01_gp_1d_inference

January 15, 2025

1 Univariate Gaussian Process Regression

1.1 Required Modules

[1]:
```
import sys
sys.path.append("..")

import torch
import pyro
import pyro.contrib.gp as gp
import pyro.infer
import pyro.optim
from pyro.infer import Trace_ELBO
pyro.set_rng_seed(0)
```

```
/home/ale/pythonEnv/tut/lib/python3.12/site-packages/tqdm/auto.py:21:
TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

1.2 Data Generation

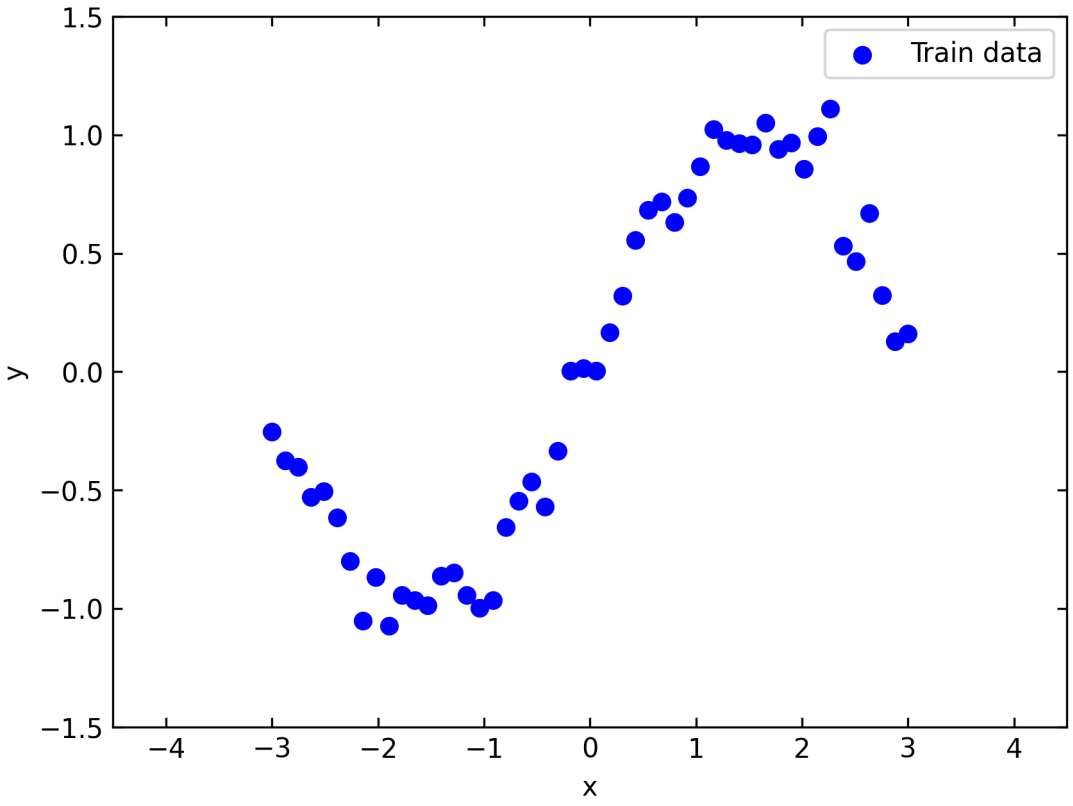
To get stated we are going to perform a GP regression over a 1D synthetic dataset, where $x \in [-3, 3]$ and data are taken from:

$$y = 1 \cdot sin(x) + \epsilon$$

where:

$$\epsilon \sim \mathcal{N}(0, 0.1)$$

[2]:
```
from src.data import generate_data, sine
from src.view import inspect_data

x_train, y_train = generate_data(-3, 3, 50, sine, 0.1, amp=1)
inspect_data(x_train=x_train, y_train=y_train)
```



1.3 GP Regression setting

The setting of GP regression formulates as:

$$y(x) = f(x) + \epsilon,$$

where:

$$f \sim \mathrm{GP}(m(x), k(x)),$$

and:

$$\epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

1.4 Initialising the Regression

GP regression reduces to finding the *hyperparameters* of the kernel through Maximum Likelihood Estimation. Since this process entails an optimisation problem, we need to set some initial guesses for the parameters.

[3]:
```
# initial guesses of kernel's parameters
var = torch.tensor(1.0) # C
length = torch.tensor(1.0) # l
noise = torch.tensor(0.1) # sigma
```

```
# load RBF kernel and GP regression
kernel = gp.kernels.RBF(input_dim=1, variance=var, lengthscale=length)
gpr = gp.models.GPRegression(x_train, y_train, kernel, noise=noise)
```

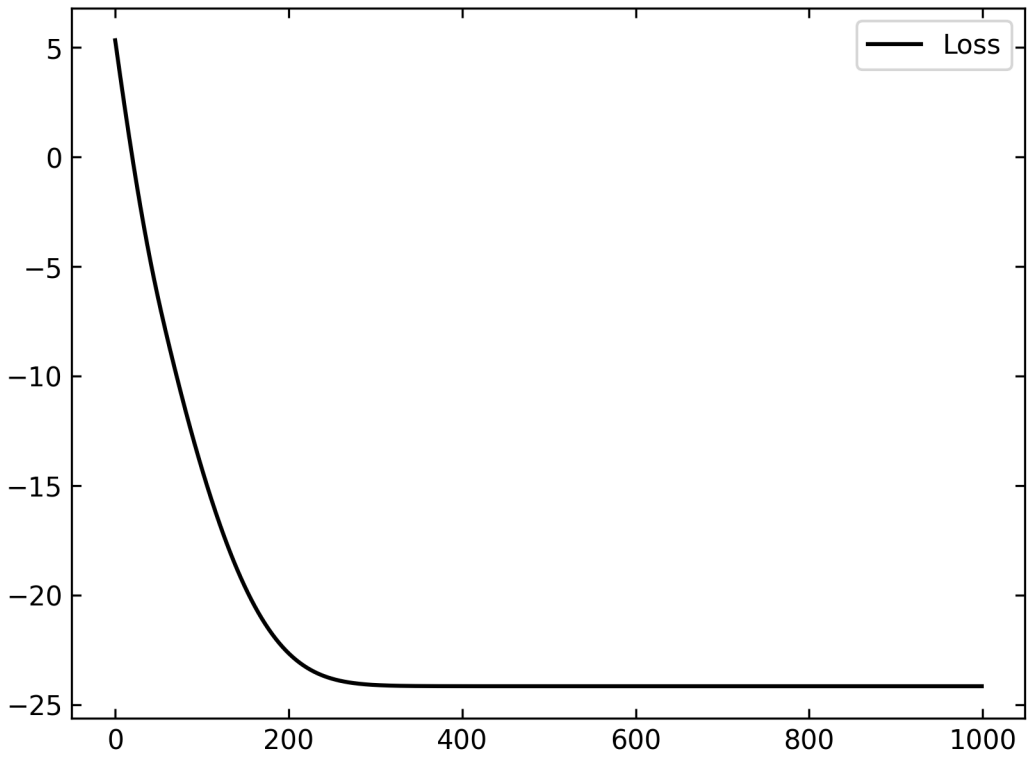
1.5 Solving the Regression Problem

[4]:
```
# load optimiser
learning_rate = 0.01
optimiser = pyro.optim.Adam({"lr": learning_rate})

# loss function
loss_fn = Trace_ELBO()
steps = 1000
```

[5]:
```
from src.view import inspect_loss
from src.inference import train_helper
# train and inspect loss function
loss = train_helper(gpr, optimiser, loss_fn, steps)
inspect_loss(loss)
```

```
--- Step 0 - Loss: 5.331584475692878
* Variance: 0.9900498390197754
* Lengthscale: 1.0100501775741577
* Noise: 0.09900498390197754

--- Step 100 - Loss: -14.241088034610115
* Variance: 0.6583241820335388
* Lengthscale: 1.6281861066818237
* Noise: 0.03769297897815704

--- Step 200 - Loss: -22.659131841728453
* Variance: 0.7875658273696899
* Lengthscale: 1.7293386459350586
* Noise: 0.018131211400032043

--- Step 300 - Loss: -24.107198993533956
* Variance: 0.9638161063194275
* Lengthscale: 1.8242852687835693
* Noise: 0.013148029334843159

--- Step 400 - Loss: -24.160679217761384
* Variance: 1.0812264680862427
* Lengthscale: 1.8766379356384277
* Noise: 0.012393265031278133

--- Step 500 - Loss: -24.161938666521678
* Variance: 1.1260730028152466
* Lengthscale: 1.8947904109954834
```

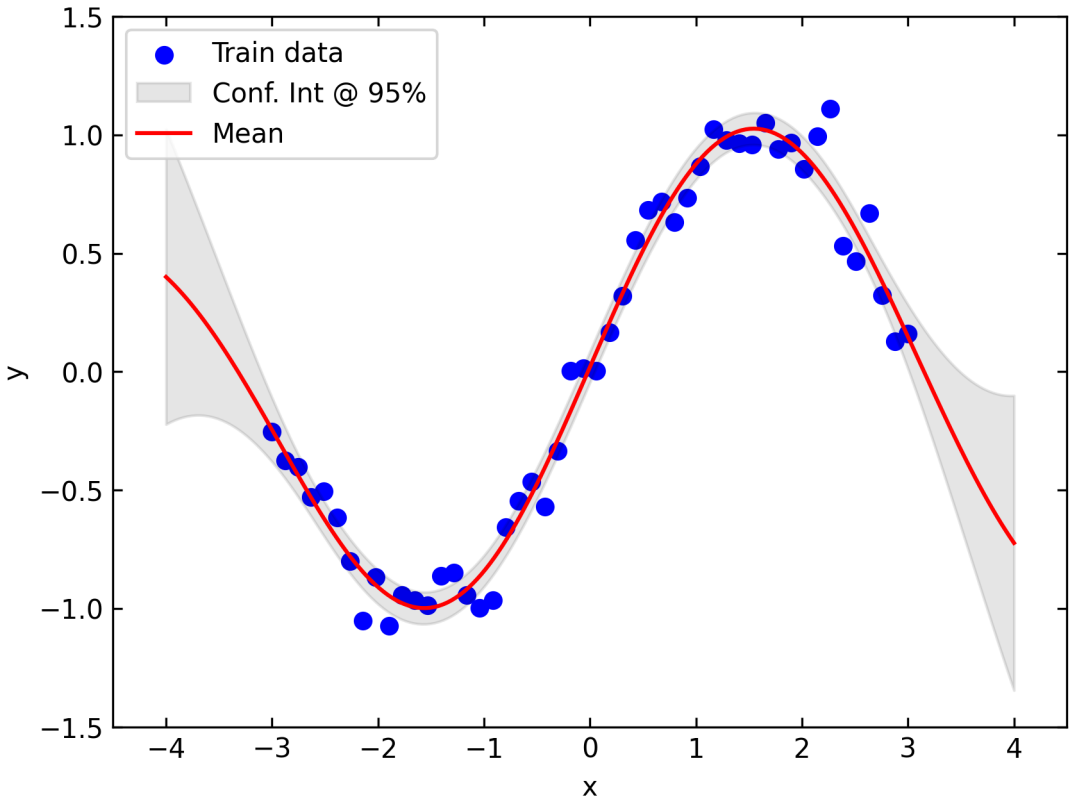
```
* Noise: 0.012331516481935978

--- Step 600 - Loss: -24.161993835584312
* Variance: 1.1374794244766235
* Lengthscale: 1.8992564678192139
* Noise: 0.012327130883932114

--- Step 700 - Loss: -24.16199557906114
* Variance: 1.139614462852478
* Lengthscale: 1.9000861644744873
* Noise: 0.012326613068580627

--- Step 800 - Loss: -24.161995611601654
* Variance: 1.139914631843567
* Lengthscale: 1.9002025127410889
* Noise: 0.012326572090387344

--- Step 900 - Loss: -24.161995612067315
* Variance: 1.1399459838867188
* Lengthscale: 1.900214672088623
* Noise: 0.012326560914516449

---------- Summary ----------
* Variance: 1.1399478912353516
* Lengthscale: 1.9002153873443604
* Noise: 0.012326554395258427
* Elaspsed time: 2.22 s
```



1.6 Making Predictions

To evaluate the above, we make an array of test data first, condition it to the training data, and finally inspect the result.

[6]:
```
# array of test data
x_test = torch.linspace(-4., 4., 200, dtype=torch.double)

# data conditioning
y_mean, y_cov = gpr(x_test, full_cov=True)
y_test_std = y_cov.diag().sqrt()
inspect_data(x_train=x_train, y_train=y_train,
             x_test=x_test, y_test=y_mean.detach(), y_test_std=y_test_std.
 ↪detach(),
             confidence=2)
```


Train data
Conf. Int @ 95%
Mean
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−4
−3
−2
−1
0
1
2
3
4
x