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

```
# Create a Model Class that inherits nn.Module
class Model(nn.Module):
  # Input layer (4 features of the flower)
  #   --> Hidden Layer1 H1 (number of neurone)
  #   --> Hidden Layer H2 (n)
  #   --> output (which 3 classes of iris flower)

  def __init__(self, in_features=4, h1=8, h2=9, out_featu

    super().__init__() # Instantiate nn.Module (parent cl

    self.fc1 = nn.Linear(in_features, h1)
    self.fc2 = nn.Linear(h1, h2)
    self.out = nn.Linear(h2, out_features)


  def forward(self, x):
    x = F.relu(self.fc1(x))
    x = F.relu(self.fc2(x))
    x = self.out(x)

    return x


```

```
# Pick a manual seed for randomization
torch.manual_seed(41)
# Create an instance of the model Model
model = Model()
```

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

```

```
url = "https://gist.githubusercontent.com/netj/8836201/ra
my_df = pd.read_csv(url)

```

```
 my_df

```

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Virginica
146	6.3	2.5	5.0	1.9	Virginica
147	6.5	3.0	5.2	2.0	Virginica
148	6.2	3.4	5.4	2.3	Virginica
149	5.9	3.0	5.1	1.8	Virginica

150 rows × 5 columns

```
 my_df.head()
```

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

```
my_df.tail()
```

	sepal.length	sepal.width	petal.length	petal.width	variety
145	6.7	3.0	5.2	2.3	Virginica
146	6.3	2.5	5.0	1.9	Virginica

147	6.5	3.0	5.2	2.0	Virginica
148	6.2	3.4	5.4	2.3	Virginica
149	5.9	3.0	5.1	1.8	Virginica

```
# Change last column from string to numbers (use as integ
my_df["variety"] = my_df.variety.replace('Setosa', 0.0)
my_df["variety"] = my_df.variety.replace('Versicolor', 1.
my_df["variety"] = my_df.variety.replace('Virginica', 2.0
my_df
# my_df['variety'] = my_df['variety'].replace('Setosa', 0
# my_df['variety'] = my_df['variety'].replace('Versicolor
# my_df['variety'] = my_df['variety'].replace('Virginica'
# my_df

```

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2.0
146	6.3	2.5	5.0	1.9	2.0
147	6.5	3.0	5.2	2.0	2.0
148	6.2	3.4	5.4	2.3	2.0
149	5.9	3.0	5.1	1.8	2.0

150 rows × 5 columns

```
# Train Test Split: Set X, y
X = my_df.drop('variety', axis=1)
y = my_df['variety']

```

```
# Convert to numpy arrays
X = X.values
y = y.values
```

```
y = y.values
```

```
X
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
       [5.1, 3.8, 1.6, 0.2],
       [4.6, 3.2, 1.4, 0.2],
       [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2],
       [7. , 3.2, 4.7, 1.4],
       [6.4, 3.2, 4.5, 1.5],
       [6.9, 3.1, 4.9, 1.5],
       [5.5, 2.3, 4. , 1.3],
       [6.5, 2.8, 4.6, 1.5],
       [6.5, 2.8, 4.6, 1.5],
       [5.7, 2.8, 4.5, 1.3],
       [6.3, 3.3, 4.7, 1.6],
       [4.9, 2.4, 3.3, 1. ],
```

```
from sklearn.model_selection import train_test_split
```

```
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y,
```

```
# Convert X features to float tensors
X_train = torch.FloatTensor(X_train)
X_test = torch.FloatTensor(X_test)
```

```
# Convert y labels to tensors long
y_train = torch.LongTensor(y_train)
y_test = torch.LongTensor(y_test)
```

```
# Set the criterion of model to mesure the error, how far
criterion = nn.CrossEntropyLoss()
# Choose Adam Optimizer, lr = learning rate (if error doe
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
```

```
model.parameters
```

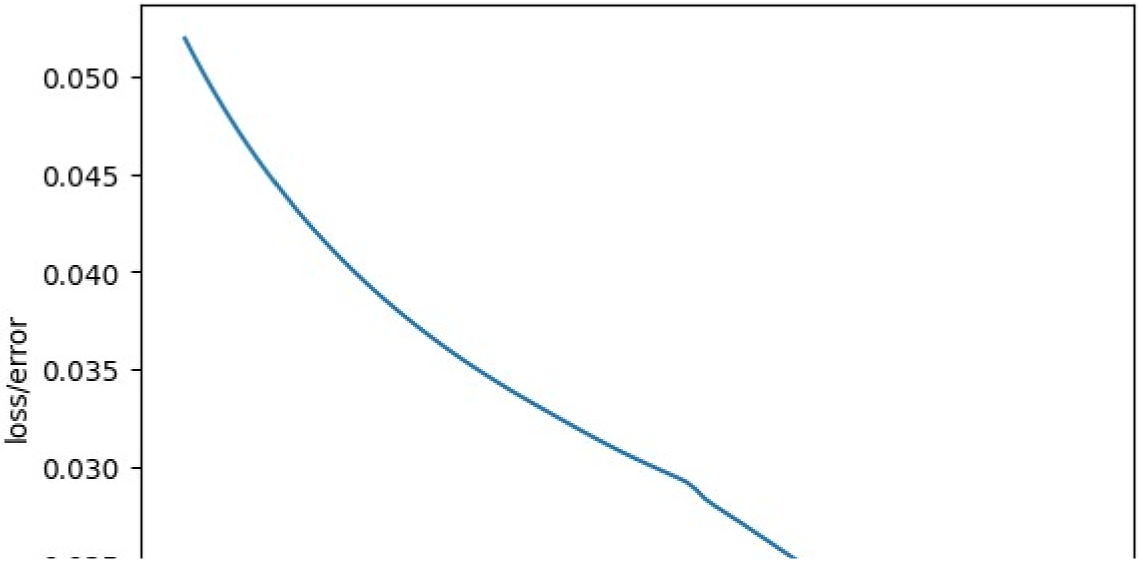
```
<bound method Module.parameters of Model(
  (fc1): Linear(in_features=4, out_features=8, bias=True)
  (fc2): Linear(in_features=8, out_features=9, bias=True)
  (out): Linear(in_features=9, out_features=3, bias=True)
)>
```

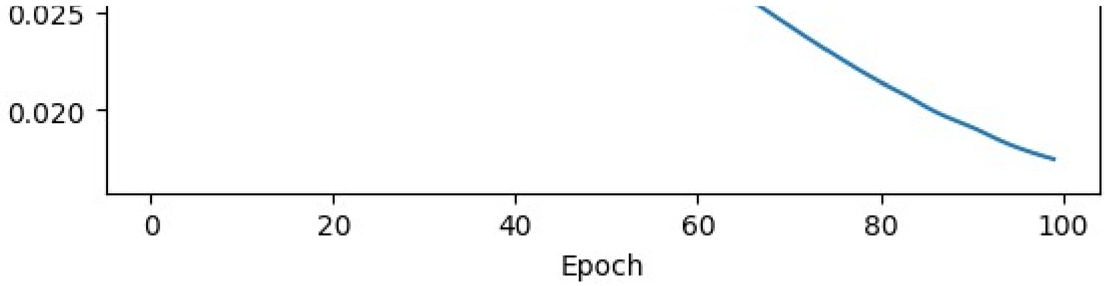
```
# Train our model!
# Epochs? (one run all the training data in our network)
epochs = 100 # How many times
losses = []
for i in range(epochs):
  # Go forward and get a prediction
  y_pred = model.forward(X_train) # Get predicted results

  # Mesure the loss/error, will be high at first
  loss = criterion(y_pred, y_train) # Predicted values vs

  # Keep track of the losses
  # Keep track of the losses
  losses.append(loss.detach().numpy())

  # Print every 10 epochs
  if i % 10 == 0:
    print(f'Epoch: {i} and loss: {loss}')

  # Do some back propagation: take the error rate of forv
  #   and feed it back through the netword to fine tune tl
  optimizer.zero_grad()
  loss.backward()
  optimizer.step()




```

```
Epoch: 0 and loss: 0.05193043872714043
Epoch: 10 and loss: 0.04446204751729965
Epoch: 20 and loss: 0.03935651853680611
Epoch: 30 and loss: 0.03563718870282173
Epoch: 40 and loss: 0.032763414084911346
Epoch: 50 and loss: 0.030291257426142693
Epoch: 60 and loss: 0.02742672711610794
Epoch: 70 and loss: 0.02430540882050991
Epoch: 80 and loss: 0.021412163972854614
Epoch: 90 and loss: 0.019128229469060898
```

```
# Graph it out!
plt.plot(range(epochs), losses)
plt.ylabel('loss/error')
plt.xlabel('Epoch')
```

```
Text(0.5, 0, 'Epoch')
```





```
1
```