Come join the fun!

Repo

github.com/normal-computing/posteriors

Paper

openreview.net/forum?id=fifXzmzeGy

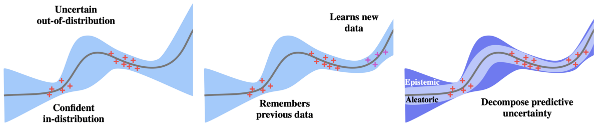
posteriors

Uncertainty quantification with PyTorch

Samuel Duffield
sam@normalcomputing.ai

with Kaelan Donatella, Johnathan Chiu, Phoebe Klett, Daniel Simpson and open source contributors

Bayesian Learning



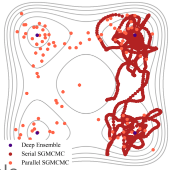
Generalization ✅ **Continual learning ✅** **Decomposition of uncertainty ✅**

Unification of optimizers, samplers and (Bayesian) deep ensembles

Formulate Bayesian samplers in continuous-time:

$$dz = [D+Q]N^{-1}\nabla \log \pi(z)dt + \sqrt{2\mathcal{T}D}dw$$

for symmetric D, skew-symmetric Q and z often on an extended space, e.g. $z=(\theta, m)$ for momenta m.



$\mathcal{T} = N^{-1} \implies$ Posterior sampling
(Stochastic gradient MCMC)

$\mathcal{T} = 0 \implies$ Optimization

Many parallel chains $\implies$ Bayesian deep ensemble
(Parallel stochastic gradient MCMC)

Many parallel chains & $\mathcal{T} = 0 \implies$ Deep ensemble

posteriors is an open source PyTorch package for scalable Bayesian learning.
The key features outlining its philosophy are:

- **Composability**: Use with transformers, lightning, Llama, torchopt, pyro, ….
- **Extensible**: Easily add new methods to within the transform framework
- **Functional**: JAX-like, easier to test, compose, debug and closer to maths
- **Scalable**: Enforced support for mini-batches
- **Swappable**: Easily change methods

```python
# Load classifier model
# Load dataloader
num_data = len(dataloader.dataset)

def log_posterior(params, batch):
    inputs, labels = batch
    logits = torch.func.functional_call(model, params, inputs)
    log_post_val = (
        -torch.nn.funtional.cross_entropy(logits, labels)
        + posteriors.diag_normal_log_prob(params) / num_data
    )
    return log_post_val, logits  # Return value and auxiliary info


transform = posteriors.vi.diag.build(
    log_posterior, torchopt.adam(), temperature=1/num_data
)  # Can swap out for any posteriors algorithm

state= transform.init(params)

for batch in dataloader:
    state, aux = transform.update(state, batch)
```

Bayesian Llama 3 🦙

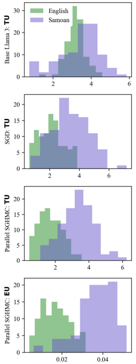
A Bayesian model allows you to decompose predictive uncertainty

Total $= \mathbf{TU} = H[p(y \mid x)]$

Aleatoric $= \mathbf{AU} = \mathbb{E}_{p(\theta \mid y_{1:N})}[H[p(y \mid x, \theta)]]$

Epistemic = $\mathbf{TU} - \mathbf{AU}$

$H[p]$ = entropy of p



Epistemic uncertainty is a better indicator of hallucinations as semantic uncertainty (synonyms, starts of sentences etc) is captured by aleatoric uncertainty

Right: Bayesian LLM fine-tuned on textbooks.
Different uncertainty metrics as classifiers of out of distribution (Samoan) inputs.

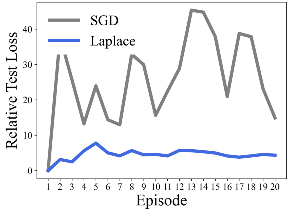
Ideal case would be a complete separation of green and purple histograms.

Continual Learning 🔄

$$p(\theta \mid y_{1:N}) \propto \underbrace{p(\theta)p(y_{1:N} \mid \theta)}_{\text{offline}} \propto \underbrace{p(\theta \mid y_{1:N-1})p(y_N \mid \theta)}_{\text{online}},$$

Exact Bayes has no forgetting, in the sense that it values y_N equal to y_1.

Simple Bayesian approximation mitigates forgetting, here in continual training of Llama 2



SGD Llama 2 forgets earlier books 📚 (episodes) as it reads new ones.
Replace point estimates with Laplace posteriors and Llama 2 has improved performance on earlier books 📚

NORMAL Computing