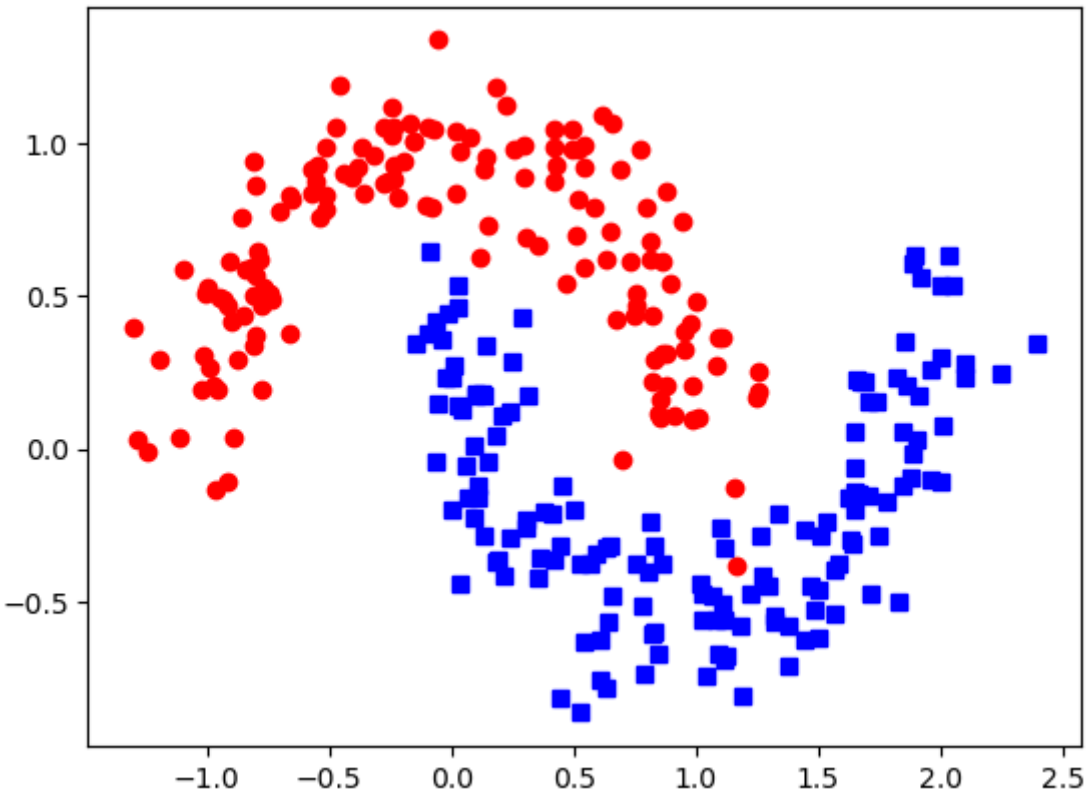
Half moon distribution - 2D space with two variables

If we have a preferential distribution it is somehow more difficult to predict the classifications using conventional statistical approach.

Here is the plot with two input values and two preferential distributed zones :



We import the python modules:

```
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
import numpy as np
import torch as tch
import torch.nn as nn
X, y = make_moons(n_samples=300, noise=0.15, random_state=0)
data = tch.tensor(X).float()
labels = tch.tensor(y).long()
plt.plot(data[labels==0,0], data[labels==0,1],'ro')
```

```
plt.plot(data[labels==1,0], data[labels==1,1], 'bs')
plt.show()
```

My approach for ANN model is to use 4 hidden layers:

```
#create the model layout

model = nn.Sequential(
    nn.Linear(2,4),   #input layer
    nn.ReLU(),        #activation layer
    nn.Linear(4,2),   #Output layer
    nn.Softmax(dim=1)
)

# loss function
lossfun = nn.CrossEntropyLoss()

#optimizer
optimizer = tch.optim.SGD(model.parameters(),lr=0.05)
```

The code to train the model going through 10000 trials:

```
#train the model
numepoch = 10000
losses = tch.zeros(numepoch)



for epochi in range(numepoch):
    #forward pass
    yHat = model(data)

    #compute losses
    loss = lossfun(yHat,labels)
    losses[epochi] = loss

    #backpropagation
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

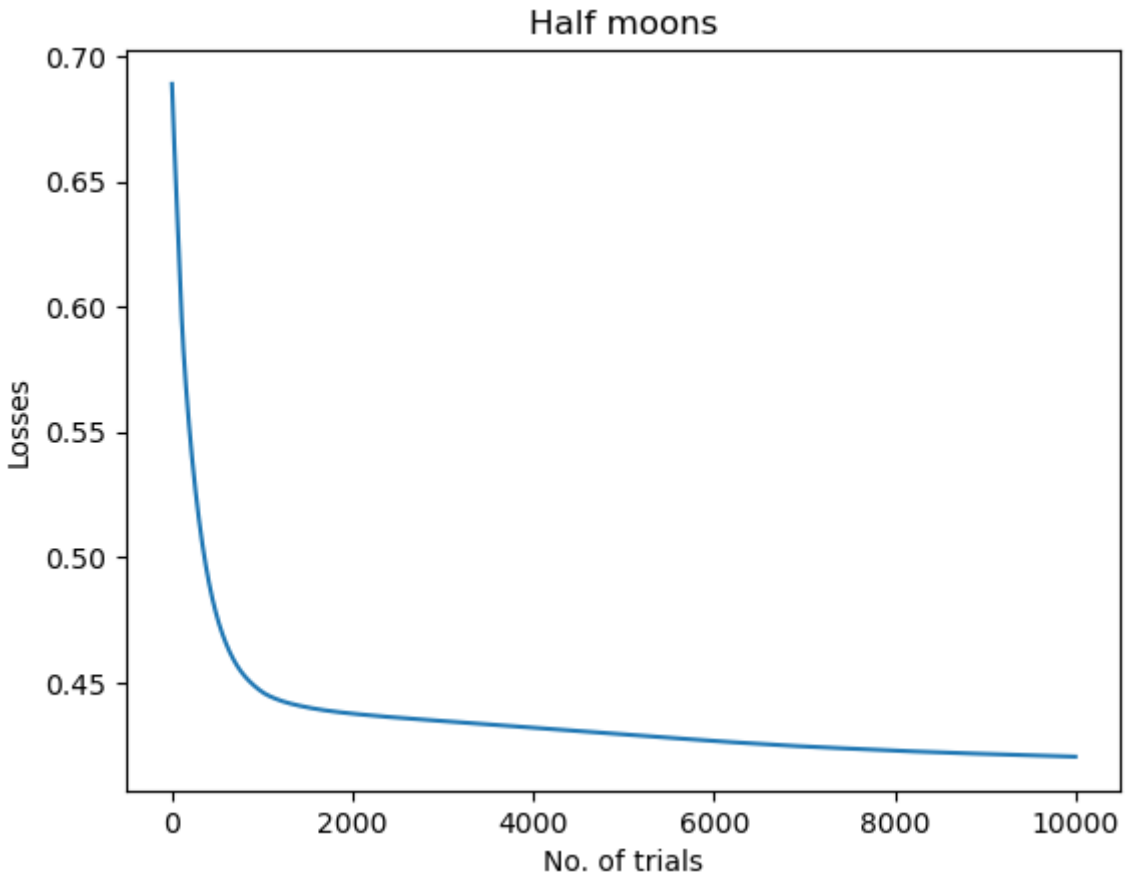
```
#final forward pass
predictions = model(data)

#compute prediction accuracy
predlabels = tch.argmax(predictions,axis=1)
totalacc = 100 * tch.mean((predlabels == labels).float())

# report accuracy
print('Final accuracy: %g%%' %totalacc)

plt.plot(losses.detach())
plt.show()
```

Final accuracy is 87.6 %. The model struggle a bit in the beginning according with losses plot.



We switch the model into evaluation mode from training mode and test with the coordinates [1,1] and hope it will predict red:

```
  model.eval()
  X_test = [[1, 1]]
test_data = tch.tensor(X_test).float()
```

```
results = model(test_data)
label_results = tch.argmax(results,axis=1)

print('test numers: ', X_test)
print(label_results)

if label_results.detach() == 0 :
    print('red')
else:
    print('blue')
```

```
test numers:  [[1, 1]]
tensor([0])
red
```

Indeed.

Revision #2
Created 20 January 2025 00:47:21 by Delta
Updated 20 January 2025 00:52:20 by Delta