Fortgeschrittene Programmierung in der Physik SE

Institute of Theoretical Physics
Computational Physics

Introduction to Baysian Neural Networks

Advisor:
Univ.-Prof. Dipl.-Phys. Dr. Wolfgang von der Linden

Tobias Leitgeb
Mat.Nr. 12006992

Summer term 2024

1 Mathematical basics

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\theta'})} \tag{1}$$

1.1 Baysian Neural Networks

Priors over wheigts, baysian inference, uncertainty quatification

1.2 Variational Inference

2 Setting up the model

A very default prior for the parameters of the BNN which has proven to work relatively well [**BNNTut**] is:

$$\boldsymbol{W} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{2}$$
$$\boldsymbol{b} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{3}$$

where we just use two normal distributions as priors for the wheigts and biases. In the used probabilistic framework NumPyro we can define the priors the following way:

```
W = numpyro.sample(f"W{i}", dist.Normal(0, 1).expand(n_in, n_out))
b = numpyro.sample(f"b{i}", dist.Normal(0, 1).expand((n_out, )))
```

Here $\boldsymbol{W}_i$ and $\boldsymbol{b}_i$ are the corresponding parameters for the i-th layer. Generally, other priors can also be used, however the normal distributions lead to good results. Priors, Network, inference, Variational inference, talk about numpyro

3 Numerical examples

maybe normal NN against BNN
transistor data, oscillator data

3.1 Transistor surrogate model/Transformer model akash

Here the data set consists of an inputs $\boldsymbol{x}$, which specify the circuit layout and of the envelope of the fourier transformed output of the simulation data $\boldsymbol{y}$ over a specified frequency range.

$$X = \begin{bmatrix} \boldsymbol{x}_1^T, \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{bmatrix}, \ Y = \begin{bmatrix} \boldsymbol{y}_1^T \\ \boldsymbol{y}_2^T \\ \vdots \\ \boldsymbol{y}_N^T \end{bmatrix} \tag{4}$$

with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $\boldsymbol{y}_i \in \mathbb{R}^p$. Therefore, we get two matrices $\boldsymbol{X}, \boldsymbol{Y}$ with shapes (N, d) and (N, p), where N is the number of training points, d is the dimensionality of the input and p is the number of frequency outputs. The BNN is designed as: $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}^p$ to yield an output for the whole frequency range.

4 Code

```
"import jax\n",
"import jax.numpy as jnp\n",
"import jax.random as jr \n",
"import numpyro.distributions as dist\n",
"\n",
"from numpyro.infer import SVI, Trace_ELBO, Predictive\n",
"from numpyro.optim import Adam\n",
"\n",
"import matplotlib.pyplot as plt\n",
"import seaborn as sns\n",
"\n",
"from typing import Tuple, List, Callable\n",
```

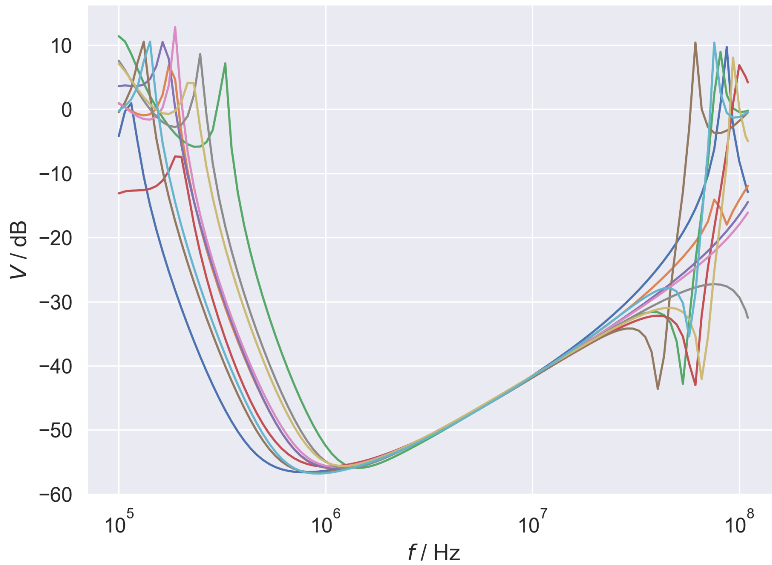

Figure 1: Samples from the training data for frequency domain

```
    "import sys\n",
    "sys.path.append(\"../\")\n",
    "from utils import plot_testset, get_data, minmaxrmspe, plot_samples\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "metadata": {},
```

Listing 1: Example Python Code

[**alvarez2012kernels**] [**raissi2017physicsIDL**]

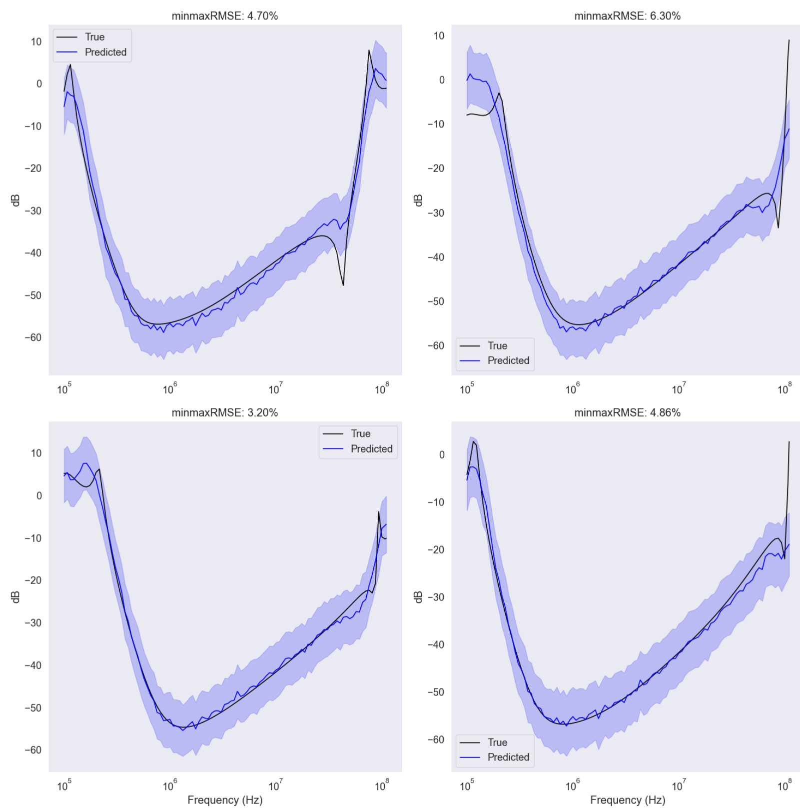

Figure 2: Samples from the training data for frequency domain