114-1 Machine Learning
Week 2 Assignment

Ming Hsun Wu

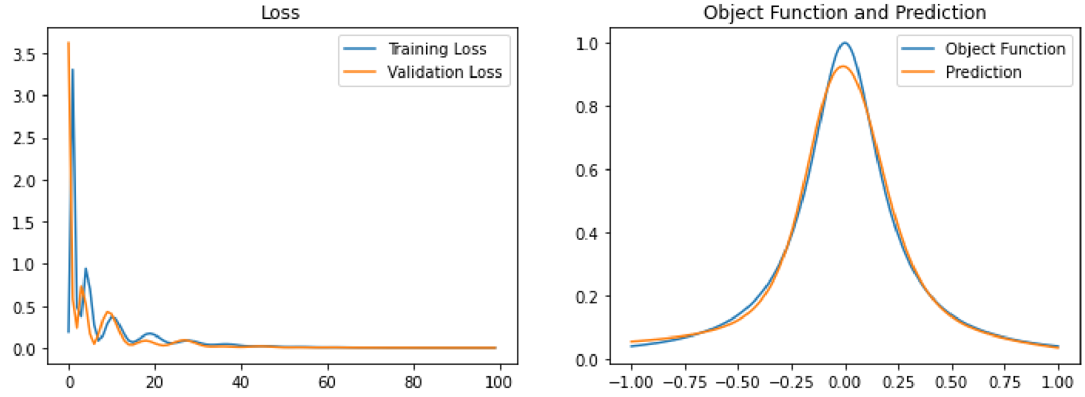
September 15, 2025

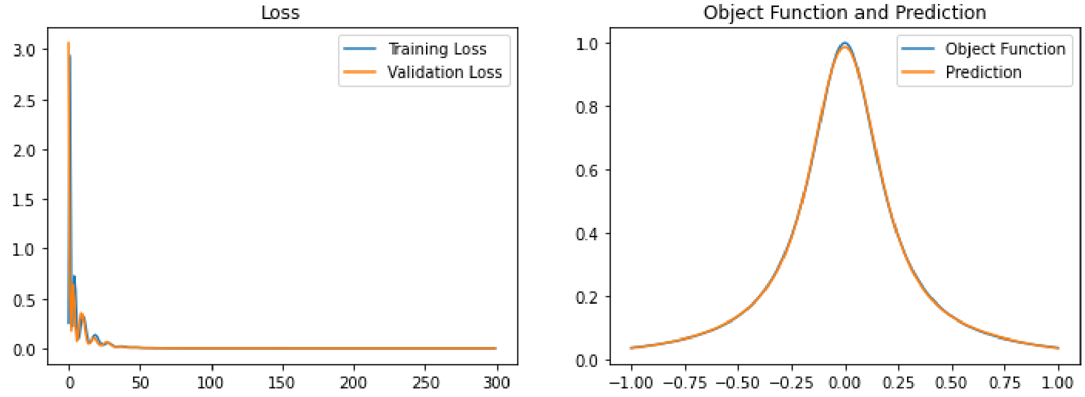
Problem 1.

Use a neural network to approximate the Runge function

$$f(x) = \frac{1}{1+25x^2}, x \in [-1, 1].$$

Solution.





Note of Problem 1.

The programming language I used is Python, and I mainly relied on the PyTorch library to carry out the simulations. Based on the trained model, the mean squared error (MSE) of the prediction results, evaluated on 500 points within the interval$[1, 1]$, is 0.000520.

Figure1 shows the results of the preliminary test. After tuning more parameters (such as the number of epochs), better results can be obtained, as shown in Figure2, with the MSE reduced to 0.000014.

```
# Algorithm Code
# (Define object function f and some parameters)
'create network'
class Curve_Fitting(torch.nn.Module):
    def __init__(self):
        super(Curve_Fitting, self).__init__()
        self.Linear_1 = torch.nn.Linear(1, 128)
        self.Linear_2 = torch.nn.Linear(128, 64)
        self.Linear_3 = torch.nn.Linear(64, 1)
        self.Tanh = torch.nn.Tanh()

    def forward(self, x):
        x = self.Linear_1(x)
        x = self.Linear_2(x)
        x = self.Tanh(x)
        x = self.Linear_3(x)

        return x

# compile model
model = Curve_Fitting()
MSE = torch.nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)
'train model and predict'
epochs = 100
train_losses = []
```

```python
val_losses = []
for epoch in range(epochs):
    # train
    model.train()
    optimizer.zero_grad()
    outputs = model(x_train)
    loss = MSE(outputs, y_train)
    loss.backward()
    optimizer.step()
    train_losses.append(loss.item())
    # val
    model.eval()
    with torch.no_grad():
        val_outputs = model(x_val)
        val_loss = MSE(val_outputs, y_val).item()
        val_losses.append(val_loss)


# estimation
x_test = torch.tensor(np.linspace(-1, 1, 500).reshape(-1,
    1).astype(np.float32))
with torch.no_grad():
    y_pred = model(x_test).numpy()

y_true = object_function(x_test.numpy())
```