A Dive into Probabilistic Optimization Patterns

Saharsha Tiwari
saharsha.tiwari@bison.howard.edu

Abstract

This term paper delves into probabilistic optimization techniques, specifically focusing on the implementation of probabilistic backpropagation and probabilistic loss functions. The objective of this project is to explore how probabilistic methods can enhance traditional optimization algorithms by introducing uncertainty into the model's learning process. By integrating probabilistic elements into the backpropagation framework, this research aims to bolster model robustness and adaptability, particularly in the presence of noisy or incomplete data. The probabilistic loss functions will be designed to account for uncertainty, providing more flexible and reliable gradient descent pathways. Through experimentation and analysis, this paper will demonstrate the effectiveness of these probabilistic approaches in optimizing machine learning models.

1 Introduction/Motivation

The resurgence of neural networks (NNs) has revolutionized various fields, achieving state-of-the-art results across numerous supervised learning problems. Their success can be attributed to their ability to learn complex function approximations from large datasets through stochastic optimization and the backpropagation (BP) algorithm. Despite these advancements, traditional BP methods present several challenges, including the necessity for extensive hyperparameter tuning, the inability to provide calibrated probabilistic predictions, and a propensity for overfitting the training data.

In contrast, Bayesian approaches to learning neural networks offer a promising solution to these challenges. By inherently incorporating uncertainty into parameter estimates, Bayesian methods can improve robustness and provide more reliable predictions. These techniques have the potential to automatically infer hyperparameter values and mitigate overfitting through a probabilistic framework. However, existing Bayesian learning methods often struggle with scalability, limiting their applicability to larger datasets and more complex network architectures.

This project aims to address these limitations by implementing probabilistic backpropagation (PBP), a method designed to enhance the scalability and efficiency of Bayesian neural network learning. PBP operates similarly to traditional backpropagation but incorporates a forward propagation of probabilities and a backward computation of gradients. This integration allows for accurate estimates of posterior variance on network weights, enabling the model to capture uncertainty effectively.

Through a series of experiments on real-world datasets, this research will evaluate the performance of PBP in terms of training speed and predictive accuracy compared to conventional techniques

2 Baseline or Initial Experiments

For the initial experiments, we implemented a classic gradient descent backpropagation for a neural network with a simple linear function $f(x) = Wx$, where W is the weight matrix. The model was trained using CrossEntropy loss on the Iris Dataset. This initial setup allowed us to validate the effectiveness of traditional backpropagation in handling the classification tasks.

2.1 Classic Backpropagation

We manually implemented the classic backpropagation with gradient descent for the CrossEntropy loss. The results were promising, with the model achieving good predictive accuracy and convergence over several iterations. The use of traditional methods, such as the mean squared error (MSE) loss, demonstrated solid performance in estimating weights and minimizing error.

2.2 Probabilistic Backpropagation

We then implemented probabilistic backpropagation (PBP), incorporating Gaussian approximations for the weight matrix W. This probabilistic approach allowed us to account for uncertainty in the model parameters, making the training process faster than the manual implementation. However, the PBP model displayed some accuracy issues, which could be attributed to the prior distributions used for the weights. Despite the speed improvements, the model's predictions lacked precision compared to the classic approach.

These early observations suggest that the choice of priors in probabilistic models may significantly impact predictive accuracy. Future experiments will focus on refining the prior distributions for the weights and further optimizing the probabilistic loss functions to enhance both speed and accuracy. Metrics such as root mean square error (RMSE) and log-likelihood will continue to be used to evaluate the performance and uncertainty estimates provided by PBP. I will also make it agnostic to any loss functions.

3 Final Experiments

The final set of experiments will focus on implementing probabilistic backpropagation (PBP) in PyTorch, with a flexible framework that allows for selective probabilistic propagation of parameters. Specifically, we aim to implement an option for each parameter, such as `required_prob_grad = True`, which will determine whether the parameter should be propagated probabilistically. This flexibility will enable us to explore hybrid propagation techniques, where some parameters follow traditional backpropagation while others are treated probabilistically.

This hybrid approach is expected to strike a balance between computational efficiency and predictive accuracy. In scenarios where certain parameter updates are more computationally expensive, the probabilistic gradient calculation will be employed to accelerate the process while maintaining robust uncertainty estimates. For simpler parameter updates, traditional backpropagation will be used to avoid unnecessary complexity.

By testing various combinations of probabilistic and traditional propagation, we aim to evaluate the impact on both training speed and model performance. The key hypothesis is that using probabilistic gradient calculations selectively, particularly for parameters where traditional calculations are more compute-intensive, will lead to improved training times without sacrificing accuracy.

This implementation will be tested on complex problems where a multi layer perception is used, comparing the hybrid approach to full probabilistic and full traditional methods in terms of different log functions and overall training time.

4 Final Goals & Evaluation

By the end of the semester, the primary goal is to successfully implement and evaluate probabilistic backpropagation (PBP) within the PyTorch framework, with an option to selectively propagate parameters probabilistically. The key objectives are twofold: (1) PBP should match the accuracy of traditional backpropagation within a margin of ±10%, and (2) the hybrid or probabilistic approach should demonstrate at least a 5x speed improvement over the traditional method. These performance metrics are derived from the results of the initial experiments, which suggested a promising trade-off between speed and accuracy.

4.1 Quantitative Metrics

The success of the experiments will be measured using the following quantitative metrics:

- **Accuracy/Entropy:** Measured in terms of accuracy and cross-entropy, with the target within ±10% of traditional backpropagation methods.
- **Training Time:** The hybrid probabilistic approach should demonstrate at least a 5x improvement in training speed compared to the full traditional backpropagation method.
- **Uncertainty Estimation:** Evaluation of posterior variance in the probabilistic models to ensure that the uncertainty estimates remain reliable when compared to fully probabilistic methods.

4.2 Decomposition of Goals

To ensure a clear progression toward these goals, the final objective has been broken down into sub-goals of varying difficulty:

- **Result A:** Achieve a baseline implementation of probabilistic backpropagation with hybrid propagation in PyTorch, expected to yield accuracy results within ±10% of traditional backpropagation (based on the initial experiments).
- **Result B:** Achieve a minimum 5x speed improvement in training time while maintaining the accuracy target (this is more ambitious but achievable with further optimizations in the selective propagation mechanism).

In addition to these core metrics, qualitative analysis will be conducted to assess the trade-offs between accuracy and speed in varying settings in terms of size, complexity, or loss functions used, as well as the robustness of the hybrid approach in different datasets and network architectures. These results will provide insights into the viability of probabilistic methods for real-world machine-learning applications. I also plan to focus on vanishing gradient problems, where probabilistic backpropagation might be a way to go forward.

5 Related Work

The development of probabilistic methods for training neural networks has garnered significant attention in recent years. One of the foundational approaches is Probabilistic Backpropagation (PBP) introduced by Hernández-Lobato and Adams Hernández-Lobato & Adams (2015). PBP addresses key limitations of traditional backpropagation by providing a scalable framework for learning Bayesian neural networks. By computing a forward propagation of probabilities and subsequently performing backward gradient computation, PBP enhances both efficiency and accuracy in predicting posterior distributions, thus mitigating issues of overfitting often associated with large neural networks.

In parallel, Expectation Backpropagation (EBP), proposed by Soudry et al. Soudry et al. (2014), offers a parameter-free training method that approximates the posterior distribution of weights using a mean-field approach. This technique is particularly advantageous in scenarios with limited computational resources, as it alleviates the need for hyperparameter tuning typically required in traditional training methods. EBP has demonstrated promising results across various tasks, highlighting its potential for broader applications in machine learning.

Further advancing the discourse on PBP, Benatan et al. Benatan & Pyzer-Knapp (2018) examine practical considerations for implementing probabilistic backpropagation in real-world settings. Their work underscores the importance of initialization strategies and convergence criteria, addressing challenges faced by practitioners when deploying PBP in complex environments. This exploration is crucial for ensuring robust and reliable implementations of Bayesian neural networks.

Collectively, these contributions reflect the ongoing evolution of probabilistic methods for neural network training, each addressing distinct challenges related to scalability, efficiency, and practical deployment. This body of work informs our exploration of advanced techniques for Bayesian learning, setting the stage for further innovations in the field.

6 Data & Technical Requirements

This project requires a combination of datasets and technical resources to effectively implement and evaluate the probabilistic optimization techniques described. The primary data sources and technical requirements are outlined below.

6.1 Datasets

The research will utilize several publicly available datasets to assess the performance of the implemented algorithms. The key datasets include:

- **Iris Dataset**: This classic dataset will be used for initial experiments, focusing on classification tasks. It consists of 150 samples with four features, representing different species of Iris flowers.
- **MNIST Dataset**: A large collection of handwritten digits, the MNIST dataset will serve as a benchmark for evaluating the effectiveness of the probabilistic backpropagation technique on a more complex classification problem.
- **Housing Prices Dataset**: This dataset contains information on various housing attributes, such as square footage, number of bedrooms, and location, alongside their corresponding sale prices. It will be used to assess regression tasks and evaluate the effectiveness of the proposed probabilistic methods in predicting housing prices.

Access to these datasets is available through platforms such as UCI Machine Learning Repository (n.d.) and Kaggle (n.d.).

6.2 Technical Requirements

The implementation of this project will rely on several software libraries and tools:

- **Programming Language**: Python will be the primary language used for development due to its extensive support for scientific computing and machine learning.
- **Machine Learning Libraries**:
  - **PyTorch**: An essential library for model training, PyTorch will be the primary framework used for this project. It will allow for the implementation of probabilistic backpropagation, enabling flexible model development and experimentation. Additionally, PyTorch will be used to compare performance metrics and assess scalability across different neural network architectures.
- **Data Processing Libraries**: Libraries such as NumPy and Pandas will be employed for data manipulation and preprocessing tasks, ensuring that datasets are adequately prepared for model training.
- **Visualization Tools**: Matplotlib and Seaborn will be used for data visualization and result interpretation, allowing for clearer communication of findings and comparisons between methods.

6.3 Computational Resources

The experiments will be conducted on a system equipped with the following minimum specifications to ensure efficient processing and training times:

- **Processor**: Intel Core i7 or equivalent for multi-threaded processing capabilities.
- **RAM**: A minimum of 16 GB to handle larger datasets and complex model architectures effectively.
- **GPU**: A dedicated NVIDIA GPU (e.g., GTX 1060 or better) to accelerate training processes, especially for deep learning models.

These resources will facilitate the implementation of probabilistic methods in neural networks and support comprehensive evaluations to achieve the project goals.

References

M. Benatan and E. O. Pyzer-Knapp. Practical considerations for probabilistic backpropagation. In *Third Workshop on Bayesian Deep Learning (NeurIPS 2018)*, Montréal, Canada, 2018. URL `http://bayesiandeeplearning.org/2018/papers/99.pdf`.

J. M. Hernández-Lobato and R. P. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. *arXiv*, February 2015. URL `https://arxiv.org/abs/1502.05336`.

Kaggle. Kaggle: Your machine learning and data science community, n.d. URL `https://www.kaggle.com/`.

Daniel Soudry, Itay Hubara, and Ron Meir. Expectation backpropagation: parameter-free training of multilayer neural networks with continuous or discrete weights. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pp. 963–971, Cambridge, MA, USA, 2014. MIT Press.

UCI Machine Learning Repository. Uci machine learning repository, n.d. URL `https://archive.ics.uci.edu/`.