## Starter Model

```
<font size="3">
import torch.nn as nn
class Classifier(nn.Module):
    def __init__(self, num_features, num_classes):
        super(Classifier, self).__init__()

        self.fully_connected_1 =
nn.Linear(num_features, 4)
        self.fully_connected_2 = nn.Linear(4,
num_classes)

    def forward(self, x):
        x = F.relu(self.fully_connected_1(x))
        x = self.fully_connected_2(x)
        return x
# Instantiate the model
classifier = Classifier(4, 2)
print(f"classifer.train: {classifier.training}") #
Should be true until test time.
print(classifier) # print the layers of the model
</font>
```

## Instantiate Optimizer

```
import torch.optim as optim
LEARNING_RATE = 0.01
minimizer = optim.SGD(classifier, lr=LEARNING_RATE)
```

## Starter Training Loop

```
# Each epoch is a full run through all examples.
for epoch in range(NUM_EPOCHS):
    # Iterate through the data, one batch at a time.
    for i, (X_data, y_labels) in
enumerate(train_loader):
        # Zero out the gradients from previous loop.
        minimizer.zero_grad()

        # Apply the classifier(model) to a batch of
examples,
        # get scores.
        y_scores = classifier(X_data) # Predict.
        # Calculate the classifier's error from labels
        loss = cross_entropy_loss(y_scores, y_labels)
        # Calculate the gradients with respect to the
        # model's weights.
        loss.backward()
        # Update the model's weights.
        minimizer.step()
```