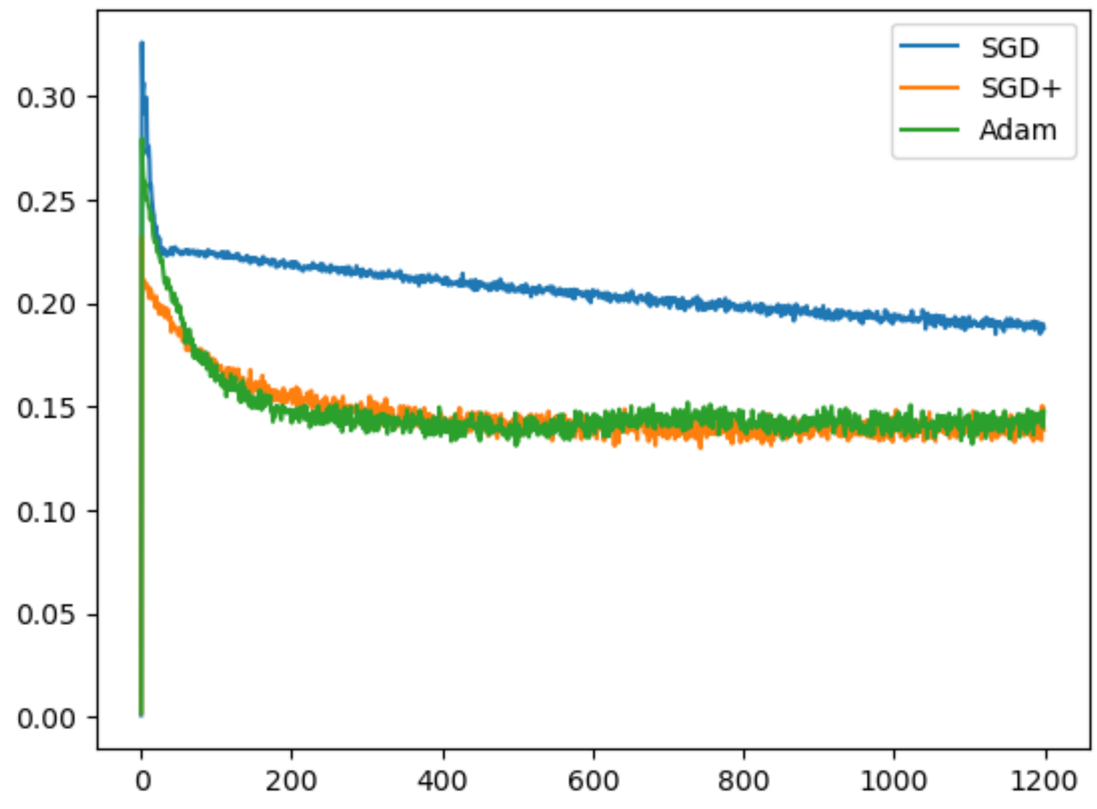


Other parameters are the same as below:

```
mp3 = ModifiedPrimer(
        one_neuron_model = True,
        expressions = ['xw=ab*xa+bc*xb+cd*xc+ac*xd'],
        output_vars = ['xw'],
        dataset_size = 5000,
        learning_rate = 1e-3,
        rate_mu = 0.95      ## for SGD+
        rate_beta1 = 0.9,   ## for Adam
        rate_beta2 = 0.99,  ## for Adam
        training_iterations = 240000,
        batch_size = 8,
        display_loss_how_often = 200,
        debug = True,
)
```

Now I change learning rate $\eta = 4 \times 10^{-3}$.

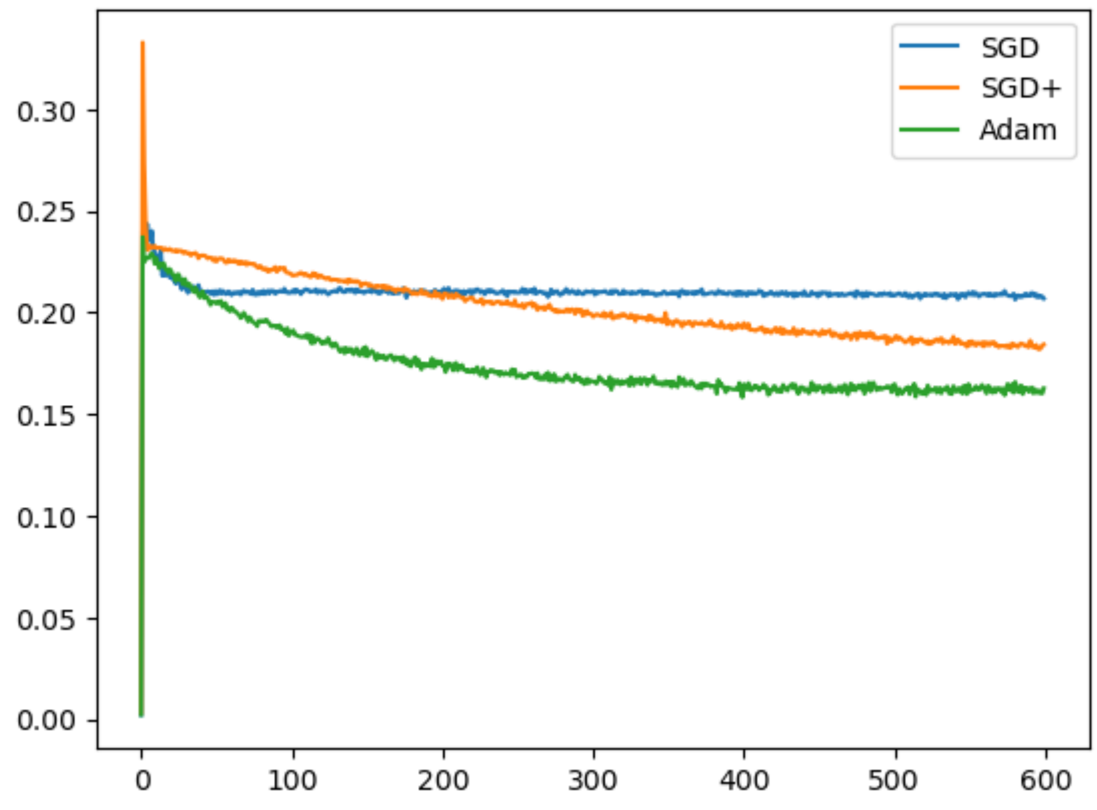
Figure 2:



I also want to make the batch size larger.

By setting `batch_size = 8` I get

Figure 3:



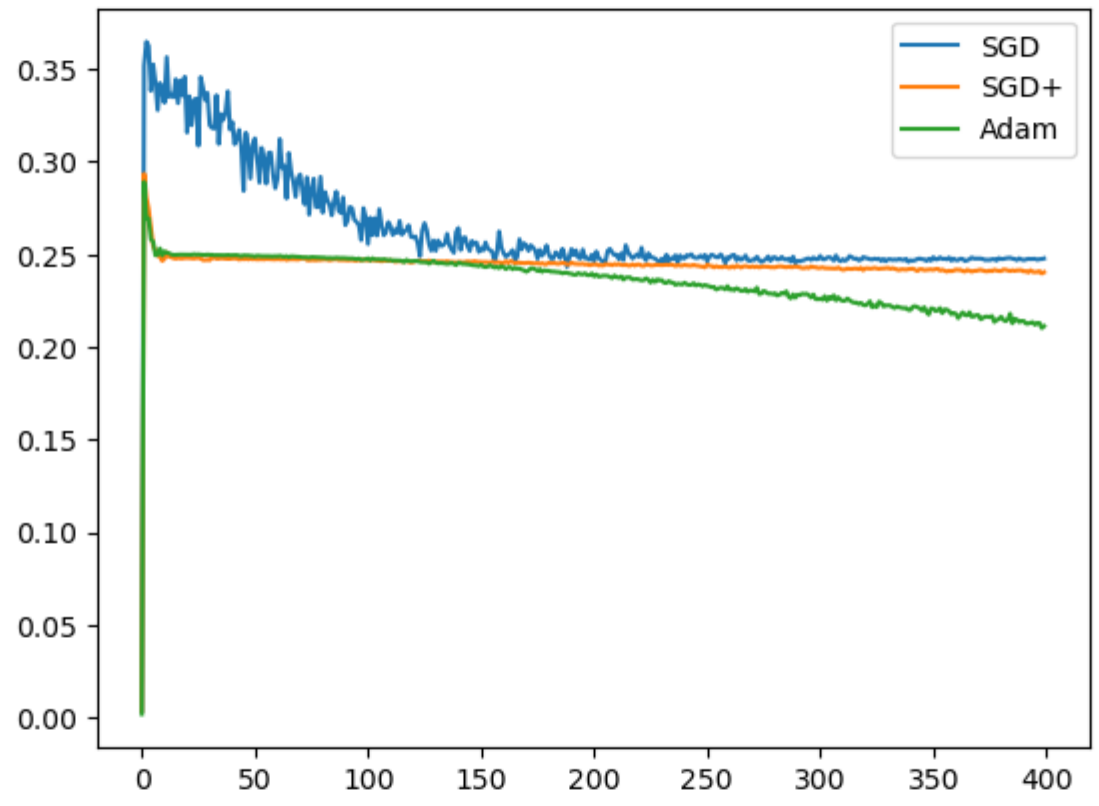
Multi-neuron model

For this model I use

```
mp1 = ModifiedPrimer(
        num_layers = 3,
        layers_config = [4,2,1],                    # num of nodes in each layer
        expressions = ['xw=ap*xp+aq*xq+ar*xr+as*xs',
                       'xz=bp*xp+bq*xq+br*xr+bs*xs',
                       'xo=cp*xw+cq*xz'],
        output_vars = ['xo'],
        dataset_size = 5000,
        learning_rate = 1e-3,
        rate_mu = 0.95,
        rate_beta1 = 0.9,
        rate_beta2 = 0.99,
        training_iterations = 40000,
        batch_size = 8,
        display_loss_how_often = 100,
        debug = True,
)
```

The result with $\eta = 1 \times 10^{-3}$ is listed below.

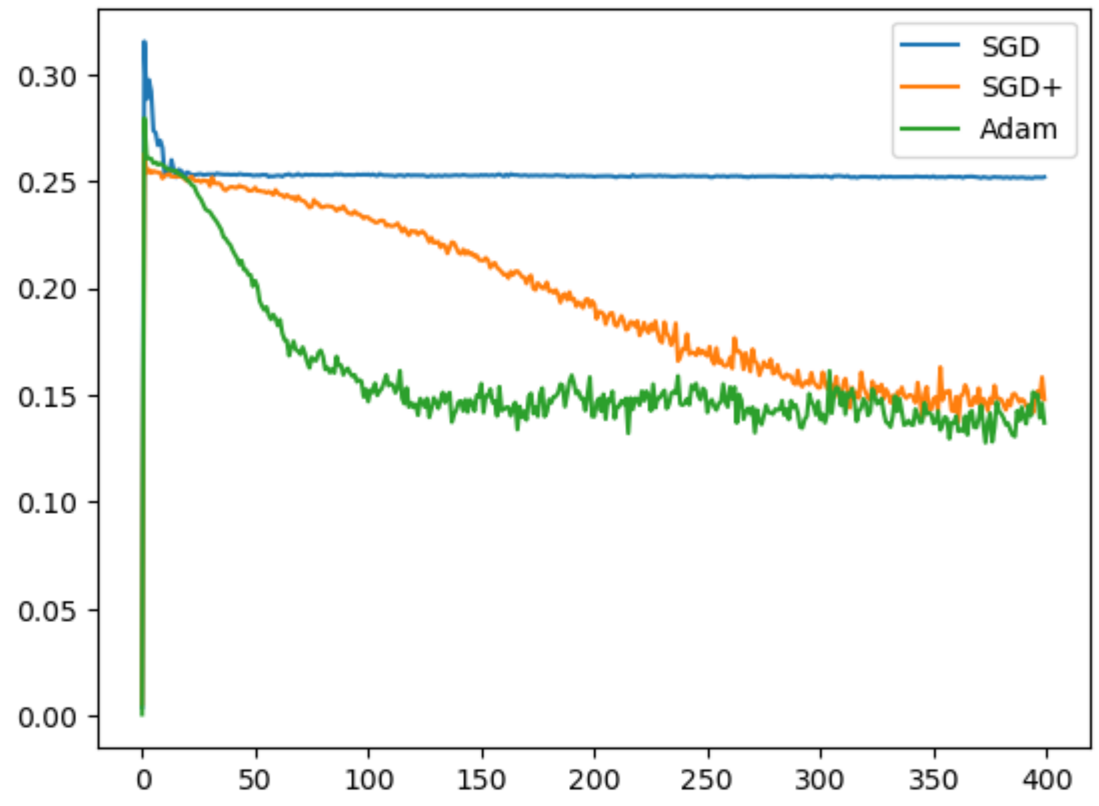
Figure 4:



I found that the Adam algorithm is still speeding up. The step size may be too small.

If I change the learning rate to $\eta = 1 \times 10^{-2}$, I get

Figure 5:



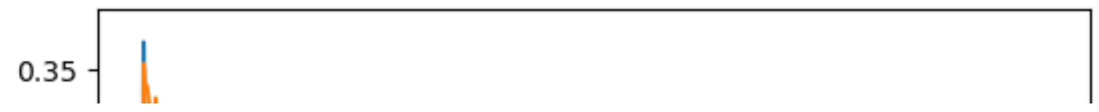
Discussion

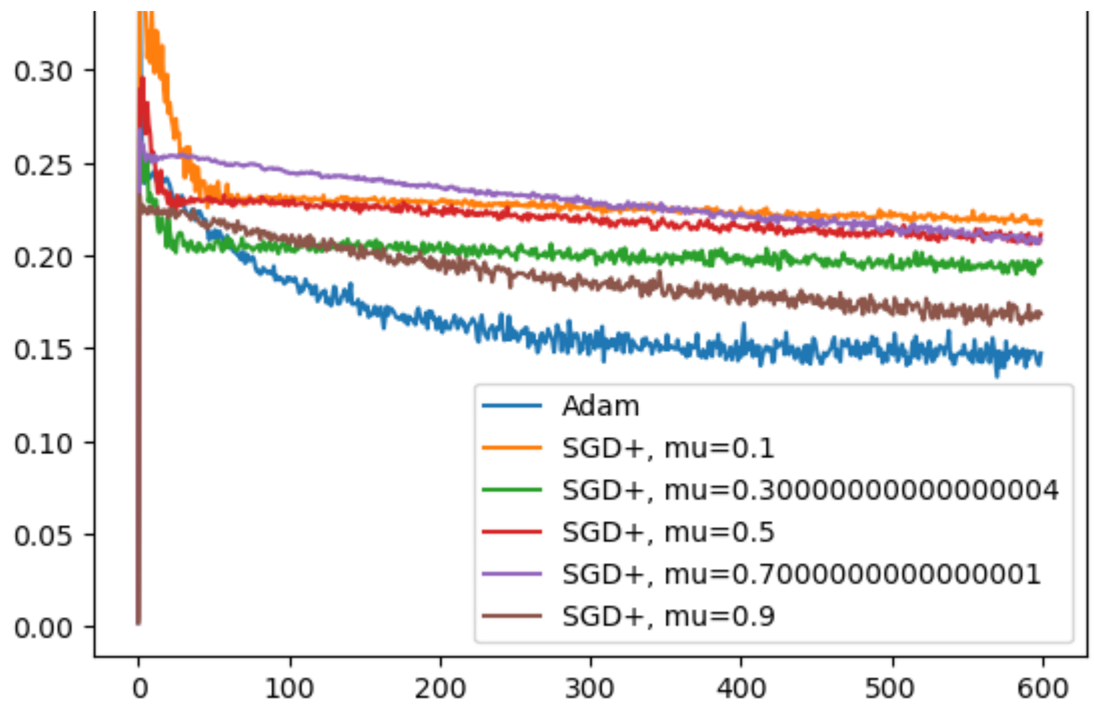
I found that the Adam algorithm outperforms the other two in most cases (with same step size).

Given enough steps Adam and SGD+ may reach the same loss around $L = 0.15$, but SGD ususlly stops above 0.15. I guess that algorithms using the momentum term $m_t = \mu m_{t-1} + (1 - \mu)\eta g_t$ makes them converge faster: Consider we are around $L = L_{\min}$ in parameter space, the total gradient $g \to 0$, but the stochastic gradient are always fluctuating. m_t is kind of mean of multiple g_t, so it should be closer to 0.

There are still some parameters to change: μ and $\beta 1, \beta 2$. By changing μ we may find its best value for SGD+, like this:

Figure 6:




0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
100
200
300
400
500
600
Adam
SGD+, mu=0.1
SGD+, mu=0.30000000000000004
SGD+, mu=0.5
SGD+, mu=0.7000000000000001
SGD+, mu=0.9