yjadikz71

January 23, 2025

[ ]:

```
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import random
```

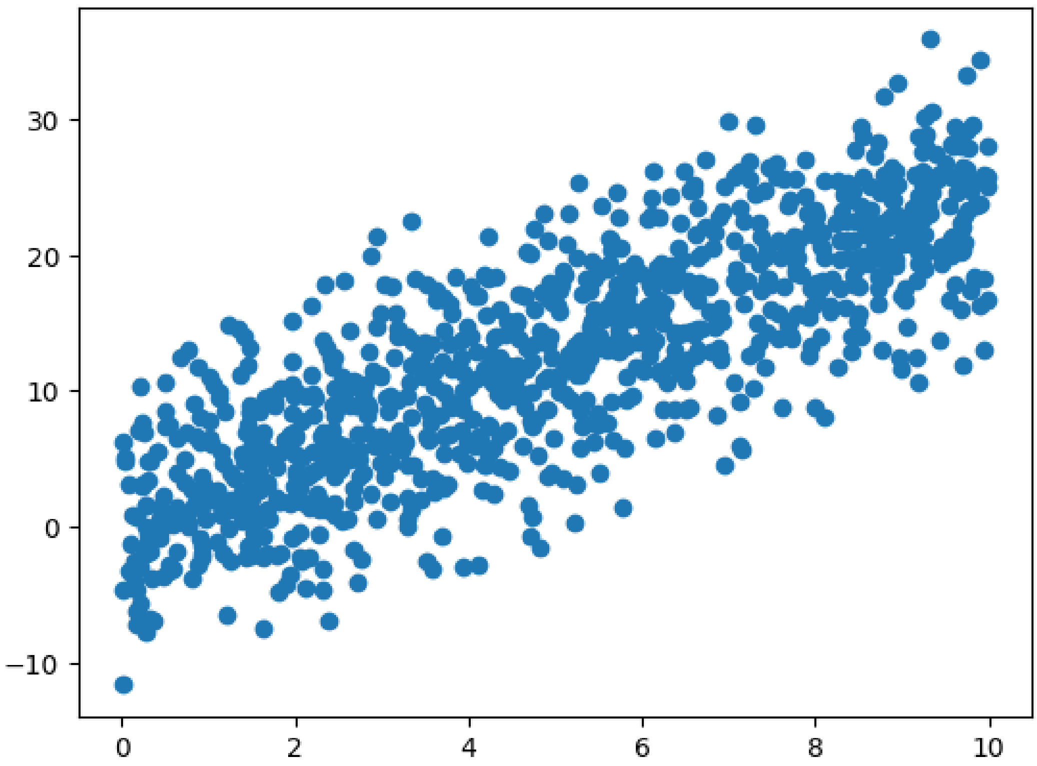
[ ]:

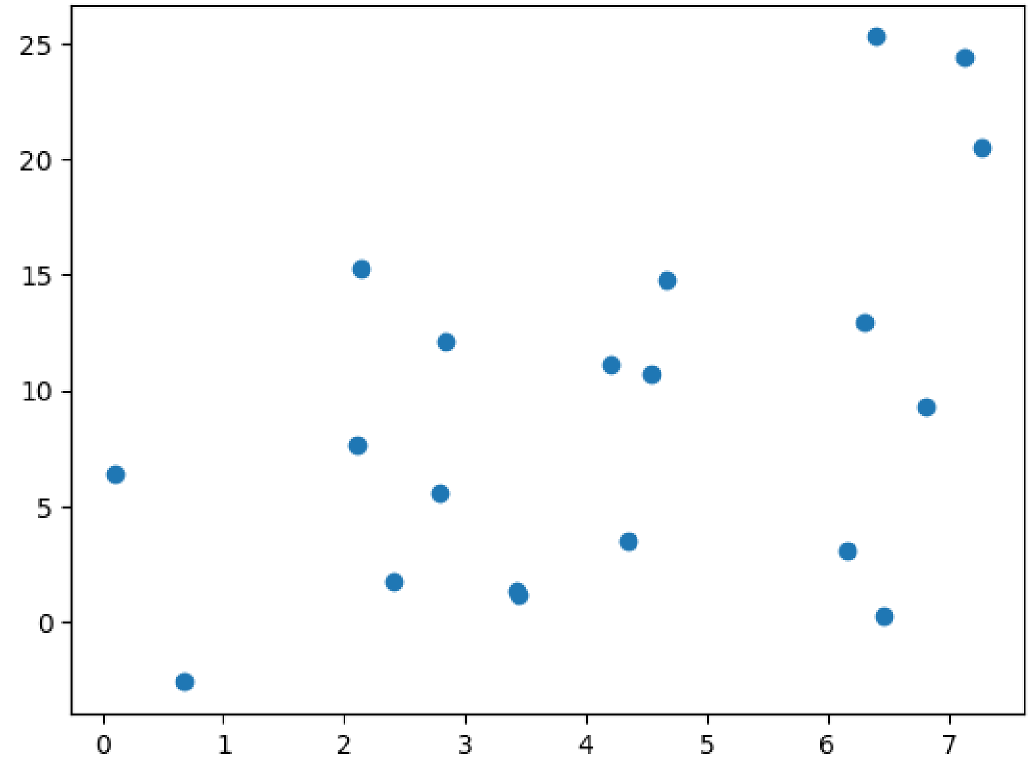
```
random.seed(10)
#Using Synthetic Dataset
X=np.random.rand(10,1)*10 #Features rand
y=2.5*X + np.random.randn(10,1)*5#randn
print(X.shape)
X_test=np.random.rand(20,1)*8
y_test=2.5*X_test + np.random.randn(20,1)*6#randn
print(y)
```

```
(10, 1)
[[23.24688435]
 [ 8.91256895]
 [18.8443837 ]
 [ 1.40498831]
 [ 7.41042097]
 [ 6.94133539]
 [15.39484356]
 [ 3.72730071]
 [10.40854618]
 [21.12852617]]
```

[ ]:

```
plt.scatter(X,y)
plt.show()

plt.scatter(X_test ,y_test)
plt.show()
```


30
20
10
0
−10
0
2
4
6
8
10


25
20
15
10
5
0
0
1
2
3
4
5
6
7

[ ]:

```
type(X)
```

[ ]:

```
numpy.ndarray
```

[ ]:

```
X_tensor=torch.tensor(X,dtype=torch.float32) #changing the datatype to make data more consistent
y_tensor=torch.tensor(y, dtype=torch.float32)

X_tests=torch.tensor(X_test,dtype=torch.float32) #changing the datatype to make data more consistent
y_tests=torch.tensor(y_test, dtype=torch.float32)
```

[ ]:

```
type(X_tensor)
```

[ ]:

```
torch.Tensor
```

[ ]:

```
class RegressionModel(nn.Module):
  def __init__(self):
    super(RegressionModel, self).__init__() #super keyword uses inheritence
    self.linear=nn.Linear(1,1) #(no. of input feature, no. of target) , nn is fully connected layer- Artificial neural network

  def forward(self,x):
    return self.linear(x)
```

[ ]:

```
model:RegressionModel = RegressionModel()
loss = nn.MSELoss() #mean squred error

criteria=torch.optim.SGD(model.parameters(), lr=0.00001) #lr=learning rate
# Gradient Descent Types -
#Batch -
#Stochastic - SGD
#Mini Batch - sampling
```

[ ]:

```
# for learning process visualization
train_losses=[]
test_losses=[]
train_accuracy=[]
train_loss=[]
```

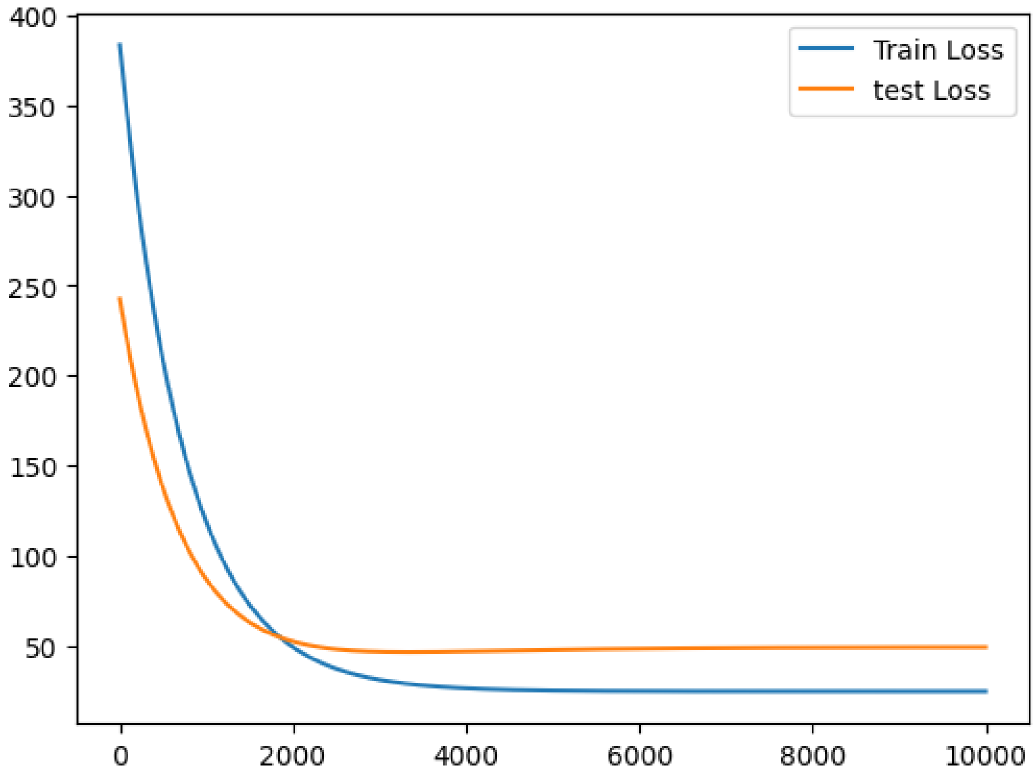
[ ]:

```
num_epochs=10000
for ep in range(num_epochs): #learning formative
  model.train()
  predicted_y = model(X_tensor)
  # actual -> y tensor
  # predicted -> predicted_y
  losses=loss(y_tensor, predicted_y)
  if(ep%100==0):
    print(losses)

  criteria.zero_grad()
  losses.backward()
  criteria.step()

  train_losses.append(losses.item()) #summative actual
  model.eval()
  with torch.no_grad():
    predictions=model(X_tests)
    test_loss=loss(y_tests, predictions)
    test_losses.append(test_loss.item())
    # 1) Model overfitting -
    # 2) Model underfitting -
```

```
tensor(383.3290, grad_fn=<MseLossBackward0>)
tensor(338.2947, grad_fn=<MseLossBackward0>)
tensor(298.9148, grad_fn=<MseLossBackward0>)
tensor(264.4791, grad_fn=<MseLossBackward0>)
tensor(234.3669, grad_fn=<MseLossBackward0>)
tensor(208.0354, grad_fn=<MseLossBackward0>)
tensor(185.0100, grad_fn=<MseLossBackward0>)
tensor(164.8754, grad_fn=<MseLossBackward0>)
tensor(147.2689, grad_fn=<MseLossBackward0>)
tensor(131.8728, grad_fn=<MseLossBackward0>)
tensor(118.4098, grad_fn=<MseLossBackward0>)
tensor(106.6370, grad_fn=<MseLossBackward0>)
tensor(96.3424, grad_fn=<MseLossBackward0>)
tensor(87.3403, grad_fn=<MseLossBackward0>)
tensor(79.4683, grad_fn=<MseLossBackward0>)
tensor(72.5847, grad_fn=<MseLossBackward0>)
tensor(66.5654, grad_fn=<MseLossBackward0>)
tensor(61.3018, grad_fn=<MseLossBackward0>)
tensor(56.6990, grad_fn=<MseLossBackward0>)
tensor(52.6741, grad_fn=<MseLossBackward0>)
tensor(49.1545, grad_fn=<MseLossBackward0>)
tensor(46.0767, grad_fn=<MseLossBackward0>)
tensor(43.3854, grad_fn=<MseLossBackward0>)
tensor(41.0320, grad_fn=<MseLossBackward0>)
tensor(38.9740, grad_fn=<MseLossBackward0>)
tensor(37.1743, grad_fn=<MseLossBackward0>)
tensor(35.6006, grad_fn=<MseLossBackward0>)
tensor(34.2245, grad_fn=<MseLossBackward0>)
tensor(33.0211, grad_fn=<MseLossBackward0>)
tensor(31.9687, grad_fn=<MseLossBackward0>)
tensor(31.0485, grad_fn=<MseLossBackward0>)
tensor(30.2437, grad_fn=<MseLossBackward0>)
tensor(29.5400, grad_fn=<MseLossBackward0>)
tensor(28.9246, grad_fn=<MseLossBackward0>)
tensor(28.3864, grad_fn=<MseLossBackward0>)
tensor(27.9158, grad_fn=<MseLossBackward0>)
tensor(27.5042, grad_fn=<MseLossBackward0>)
tensor(27.1442, grad_fn=<MseLossBackward0>)
tensor(26.8295, grad_fn=<MseLossBackward0>)
tensor(26.5542, grad_fn=<MseLossBackward0>)
tensor(26.3134, grad_fn=<MseLossBackward0>)
tensor(26.1029, grad_fn=<MseLossBackward0>)
tensor(25.9187, grad_fn=<MseLossBackward0>)
tensor(25.7576, grad_fn=<MseLossBackward0>)
tensor(25.6168, grad_fn=<MseLossBackward0>)
tensor(25.4935, grad_fn=<MseLossBackward0>)
tensor(25.3858, grad_fn=<MseLossBackward0>)
tensor(25.2915, grad_fn=<MseLossBackward0>)
tensor(25.2090, grad_fn=<MseLossBackward0>)
tensor(25.1369, grad_fn=<MseLossBackward0>)
tensor(25.0738, grad_fn=<MseLossBackward0>)
tensor(25.0185, grad_fn=<MseLossBackward0>)
tensor(24.9702, grad_fn=<MseLossBackward0>)
tensor(24.9279, grad_fn=<MseLossBackward0>)
tensor(24.8909, grad_fn=<MseLossBackward0>)
tensor(24.8585, grad_fn=<MseLossBackward0>)
tensor(24.8302, grad_fn=<MseLossBackward0>)
tensor(24.8053, grad_fn=<MseLossBackward0>)
tensor(24.7836, grad_fn=<MseLossBackward0>)
tensor(24.7645, grad_fn=<MseLossBackward0>)
tensor(24.7479, grad_fn=<MseLossBackward0>)
tensor(24.7332, grad_fn=<MseLossBackward0>)
tensor(24.7204, grad_fn=<MseLossBackward0>)
tensor(24.7092, grad_fn=<MseLossBackward0>)
tensor(24.6993, grad_fn=<MseLossBackward0>)
tensor(24.6907, grad_fn=<MseLossBackward0>)
tensor(24.6831, grad_fn=<MseLossBackward0>)
tensor(24.6764, grad_fn=<MseLossBackward0>)
tensor(24.6706, grad_fn=<MseLossBackward0>)
tensor(24.6654, grad_fn=<MseLossBackward0>)
tensor(24.6609, grad_fn=<MseLossBackward0>)
tensor(24.6569, grad_fn=<MseLossBackward0>)
tensor(24.6533, grad_fn=<MseLossBackward0>)
tensor(24.6502, grad_fn=<MseLossBackward0>)
tensor(24.6475, grad_fn=<MseLossBackward0>)
tensor(24.6450, grad_fn=<MseLossBackward0>)
tensor(24.6429, grad_fn=<MseLossBackward0>)
tensor(24.6409, grad_fn=<MseLossBackward0>)
tensor(24.6392, grad_fn=<MseLossBackward0>)
tensor(24.6377, grad_fn=<MseLossBackward0>)
tensor(24.6363, grad_fn=<MseLossBackward0>)
tensor(24.6351, grad_fn=<MseLossBackward0>)
tensor(24.6340, grad_fn=<MseLossBackward0>)
tensor(24.6330, grad_fn=<MseLossBackward0>)
tensor(24.6321, grad_fn=<MseLossBackward0>)
tensor(24.6313, grad_fn=<MseLossBackward0>)
tensor(24.6306, grad_fn=<MseLossBackward0>)
tensor(24.6299, grad_fn=<MseLossBackward0>)
tensor(24.6293, grad_fn=<MseLossBackward0>)
tensor(24.6287, grad_fn=<MseLossBackward0>)
tensor(24.6281, grad_fn=<MseLossBackward0>)
tensor(24.6277, grad_fn=<MseLossBackward0>)
tensor(24.6272, grad_fn=<MseLossBackward0>)
tensor(24.6268, grad_fn=<MseLossBackward0>)
tensor(24.6263, grad_fn=<MseLossBackward0>)
tensor(24.6260, grad_fn=<MseLossBackward0>)
tensor(24.6256, grad_fn=<MseLossBackward0>)
tensor(24.6252, grad_fn=<MseLossBackward0>)
tensor(24.6249, grad_fn=<MseLossBackward0>)
tensor(24.6246, grad_fn=<MseLossBackward0>)
```

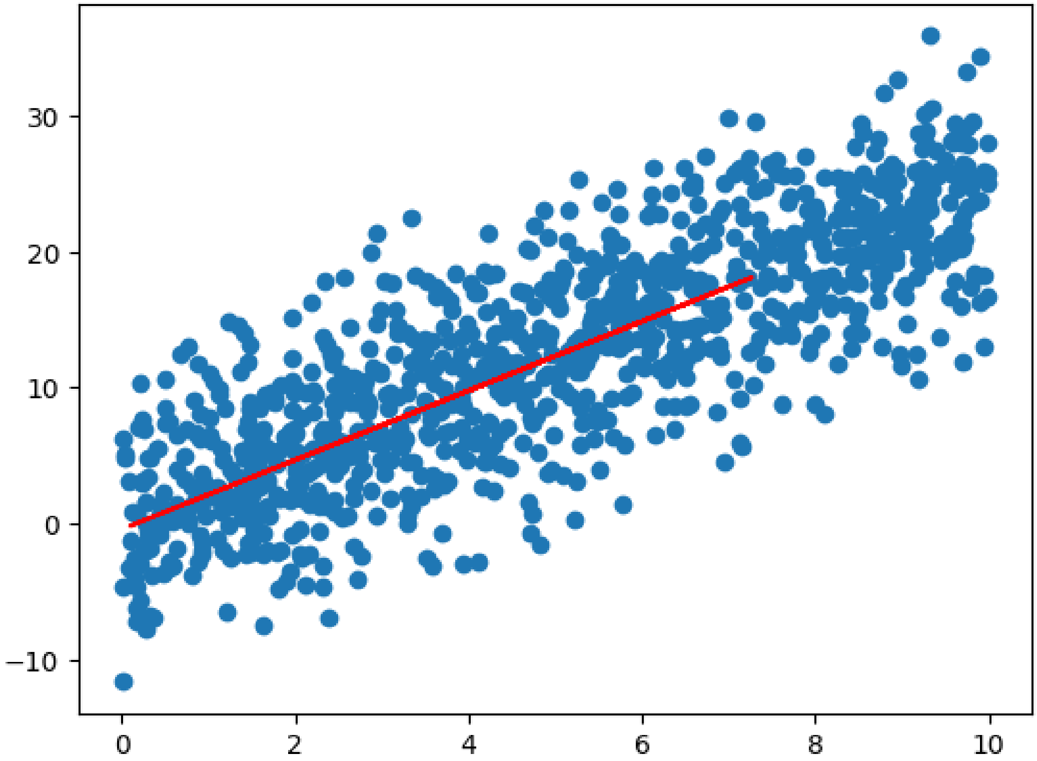
[ ]:

```
plt.plot(train_losses, label='Train Loss')
plt.plot(test_losses, label='test Loss')
plt.legend()
plt.show()
```



[ ]:

```
plt.scatter(X,y)
predictions=model(X_tests)
plt.plot(X_test,predictions.detach().numpy() ,color='red')
plt.show()
```

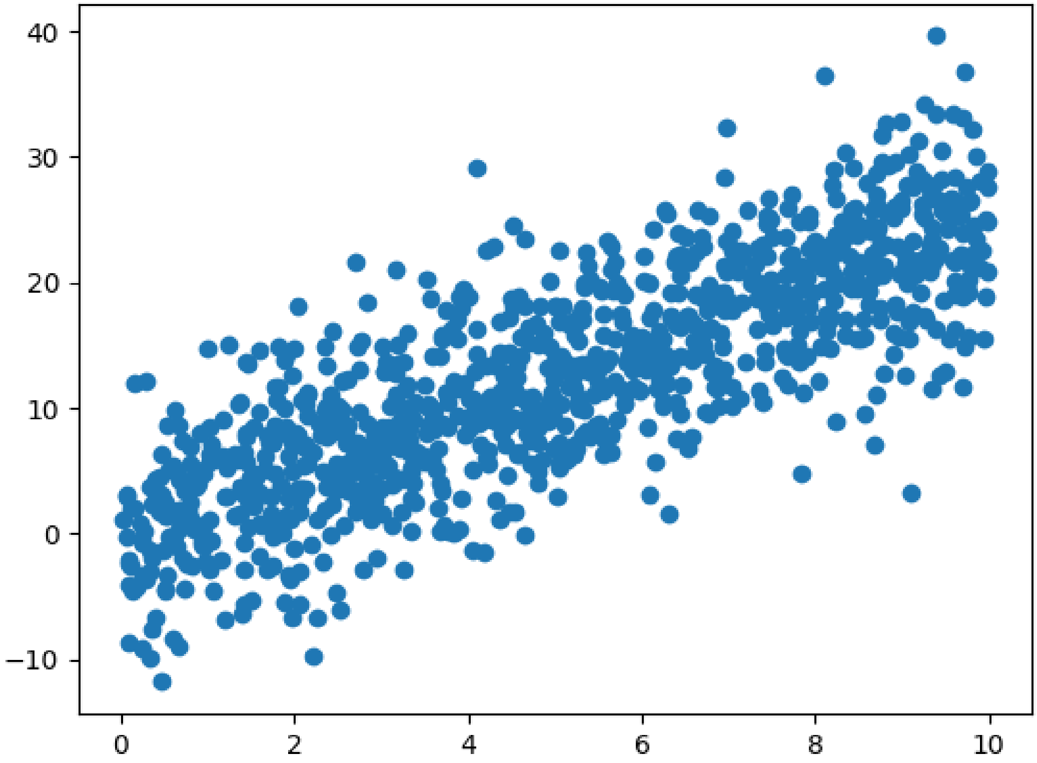


[ ]:

```
predictions=model(X_test)
plt.scatter(X,y)
plt.plot(X_test,predictions ,color='red')
plt.show()
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
<ipython-input-106-45c04227b892> in <cell line: 3>()
      1 predictions=model(X_test)
      2 plt.scatter(X,y)
----> 3 plt.plot(X_test,predictions ,color='red')
      4 plt.show()

/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py in plot(scalex,␣
 ↪scaley, data, *args, **kwargs)
   3576     **kwargs,
   3577 ) -> list[Line2D]:
-> 3578     return gca().plot(

   3579         *args,
   3580         scalex=scalex,
/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_axes.py in plot(self,␣
↪scalex, scaley, data, *args, **kwargs)
   1721         lines = [*self._get_lines(self, *args, data=data, **kwargs)]
   1722         for line in lines:
-> 1723             self.add_line(line)
   1724         if scalex:
   1725             self._request_autoscale_view("x")

/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_base.py in␣
↪add_line(self, line)
   2307             line.set_clip_path(self.patch)
   2308
-> 2309         self._update_line_limits(line)
   2310         if not line.get_label():
   2311             line.set_label(f'_child{len(self._children)}')

/usr/local/lib/python3.10/dist-packages/matplotlib/axes/_base.py in␣
↪_update_line_limits(self, line)
   2330         Figures out the data limit of the given line, updating self.
↪dataLim.
   2331         """
-> 2332         path = line.get_path()
   2333         if path.vertices.size == 0:
   2334             return

/usr/local/lib/python3.10/dist-packages/matplotlib/lines.py in get_path(self)
   1030         """Return the `~matplotlib.path.Path` associated with this line
↪"""
   1031         if self._invalidy or self._invalidx:
-> 1032             self.recache()
   1033         return self._path
   1034

/usr/local/lib/python3.10/dist-packages/matplotlib/lines.py in recache(self,␣
↪always)
    672         if always or self._invalidy:
    673             yconv = self.convert_yunits(self._yorig)
--> 674             y = _to_unmasked_float_array(yconv).ravel()
    675         else:
    676             y = self._y

/usr/local/lib/python3.10/dist-packages/matplotlib/cbook.py in␣
↪_to_unmasked_float_array(x)
   1343         return np.ma.asarray(x, float).filled(np.nan)
   1344     else:
-> 1345         return np.asarray(x, float)
   1346
   1347
```

```
/usr/local/lib/python3.10/dist-packages/torch/_tensor.py in __array__(self,␣
 ↪dtype)
   1149             return self.numpy()
   1150         else:
-> 1151             return self.numpy().astype(dtype, copy=False)
   1152 
   1153     # Wrap Numpy array again in a suitable tensor when done, to support
 ↪e.g.

RuntimeError: Can't call numpy() on Tensor that requires grad. Use tensor.
 ↪detach().numpy() instead.
```



```
[ ]:
```