```python
import numpy as np
import torch
```

Gradient Descent Practice

```python
inputs = np.array([[73, 67, 43],
                   [91, 88, 64],
                   [87, 134, 58],
                   [102, 43, 37],
                   [69, 96, 70]], dtype='float32')
```

```python
targets = np.array([[56, 70],
                    [81, 101],
                    [119, 133],
                    [22, 37],
                    [103, 119]], dtype='float32')
```

```python
inputs = torch.from_numpy(inputs)
targets = torch.from_numpy(targets)
```

```python
# Equation Looks like:
  # apple = w11*x + w12*y + w13*z + b1
  # orange = w21*x + w22*y + w23*z + b2

# Weight Initialization:
w = torch.randn(size=(2, 3), requires_grad=True) # 2x3
# Biases Initialization:
b = torch.randn(size=(2, ), requires_grad=True)

display(w)
display(b)
```

```
tensor([[-1.3902, -0.1076, -1.2027],
        [ 0.1122, -0.8120,  0.7782]], requires_grad=True)
tensor([-0.2583,  0.3550], requires_grad=True)
```

```python
# Model
def model(inputs):
  return inputs @ w.t() + b
```

```python
# Loss
def loss_(predictions, truths):
  diff = predictions - truths

  return torch.sum(diff * diff) / diff.numel()
```

```python
# Full Training
epochs = 100
learning_rate = 3e-5
```

```python
for epoch in range(epochs):
  print(f"Epoch Number: {epoch}: ")
  predictions = model(inputs)
  loss = loss_(predictions, targets)
  print(f"\tLoss: ", loss.detach().numpy())
  loss.backward()
  with torch.no_grad():
    factor = w*learning_rate
    factor_2 = b*learning_rate
    w.sub_(factor)
    b.sub_(factor_2)
```

```
Epoch Number: 0:
	Loss:  44361.797
Epoch Number: 1:
	Loss:  44360.145
Epoch Number: 2:
	Loss:  44358.496
Epoch Number: 3:
	Loss:  44356.844
Epoch Number: 4:
	Loss:  44355.19
Epoch Number: 5:
	Loss:  44353.543
Epoch Number: 6:
	Loss:  44351.883
Epoch Number: 7:
	Loss:  44350.24
Epoch Number: 8:
	Loss:  44348.586
Epoch Number: 9:
	Loss:  44346.938
Epoch Number: 10:
	Loss:  44345.29
Epoch Number: 11:
	Loss:  44343.633
Epoch Number: 12:
	Loss:  44341.98
Epoch Number: 13:
	Loss:  44340.332
Epoch Number: 14:
```

```
    Loss:  44338.68
Epoch Number: 15:
    Loss:  44337.03
Epoch Number: 16:
    Loss:  44335.38
Epoch Number: 17:
    Loss:  44333.734
Epoch Number: 18:
    Loss:  44332.08
Epoch Number: 19:
    Loss:  44330.43
Epoch Number: 20:
    Loss:  44328.773
Epoch Number: 21:
    Loss:  44327.13
Epoch Number: 22:
    Loss:  44325.48
Epoch Number: 23:
    Loss:  44323.83
Epoch Number: 24:
    Loss:  44322.18
Epoch Number: 25:
    Loss:  44320.54
Epoch Number: 26:
    Loss:  44318.883
Epoch Number: 27:
    Loss:  44317.24
Epoch Number: 28:
    Loss:  44315.586
Epoch Number: 29:
    Loss:  44313.945
Epoch Number: 30:
    Loss:  44312.29
Epoch Number: 31:
    Loss:  44310.65
Epoch Number: 32:
    Loss:  44308.992
Epoch Number: 33:
    Loss:  44307.344
Epoch Number: 34:
    Loss:  44305.695
Epoch Number: 35:
    Loss:  44304.055
```

```
Epoch Number: 36:
    Loss:  44302.406
Epoch Number: 37:
    Loss:  44300.766
Epoch Number: 38:
    Loss:  44299.113
Epoch Number: 39:
    Loss:  44297.47
Epoch Number: 40:
    Loss:  44295.824
Epoch Number: 41:
    Loss:  44294.18
Epoch Number: 42:
    Loss:  44292.53
Epoch Number: 43:
    Loss:  44290.89
Epoch Number: 44:
    Loss:  44289.242
Epoch Number: 45:
    Loss:  44287.6
Epoch Number: 46:
    Loss:  44285.95
Epoch Number: 47:
    Loss:  44284.31
Epoch Number: 48:
    Loss:  44282.664
Epoch Number: 49:
    Loss:  44281.023
Epoch Number: 50:
    Loss:  44279.375
Epoch Number: 51:
    Loss:  44277.727
Epoch Number: 52:
    Loss:  44276.08
Epoch Number: 53:
    Loss:  44274.44
Epoch Number: 54:
    Loss:  44272.793
Epoch Number: 55:
    Loss:  44271.15
Epoch Number: 56:
    Loss:  44269.508
Epoch Number: 57:
```

```
    Loss:  44267.86
Epoch Number: 58:
    Loss:  44266.22
Epoch Number: 59:
    Loss:  44264.57
Epoch Number: 60:
    Loss:  44262.926
Epoch Number: 61:
    Loss:  44261.285
Epoch Number: 62:
    Loss:  44259.633
Epoch Number: 63:
    Loss:  44257.992
Epoch Number: 64:
    Loss:  44256.35
Epoch Number: 65:
    Loss:  44254.703
Epoch Number: 66:
    Loss:  44253.055
Epoch Number: 67:
    Loss:  44251.414
Epoch Number: 68:
    Loss:  44249.77
Epoch Number: 69:
    Loss:  44248.13
Epoch Number: 70:
    Loss:  44246.484
Epoch Number: 71:
    Loss:  44244.84
Epoch Number: 72:
    Loss:  44243.195
Epoch Number: 73:
    Loss:  44241.55
Epoch Number: 74:
    Loss:  44239.902
Epoch Number: 75:
    Loss:  44238.26
Epoch Number: 76:
    Loss:  44236.617
Epoch Number: 77:
    Loss:  44234.973
Epoch Number: 78:
    Loss:  44233.33
```

```
Epoch Number: 79:
    Loss:  44231.684
Epoch Number: 80:
    Loss:  44230.04
Epoch Number: 81:
    Loss:  44228.4
Epoch Number: 82:
    Loss:  44226.758
Epoch Number: 83:
    Loss:  44225.11
Epoch Number: 84:
    Loss:  44223.473
Epoch Number: 85:
    Loss:  44221.83
Epoch Number: 86:
    Loss:  44220.18
Epoch Number: 87:
    Loss:  44218.54
Epoch Number: 88:
    Loss:  44216.895
Epoch Number: 89:
    Loss:  44215.254
Epoch Number: 90:
    Loss:  44213.605
Epoch Number: 91:
    Loss:  44211.96
Epoch Number: 92:
    Loss:  44210.32
Epoch Number: 93:
    Loss:  44208.68
Epoch Number: 94:
    Loss:  44207.035
Epoch Number: 95:
    Loss:  44205.395
Epoch Number: 96:
    Loss:  44203.742
Epoch Number: 97:
    Loss:  44202.1
Epoch Number: 98:
    Loss:  44200.46
Epoch Number: 99:
    Loss:  44198.816
```

```
final_predictions = model(inputs)
display(final_predictions)
display(targets)
```

```
tensor([[-160.1831,  -12.3593],
        [-212.5647,  -11.0532],
        [-204.7594,  -53.3977],
        [-190.6086,    5.6595],
        [-190.1254,  -15.3354]], grad_fn=<AddBackward0>)
tensor([[ 56.,  70.],
        [ 81., 101.],
        [119., 133.],
        [ 22.,  37.],
        [103., 119.]])
```

Questions for Review

Try answering the following questions to test your understanding of the topics covered in this notebook:

1. What is a linear regression model? Give an example of a problem formulated as a linear regression model.

- In linear regression, we assume a relation between inputs and the targets and then we weigh the inputs based on their importance by calculating their gradients.

2. What are input and target variables in a dataset? Give an example.

- Input: The input to the model. Can be called features too. Target: Truth values.

3. What are weights and biases in a linear regression model?

- Weights: Importance of a specific feature/input. Bias: A threshold / cutoff for the targets.

4. How do you represent tabular data using PyTorch tensors?

- torch.tensor

5. Why do we create separate matrices for inputs and targets while training a linear regression model?

- Because we have to calculate the loss.

6. How do you determine the shape of the weights matrix & bias vector given some training data?

- Weight matrix: (number of targets, number of features). Bias Vector: (number of targets).

7. How do you create randomly initialized weights & biases with a given shape?

- torch.randn(size=())

8. How is a linear regression model implemented using matrix operations? Explain with an example. Done.
9. How do you generate predictions using a linear regression model?

- model(inputs)

10. Why are the predictions of a randomly initialized model different from the actual targets?

- Because the weights aren't right.

11. What is a loss function? What does the term “loss” signify?

- The error in predictions of the model.

12. What is mean squared error?

- error^2 / num_of_inputs

13. Write a function to calculate mean squared using model predictions and actual targets.

- Done

14. What happens when you invoke the `.backward` function on the result of the mean squared error loss function?

- Calculates gradient

15. Why is the derivative of the loss w.r.t. the weights matrix itself a matrix? What do its elements represent?

- How much do we have to penalize the weights.

16. How is the derivate of the loss w.r.t. a weight element useful for reducing the loss? Explain with an example.

- We have to lower or increase the weight so that it can approach the minimum.

17. Suppose the derivative of the loss w.r.t. a weight element is positive. Should you increase or decrease the element’s value slightly to get a lower loss?

- Decrease

18. Suppose the derivative of the loss w.r.t. a weight element is negative. Should you increase or decrease the element’s value slightly to get a lower loss?

- Increase

19. How do you update the weights and biases of a model using their respective gradients to reduce the loss slightly?

- Multiply the weight derivative by learning rate.

20. What is the gradient descent optimization algorithm? Why is it called “gradient descent”?

- We are descending towards local minimum.

21. Why do you subtract a “small quantity” proportional to the gradient from the weights & biases, not the actual gradient itself?

- Learning rate.

22. What is learning rate? Why is it important?

- A factor with which the weights are to be managed. Important to reach minima.

23. What is `torch.no_grad`?

- Explicitly tell pytorch to not calculate gradients because they are being done in the context explicitly.

24. Why do you reset gradients to zero after updating weights and biases?

- Don't add up.

25. What are the steps involved in training a linear regression model using gradient descent?

- 1. Predictions, 2. Loss, 3. Gradients, 4. Update weights.

26. What is an epoch?

- Cycle.

27. What is the benefit of training a model for multiple epochs?

- Slowly slowly descending towards local minimum.

28. How do you make predictions using a trained model?

- model(inputs)

29. What should you do if your model's loss doesn't decrease while training? Hint: learning rate.

- Lower learning rate.

30. What is `torch.nn`?

- neural network module of pytorch

31. What is the purpose of the `TensorDataset` class in PyTorch? Give an example.

- Convert to train test batch.

32. What is a data loader in PyTorch? Give an example.

- For batching of data.

33. How do you use a data loader to retrieve batches of data?

- Updating of model with batches.

34. What are the benefits of shuffling the training data before creating batches?

- Randomize the targets variable.

35. What is the benefit of training in small batches instead of training with the entire dataset?

- To learn the weights sequentially.

36. What is the purpose of the `nn.Linear` class in PyTorch? Give an example.

- Linear module.

37. How do you see the weights and biases of a `nn.Linear` model?

- model.weight, model.bias

38. What is the purpose of the `torch.nn.functional` module?

- For loss functions

39. How do you compute mean squared error loss using a PyTorch built-in function?

- F.mse_loss

40. What is an optimizer in PyTorch?

- helps achieve gradient descent quicker.

41. What is `torch.optim.SGD`? What does SGD stand for?

- stochastic gradient descent.

42. What are the inputs to a PyTorch optimizer?

- model's parameters and learning rate.

43. Give an example of creating an optimizer for training a linear regression model.

- torch.optim.SGD(model.parameters(), lr=1e-5)

44. Write a function to train a `nn.Linear` model in batches using gradient descent.

- Done

45. How do you use a linear regression model to make predictions on previously unseen data?

- model(unseen_input)