model

September 11, 2024

[ ]:

```
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F

from torch.utils.data import DataLoader, TensorDataset
```

[ ]:

```
class SimpleNN(nn.Module):
    def __init__(self, input_size, hidden_size1, hidden_size2, output_size):
        super(SimpleNN, self).__init__()

        self.fc1 = nn.Linear(input_size, hidden_size1)
        self.fc2 = nn.Linear(hidden_size1, hidden_size2)
        self.fc3 = nn.Linear(hidden_size2, output_size)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x
```

[ ]:

```
X_train = torch.randn(10, 3)  # 10 samples, 10 features each
y_train = torch.randint(0, 1, (10,))  # 10 samples, 1 output

train_dataset = TensorDataset(X_train, y_train)
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
```

[ ]:

```
input_size = 3  # Number of input features
hidden_size1 = 3  # Size of the first hidden layer
hidden_size2 = 4  # Size of the second hidden layer
output_size = 1  # Number of classes

# Initialize the model, loss function, and optimizer
model = SimpleNN(input_size, hidden_size1, hidden_size2, output_size)
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

[ ]:

```
num_epochs = 10

for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0

    for inputs, labels in train_loader:
        inputs = inputs.float()
        labels = labels.float()

        # Zero the gradients
        optimizer.zero_grad()

        outputs = model(inputs)

        loss = 1/2 * criterion(outputs, labels)

        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {running_loss/
 ↪len(train_loader):.4f}")
```

```
Epoch [1/10], Loss: 0.0130
Epoch [2/10], Loss: 0.0127
Epoch [3/10], Loss: 0.0123
Epoch [4/10], Loss: 0.0119
Epoch [5/10], Loss: 0.0115
Epoch [6/10], Loss: 0.0112
Epoch [7/10], Loss: 0.0109
Epoch [8/10], Loss: 0.0106
Epoch [9/10], Loss: 0.0103
Epoch [10/10], Loss: 0.0100
```

```
d:\DataScience\Anaconda3\envs\dl4cv\lib\site-
packages\torch\nn\modules\loss.py:538: UserWarning: Using a target size
(torch.Size([10])) that is different to the input size (torch.Size([10, 1])).
This will likely lead to incorrect results due to broadcasting. Please ensure
they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
```

[ ]:

```
X_test = torch.randn(20, 3)
y_test = torch.randint(0, 3, (20,))

model.eval()
```

```python
with torch.no_grad():
    test_outputs = model(X_test)
    _, predicted = torch.max(test_outputs, 1)
    accuracy = (predicted == y_test).float().mean()
    print(f"Test Accuracy: {accuracy:.2f}")
```

```
Test Accuracy: 0.30
```