Starter Model

```
<font size="3 " >
import torch.nn as nn
class Classi fie r(n n.M odule):
       def __init __( self, num_fe atures,
num_cl asses):
                sup er( Cla ssi fier, self)._ _i -
nit__()

                sel f.f ull y_c onn ected_1 =
nn.Lin ear (nu m_f eat ures, 4)
                sel f.f ull y_c onn ected_2 =
nn.Lin ear(4, num_cl asses)

       def forwar d(self, x):
               x = F.relu (se lf.f ul ly_ con nec -
ted _1(x))
               x = self.f ull y_c onn ect ed_2(x)
                return x
# Instan tiate the model
classifier = Classi fier(4, 2)
print( f"cl ass ife r.t rain: {class ifi er.t ra -
ini ng} ") # Should be true until test time.
print( cla ssi fier) # print the layers of the
model
</  fon  t>
```

Instantiate Optimizer

```
import torch.optim as optim
LEARNI NG_RATE = 0.01
minimizer = optim.S GD (cl ass ifier, lr=LEA RNI -
NG_ RATE)
```

Starter Training Loop

```
# Each epoch is a full run through all examples.
for epoch in range( NUM _EP OCHS):
       # Iterate through the data, one batch at a
time.
       for i, (X_data, y_labels) in enumer ate -
(tr ain _lo ader):
               # Zero out the gradients from
previous loop.
                min imi zer.ze ro_ grad()

               # Apply the classi fie r(m odel) to
a batch of examples,
               # get scores.
                y_s cores = classi fie r(X _data)
# Predict.
               # Calculate the classi fier's error
from labels
               loss = cross_ ent rop y_l oss (y_ -
scores, y_labels)
               # Calculate the gradients with
respect to the
               # model's weights.
                los s.b ack ward()
               # Update the model's weights.
                min imi zer.step()
```


Sponsored by **Readable.com**
Measure your website readability!
https://readable.com