PyTorch Tutorial
TF4063

Fadjar Fathurrahman

1 Linear regression

```python
import torch
import torch.nn as nn
import numpy as np
import matplotlib.pyplot as plt

RANDOM_SEED = 1234
np.random.seed(RANDOM_SEED)
torch.manual_seed(RANDOM_SEED)

def create_dataset(Ndata):
    w_true = [0.25, 1.1]
    noise_var = 0.01
    x_train = -5.0 + 10*np.random.rand(Ndata)
    y_noise = np.sqrt(noise_var)*np.random.randn(Ndata)
    y_train = w_true[1]*x_train + w_true[0] + y_noise
    return x_train, y_train

Ndata = 20
x_train, y_train = create_dataset(Ndata)
plt.clf()
plt.plot(x_train, y_train, linewidth=0, marker="o")
plt.savefig("IMG_data.pdf")

# Convert data to tensor, we need to reshape it first
x_train_ = np.reshape(x_train, (Ndata,1))
y_train_ = np.reshape(y_train, (Ndata,1))
# Default floating point number in PyTorch is float32
inputs = torch.tensor(x_train_, dtype=torch.float32)
targets = torch.tensor(y_train_, dtype=torch.float32)

# Hyper-parameters
input_size = 1 # only one feature
output_size = 1 #
num_epochs = 200
learning_rate = 0.1

# Linear regression model
model = nn.Linear(input_size, output_size)

# Loss and optimizer
criterion = nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)
```

```python
# Train the model
for epoch in range(num_epochs):
    # Forward pass
    outputs = model(inputs)
    loss = criterion(outputs, targets)
    # Backward and optimize
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if (epoch+1) % 5 == 0:
        print ('Epoch [{}/{}], Loss: {:.4f}'.format(epoch+1, num_epochs, loss.item()))

print("Parameters")
for name, param in model.named_parameters():
    print(name, " ", param.data)

# Plot the graph
predicted = model(inputs).detach().numpy()
plt.plot(x_train, y_train, 'ro', label='Original data')
plt.plot(x_train, predicted, label='Fitted line')
plt.legend()
plt.show()

# Save the model checkpoint
torch.save(model.state_dict(), 'model.ckpt')
```