III. Lab Three - 31298516

A. Rastrigin

The Rastrigin function is a very typical function which is often used to test the performance of optimisation algorithms. And I implement it by the following code:

```
pi = torch.acos(torch.zeros(1)).item() * 2
coe = 1.0
class Rastrigin(nn.Module):
    def __init__(self):
        super().__init__()
        self.X = nn.Parameter(torch.tensor
            ↪ ([5.]))
        self.Y = nn.Parameter(torch.tensor
            ↪ ([5.]))
    def forward(self):
        return (self.X**2 - coe * torch.cos
            ↪ (2 * pi * self.X)) + (self.Y
            ↪ **2 - coe * torch.cos(2 * pi *
            ↪  self.Y)) + 2*coe
```

Spatially, it is a concave plane like a piece of cloth, and you hold four corners not too hard at the same time. And the higher the 'coe' is, the more perturbing it will be. The 'coe' value of the one we use is 1.0. The smaller value of Z it achieves, the better performance it has. In the following work, we will check the performances of the four different optimisers by searching from point (5,5) in limited 100 iterations, and then make a comparison between their values of Z. The numerical results are shown below; also, the visualisation of the paths and losses of them are namely shown in 'Figure 5' and 'Figure 6'.

1) SGD (lr=0.01):
   Z=17.053 at [2.822, 2.822, 17.053]
2) SGD&Momentum (lr=0.01, momentum=0.9):
   Z=1.904 at [-0.947, -0.947, 1.905]
3) Adagrad (lr=0.01):
   Z=47.806 at [4.842, 4.842, 47.806]
4) Adam (lr=0.01):
   Z=31.297 at [3.939, 3.939, 31.297]

As can be seen, SGD+Momentum performs the best which have already been hovering at the bottom for a long time while others are still struggling, and Adam performs the worst — it is almost vanished in this graph(look at the grey arrow). And the loss function of Adam optimiser is almost a straight line.

B. Iris SVM

Subsequently, we implemented a soft-margin SVM for solving a two-class classification problem using the iris dataset and train it with two different optimisers; namely the SGD as well as Adam with 0.0001 weight decay, 0.01 learning rate, 25 batch size and 100 epochs. The

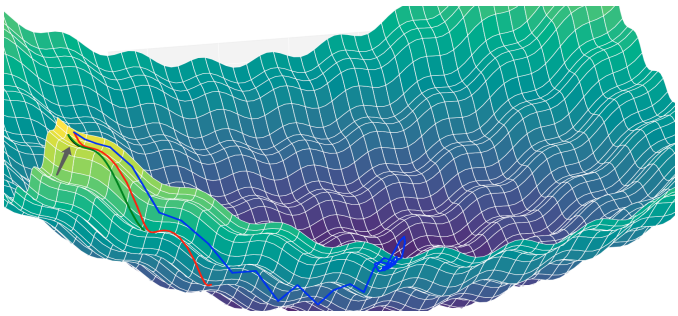

Figure 5: 3-D plot of the paths of four different optimisers, red - SGD; blue - SGD&Momentum; black - Adagrad; green - Adam.

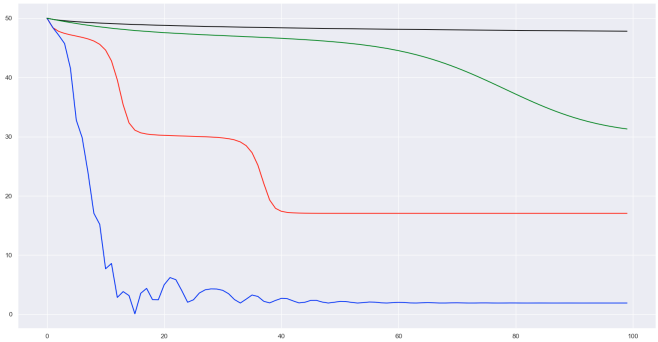

Figure 6: Losses curves of four different optimisers, red - SGD; blue - SGD&Momentum; black - Adagrad; green - Adam.

distribution diagrams of their validation accuracy which contain 1024 dependent runs in each one are shown below.

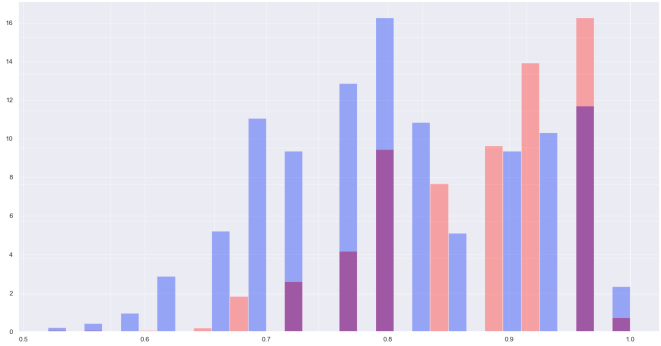

Figure 7: Distribution diagram of the validation accuracy of SGD optimiser(blue), and of Adam optimiser(red).

It can be seen that the average performance of Adam optimiser is much better than SGD in this work, and it denotes that the performance of SGD optimiser is unstable and relies more on the initialisations. But to be honest I didn't notice anything that is counter-intuitive about random initialisations.