**Bayesian Inference**

Bayesian inference uses Bayes’ rule to compute the posterior distribution of model parameters by incorporating prior knowledge to the likelihood of data, which is given by

\[
P(\theta|D) = \frac{P(D|\theta)P(\theta)}{\int_\theta P(D|\theta)P(\theta)\,d\theta} \propto P(D|\theta)P(\theta) \tag{1}
\]

, where \(D\) is the data set, \(\theta\) is the model parameter, \(P(\theta)\) is the prior, \(P(D|\theta)\) is the likelihood of data and \(P(\theta|D)\) is the posterior distribution.

**References**


**Contact Information**

Email kx216@cam.ac.uk  
Phone +44 770 771 9899

**Progress and Next Step**

By the day of poster session, HMC sampler has been implemented in Julia and a wrapper of common distributions is also available. Some handwritten models were used to test these two components. In addition, the student is also contributing to the documentation work.

The next step for the project is to implement the compiler for HMC in the PP framework. When this is done, it will be embedded into Turing.jl, working with existing inference algorithms. More evaluations on the performance of HMC and Turing.jl will be conducted.

**Example 1 - Univariate Gaussian**

A univariate Gaussian with conjugate priors and data \(D = \{1.1, 1.1, 0.9, 1.3, 0.6\}\) can be defined and learnt by the code below.

\[
D = \{1.1, 1.1, 0.9, 1.3, 0.6\}
\]

@model gauss begin
  @param σ ~ InverseGamma(2, 3)
  @param μ ~ Normal(0, \sqrt{σ})
  for x in D
    @observe x ~ Normal(μ, \sqrt{σ})
  end
  @predict μ σ
end
samples = sample(gauss, HMC(500))

Figure 2: Trace of \(μ\) and \(σ\)

**Example 2 - Bayesian Neural Network**

PP framework can also be used to train neural networks by interpreting the loss function as likelihood and the regularisation term as prior.

To train a Bayesian neural network (BNN) with structure in Figure 3, learning the exclusive-or function, the program on the right can be used.

By the day of poster session, HMC sampler has been implemented in Julia and a wrapper of common distributions is also available. Some handwritten models were used to test these two components. In addition, the student is also contributing to the documentation work.

The next step for the project is to implement the compiler for HMC in the PP framework. When this is done, it will be embedded into Turing.jl, working with existing inference algorithms. More evaluations on the performance of HMC and Turing.jl will be conducted.

**Example 2 - Bayesian Neural Network**

PP framework can also be used to train neural networks by interpreting the loss function as likelihood and the regularisation term as prior.

To train a Bayesian neural network (BNN) with structure in Figure 3, learning the exclusive-or function, the program on the right can be used.

\[
xs = \{(0; 0); (0; 1); (1; 0); (1; 1)\}
\]

@model bnn begin
  weights = TArray(Float64, 9)
  @param σ ~ InverseGamma(2, 3)
  for w in weights
    @param w ~ Normal(0, \sqrt{σ})
  end
  for i in 1:4
    y = nn(xs[i], weights)
    @observe ts[i] ~ Bernoulli(y)
  end
  @predict weights
end
samples = sample(bnn, HMC(500))

Figure 3: Structure of the Bayesian neural network

**Contact Information**

Email kx216@cam.ac.uk  
Phone +44 770 771 9899

**Progress and Next Step**

By the day of poster session, HMC sampler has been implemented in Julia and a wrapper of common distributions is also available. Some handwritten models were used to test these two components. In addition, the student is also contributing to the documentation work.

The next step for the project is to implement the compiler for HMC in the PP framework. When this is done, it will be embedded into Turing.jl, working with existing inference algorithms. More evaluations on the performance of HMC and Turing.jl will be conducted.