bnn

November 27, 2024

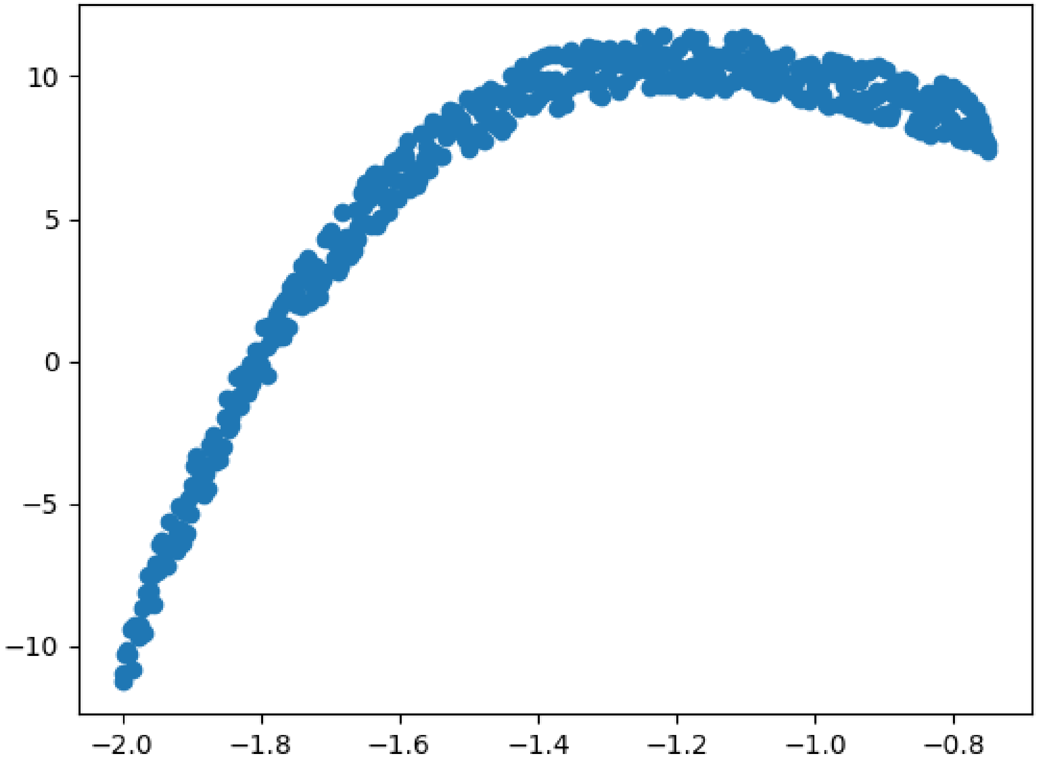
[1]:

```
import numpy as np
from sklearn import datasets
import torch
import torch.nn as nn
import torch.optim as optim
import torchbnn as bnn
import matplotlib.pyplot as plt
```

[3]:

```
# x = torch.cat((torch.linspace(-2, 0.5, int(0.75*500)), torch.linspace(1.7, 2, int(0.25*500))))
x = torch.linspace(-2,-0.75, 500)
y = x.pow(5) -10* x.pow(1) + 2*torch.rand(x.size())
x = torch.unsqueeze(x, dim=1)
y = torch.unsqueeze(y, dim=1)

plt.scatter(x.data.numpy(), y.data.numpy())
plt.show()
```



[9]:
```
def clean_target(x):
    return x.pow(5) -10* x.pow(1)+1
def target(x):
    return x.pow(5) -10* x.pow(1) + 2*torch.rand(x.size())
```

[4]:
```
model = nn.Sequential(
    bnn.BayesLinear(prior_mu=0, prior_sigma=0.1, in_features=1,␣
 ↪out_features=1000),
    nn.ReLU(),
    bnn.BayesLinear(prior_mu=0, prior_sigma=0.1, in_features=1000,␣
 ↪out_features=1),
)
```

[5]:
```
mse_loss = nn.MSELoss()
kl_loss = bnn.BKLLoss(reduction='mean', last_layer_only=False)
kl_weight = 0.01

optimizer = optim.Adam(model.parameters(), lr=0.01)
```

[6]:
```
for step in range(2000):
    pre = model(x)
    mse = mse_loss(pre, y)
    kl = kl_loss(model)
    cost = mse + kl_weight*kl

    optimizer.zero_grad()
    cost.backward()
    optimizer.step()

print('- MSE : %2.2f, KL : %2.2f' % (mse.item(), kl.item()))
```

```
- MSE : 0.94, KL : 8.45
```

[10]:

```
x_test = torch.linspace(-2, 2, 300)
y_test = target(x_test)

x_test = torch.unsqueeze(x_test, dim=1)
y_test = torch.unsqueeze(y_test, dim=1)
```

[11]:

```
models_result = np.array([model(x_test).data.numpy() for k in range(10000)])
models_result = models_result[:,:,0]
models_result = models_result.T
mean_values = np.array([models_result[i].mean() for i in␣
 ↪range(len(models_result))])
std_values = np.array([models_result[i].std() for i in␣
 ↪range(len(models_result))])
```

[ ]:

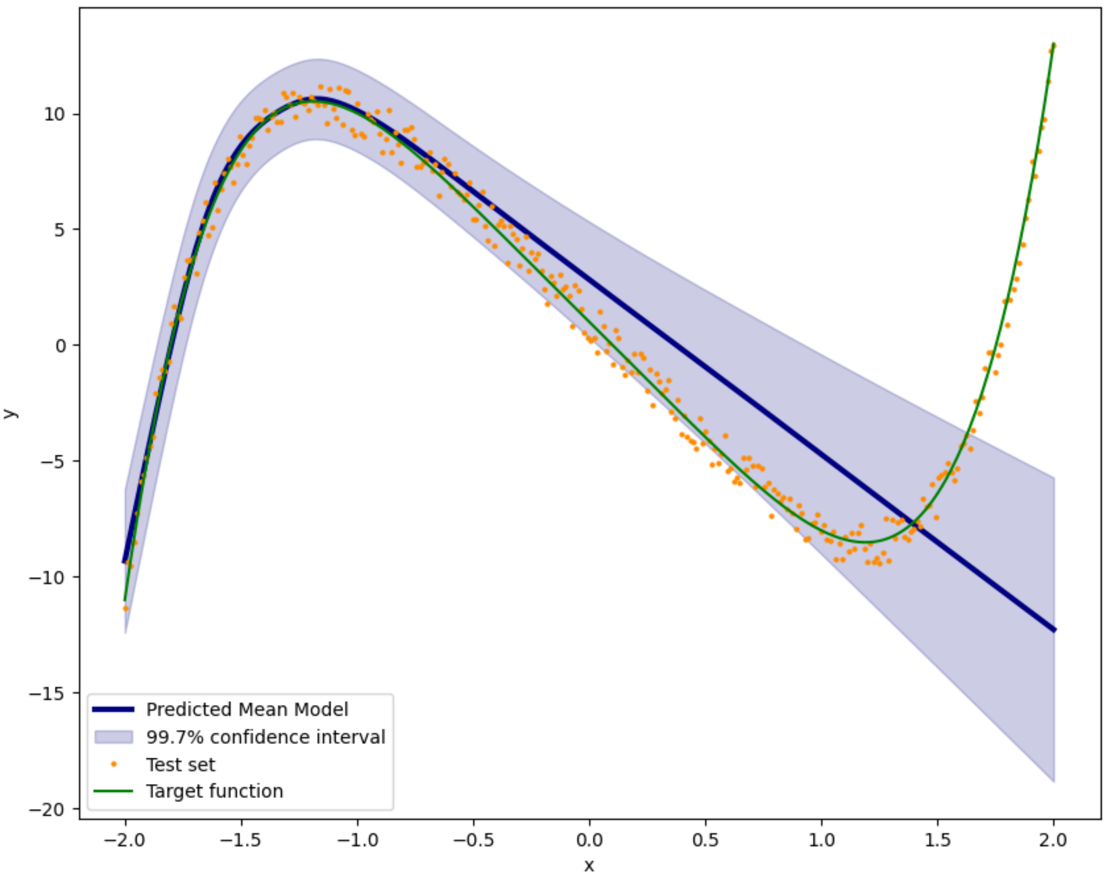
```
models_result.shape
```

[ ]:

```
(300, 10000)
```

[12]:

```
plt.figure(figsize=(10,8))
plt.plot(x_test.data.numpy(),mean_values,color='navy',lw=3,label='Predicted␣
 ↪Mean Model')
plt.fill_between(x_test.data.numpy().T[0],mean_values-3.
 ↪0*std_values,mean_values+3.0*std_values,alpha=0.2,color='navy',label='99.7%␣
 ↪confidence interval')
#plt.plot(x_test.data.numpy(),mean_values,color='darkorange')
plt.plot(x_test.data.numpy(),y_test.data.numpy(),'.
 ↪',color='darkorange',markersize=4,label='Test set')
plt.plot(x_test.data.numpy(),clean_target(x_test).data.
 ↪numpy(),color='green',markersize=4,label='Target function')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```


10
5
0
−5
−10
−15
−20
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x
Predicted Mean Model
99.7% confidence interval
Test set
Target function