Bayesian Introspection

Riccardo Ruggieri

May 2024

Contents

1 Introduction 2

2 Preliminaries 3

2.1 Approximation of the Evidence 3

2.2 Approximation of the Posterior 3

2.2.1 Approximation of the Hessian 4

2.3 Hyperparameter Tuning - Prior Precision 4

2.4 Why sampling? 4

3 Epistemic Uncertainty and Next Token Prediction 4

4 Experiments 5

4.1 Sampling Procedures 5

4.2 Task 5

4.3 Results 5

5 Prior Work 6

6 Conclusion 6

A Fisher Matrix and Laplace Approximation 9

B Approximations of Expected Fisher Matrix 9

Abstract

This paper introduces a method for enhancing token prediction accuracy by integrating Bayesian principles into LLMs routines. We employ Laplace approximation as a means to incorporate Bayesian thinking within our predictive models. Through empirical evaluation, we show that our approach achieves comparable performance to standard *temperature sampling*, a common subroutine employed in many state-of-the-art techniques. Our findings suggest that Bayesian integration holds promise for improving *safe* token prediction tasks without compromising on performance of pre-trained LLMs.

1 Introduction

The advent of *Large Language Models* (LLMs) has revolutionized the field of natural language processing. By leveraging billions or even trillions of parameters and complex neural architectures, LLMs have achieved unprecedented performance on tasks [Bubeck et al., 2023] such as machine translation, text summarization, sentiment analysis, and dialogue generation. The success of these models can be attributed to their ability to capture and generalize from the vast amounts of textual data they are trained on.

However, despite their impressive capabilities, LLMs often suffer from overconfidence and hallucination [Miao et al., 2021, Li et al., 2023]. This means they tend to generate fluent and plausible, yet inaccurate or inconsistent text. For example, an LLM may confidently provide an incorrect answer to a factual question, or contradict itself across multiple responses. This unreliable behavior limits the practical applicability of LLMs in domains that require a high degree of accuracy and consistency, such as question answering or task-oriented dialogue.

To address this limitation, we propose leveraging the model's *epistemic* uncertainty [Kendall and Gal, 2017] to improve answer quality. Unlike *aleatoric* uncertainty, which arises from inherent randomness in the data, epistemic uncertainty captures the model's lack of knowledge due to insufficient training data or model capacity. By making the model aware of its own knowledge gaps and encouraging it to critically examine its outputs, we aim to reduce overconfidence and improve reliability.

In particular, our approach is inspired by the *Bayesian model average* (BMA) principle [Wilson and Izmailov, 2020], without requiring updates of the model parameters. Concretely, we approximate the posterior distribution of a pre-trained model using a publicly available dataset (which should represent the dataset on which the model has been trained) and using the *Laplace Approximation* technique [MacKay, 1992]. This posterior distribution of the weights of the pre-trained model can be interpreted as the epistemic uncertainty of the model, i.e., how much the model is uncertain of its own knowledge. Subsequently, we exploit this information in order to generate answers which reflect the amount of epistemic uncertainty of the model w.r.t. the training given. Hence we aggregate the answers and choose the best answer based on criteria such as majority vote.

We conduct a preliminary empirical evaluation on a well known truthfulness benchmark. The results show that our uncertainty-aware decoding approach is promising and it is worth further investigation.

2 Preliminaries

In this section, we introduce some notation and basic notions regarding both neural networks in general and *Laplace approximation* [Daxberger et al., 2022], a tool to effectively approximate the posterior distribution of the model. Given an *iid* dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ where $x_n \in \mathbb{R}^N$ and $y_n \in \mathbb{R}^M$, and a model $f_\theta : \mathbb{R}^N \to \mathbb{R}^M$, its parameters $\theta \in \mathbb{R}^P$ are trained to minimize the regularized empirical risk $r(\theta) + l(x_n, y_n)$ on the given dataset, i.e.

$$\theta_{MAP} = \arg\min_{\theta \in \mathbb{R}^P} \mathcal{L}_\theta(\mathcal{D}) = \arg\min_{\theta \in \mathbb{R}^P} \left(r(\theta) + \sum_{n=1}^{N} l_\theta(x_n, y_n) \right)$$

where, from a Bayesian perspective, $l(x_n, y_n)_\theta = -\log p(y_n | f_\theta(x_n))$ corresponds to the negative log-likelihood, and $r(\theta) = -\log p(\theta)$ is the negative log-prior.

2.1 Approximation of the Evidence

Through these lens, the training loss can be interpreted as the negative log-posterior. This means that, by minimizing this loss, we are finding the maximum-a-posteriori (MAP). By Bayes formula,

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int_\Theta p(\mathcal{D}|\theta')p(\theta')\, d\theta'} = \frac{e^{-\mathcal{L}_\theta(\mathcal{D})}}{Z}$$

Let us define $h(\theta) := e^{-\mathcal{L}_\theta(\mathcal{D})}$. If we approximate $\log h(\theta)$ using the Taylor expansion (up to the second order) around θ_{MAP} we get:

$$\log h(\theta) \cong \log h(\theta_{MAP}) - \frac{1}{2}(\theta - \theta_{MAP})^T \Lambda^{-1} (\theta - \theta_{MAP}) \ , \tag{1}$$

where $\Lambda = -\nabla_\theta^2 \log h(\theta_{MAP})$. Notice that the first term vanishes since we are expanding around a MAP point, where the gradient is zero.

Now, we notice that, with this approximation, we can find a closed form for the model evidence Z (i.e., the normalizing constant in the Bayes formula):

$$Z = \int_\Theta e^{\log h(\theta')}\, d\theta' \cong h(\theta_{MAP}) \int_\Theta e^{-\frac{1}{2}(\theta' - \theta_{MAP})^T \Lambda^{-1} (\theta' - \theta_{MAP})\, d\theta'}$$

$$= h(\theta_{MAP}) (2\pi)^{\frac{d}{2}} det(\Lambda)^{\frac{1}{2}} \ ,$$

where, in the last equality, we used the properties of d-dimensional multivariate Gaussians. This is the so called *approximation of the evidence.*

2.2 Approximation of the Posterior

Now, let us apply again the Taylor approximation described in (1) to the Bayes formula:

$$p(\theta|\mathcal{D}) = \frac{e^{\log h(\theta)}}{Z} \cong \frac{1}{\sqrt{(2\pi)^d det(\Lambda)}} e^{-\frac{1}{2}(\theta - \theta_{MAP})^T \Lambda^{-1} (\theta - \theta_{MAP})}$$

This is the so-called Laplace Approximation (LA). We notice that, if we use this approximation, the posterior is a Gaussian distribution:

$$p(\theta|\mathcal{D}) \sim \mathcal{N}_\theta(\theta_{MAP}, \Lambda^{-1})$$

where $\Lambda = \nabla_\theta^2 \mathcal{L}(\mathcal{D}, \theta)_{\theta = \theta_{MAP}}$ is the Hessian of the loss.

In the context of a pre-trained LLM, this means that we can interpret the weight of the pre-trained model as the mean vector of the Gaussian distribution θ_{MAP}, while we need to estimate the Hessian Λ in order to get covariance matrix. Our approach is to consider an open source dataset (such as the Open Web Text dataset) and interpret this dataset as if it was the dataset used in the pre-training phase of the LLM.

2.2.1 Approximation of the Hessian

Exploiting epistemic uncertainty requires reliable knowledge of the variance of the distribution characterizing the model. As discussed in the preceding section, we aim to compute the Hessian matrix in the context of the Laplace Approximation (LA). However, for large language models (LLMs), the sheer number of parameters makes this computation impractical. Various approximations of the Hessian exist, such as the Fisher approximation, the Expected Fisher method, and the diagonal approximation (for further discussions, see Appendix A). Despite these methods, their computational complexity remains prohibitive for our purposes (see Appendix B). Therefore, we adopt a simpler routine based on the Monte-Carlo estimation of the Fisher matrix. Thus, the term *posterior sampling* must be interpreted in the sense of a Laplace approximation on the last layer of the model.

2.3 Hyperparameter Tuning - Prior Precision

As highlighted by Daxberger et al. [2022], modern Laplace approximation heavily relies on the tuning of prior variance/precision parameters. We set the prior precision at 10^2, 10^3, 10^4 (higher prior precision means lower exploration in the sense of posterior sampling) over different test scenarios, aiming for a compromise between exploration and exploitation within the context of posterior sampling. Nonetheless, more principled approaches to prior precision tuning such as marginal likelihood and grid search optimization are also available (see [Daxberger et al., 2022]).

2.4 Why sampling?

The concept of a sampling subroutine is well-established in the realm of LLMs. Numerous state-of-the-art techniques [Wang et al., 2023, Aggarwal et al., 2023, Kadavath et al., 2022] which prioritize accuracy metrics adopt sampling (in a broader sense) as a main paradigm. This can be achieved by prompting the model to generate multiple responses, from which the optimal one is selected either through a majority vote or implicit aggregation within the model itself.

3 Epistemic Uncertainty and Next Token Prediction

In this section, we explain how the information acquired regarding the epistemic uncertainty of the model can be directly exploited for the next token prediction task. We achieve this by adding Gaussian noise to the last layer of the model. This procedure can be summarized as follows: to generate token answers using the Laplace Approximation (LA), we begin by initializing an empirical Hessian matrix to capture the curvature of the loss function around the model's parameters. We then compute the standard deviation from this Hessian, representing the uncertainty in the parameters. Using a Monte-Carlo sampling approach, we iterate a predefined number of times, generating random samples from a normal distribution based on this standard deviation. For each sample, we temporarily update the model's weights and generate token answers, restoring the weights after each iteration. This process yields token answers that reflect the probabilistic uncertainty in the model's predictions. Thus, what we described as a straightforward subroutine to enhance the model's awareness of its responses to specific questions is, in reality, a prompt inserted directly within the model.

4 Experiments

This section presents experimental evidence demonstrating that a posterior sampling policy achieves comparable results to softmax (logits) sampling policies. Specifically, we refer to the former as *Bayesian introspection*, while the latter is denoted as *softmax introspection* (implemented with Temperature sampling with $T = 1$). Our experiments are conducted on the *TruthfulQA* benchmark [Lin et al., 2022], available from *HuggingFace Datasets*. Our reference model is *Phi-2*, a small language model developed by Microsoft Research with 2.7 billion parameters. It is used inside the framework proposed by [Gao et al., 2023][1]. We focus on the MC1 sub-category of TruthfulQA, which involves multiple-choice tasks. Thus, our approach can be viewed as an application of posterior sampling to (only) next-token prediction task. Future research may extend this technique to open text-generation tasks which require more careful aggregation policies.

4.1 Sampling Procedures

In this subsection, we succinctly describe the two different procedures adopted:

- *Softmax introspection*: This method samples the next token directly ($T = 1$) from the logits of the model, generating answers directly from the model's decoder.
- *Bayesian introspection*: In contrast, this approach utilizes sampling with Gaussian noise to generate the next token through posterior-like sampling.

4.2 Task

Here, we provide a brief overview of how the MC1 (Multiple Choice 1) task operates within the TruthfulQA benchmark and the specific areas of language model capabilities it aims to stress. The MC1 task is designed to present language models with challenging scenarios, particularly emphasizing their ability to discern and generate truthful responses in the presence of deceptive or imitative falsehoods. Each question in the MC1 task offers multiple-choice options, typically consisting of 4-5 strings representing the suggested answers. Each question has only one correct answer.

4.3 Results

Here we report the experimental results of Bayesian introspection across 3 different settings of prior precision. On the one hand, Softmax introspection achieves $28.03 \pm 1.57\%$ of correct answers on the same task, on the other hand, the results of Bayesian introspection can be seen in table 1. In both cases, these procedures outperform the score of the same model without any kind of introspection ($20.69 \pm 1.42\%$).

[1]For our GitHub repository, see lm-eval

Prior Precision	**Accuracy (%)**
10^2	$26.93 \pm 1.55\%$
10^3	$27.05 \pm 1.55\%$
10^4	$27.17 \pm 1.55\%$

Table 1: MC1 task accuracy - Bayesian introspection

5 Prior Work

As previously mentioned, the concept of sampling holds a significant position in the research field of modern Large Language Models (LLMs). In this section, we aim to provide precise definitions of two specialized techniques, informally referred to in earlier sections as *softmax sampling*. Throughout this paper, we adopt this term whenever we refer to a sampling algorithm based on the probability distribution of the last layer, namely the softmax output layer in the transformer architecture proposed [Vaswani et al., 2023].

One such sampling procedure is recognized as *temperature sampling* ([Ackley et al., 1985], [Ficler and Goldberg, 2017]). Temperature sampling involves adjusting the temperature parameter of the softmax function during sampling, which controls the smoothness of the probability distribution. Higher temperatures lead to a more uniform distribution, while lower temperatures emphasize high-probability tokens.

Similarly, *nucleus sampling* ([Holtzman et al., 2020]) and *top K-sampling* ([Fan et al., 2018]) are well-established techniques in the domain of language modeling. Nucleus sampling, also known as top-p sampling, involves sampling from the smallest subset of the vocabulary whose cumulative probability exceeds a certain threshold, denoted by a parameter p. Top K-sampling, on the other hand, involves sampling from the top K most probable tokens according to the logits of the model's output layer.

Despite their differences, these techniques share a common dependency on the logits of the decoder, offering various strategies to sample from the output probability distribution of LLMs.

On the other hand, our work focuses on the concept of *posterior sampling* in the sense of *Laplace approximation.* Unlike techniques such as temperature sampling, nucleus sampling, and top K-sampling, which are primarily based on manipulating the logits layer, posterior sampling relies only on the (approximate) parameter distribution of the model.

6 Conclusion

Integrating Bayesian thinking inside large neural networks has been a significant area of research in recent years. The challenge is that with the increasing number of parameters, architectures such as Bayesian Neural Networks (BNNs) have proven to be unscalable. Laplace Approximation provides an interesting tool to incorporate these principles in a computationally feasible way inside large language models (LLMs).

The main limitation of this approach is that what we called posterior sampling only constitutes half of the solution: to adequately address epistemic uncertainty, a good reinforcement learning (RL) algorithm should learn from its experience (via external rewards) and *adjust* its behavior (by updating its reference distribution accordingly). This highlights the potential for future research to extend this technique to RL in LLMs. Another practical limitation is the lack of access to the actual datasets on which the models have been trained. Computing the Hessian on these datasets would provide a statistically more principled way of obtaining reliable variance information.

A final remark: from the perspective of real 'truthfulness' (which differs from accuracy), our approach is safer than other state-of-the-art techniques. This safety arises because the epistemic uncertainty captured by the Laplace Approximation is inherent to the model, thereby limiting its ability to provide completely grounded answers. Therefore, while our results in terms of accuracy metrics are comparable to those of current state-of-the-art techniques, the additional epistemic uncertainty information is crucial for further advancements in the field of safe policies for next token prediction.

References

David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cogn. Sci.*, 9:147–169, 1985. URL `https://api.semanticscholar.org/CorpusID:12174018`.

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms, 2023.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux – effortless bayesian deep learning, 2022.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation, 2018.

Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation, 2017.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL `https://zenodo.org/records/10256836`.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2020.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems*, 32, 2019.

Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the empirical fisher approximation for natural gradient descent, 2020.

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464, 2023.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods, 2022.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Mengqi Miao, Fandong Meng, Yijin Liu, Xiao-Hua Zhou, and Jie Zhou. Prevent the language model from being overconfident in neural machine translation. *arXiv preprint arXiv:2105.11098*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.

Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

A Fisher Matrix and Laplace Approximation

Expected Fisher Matrix The Laplace approximation discussed previously requires the Hessian of the loss function. To simplify this, we decompose the Hessian into the sum of the Hessian of the likelihood and the Hessian of the prior: $H = H_l + H_p$. In practice, the true Hessian of the likelihood is replaced with the expected Fisher matrix[2].

$$H_l \approx |\mathcal{D}|\mathbb{E}_x\mathbb{E}_{y\sim P_\theta(y|x)}\left[-\nabla^2 \log P_\theta(y|x)\right] \tag{2}$$

Although the Hessian on the left side depends on the true regression targets, while the right side does not, this approximation is accurate when regression residuals are small [Kunstner et al., 2019]. To make the formula in (2) practical, we use a Monte-Carlo estimate to replace the expectation with respect to the data points.

$$|\mathcal{D}|\mathbb{E}_x\mathbb{E}_{y\sim P_\theta(y|x)}\left[-\nabla^2 \log P_\theta(y|x)\right] \approx \sum_{x\in\mathcal{D}} \mathbb{E}_{y\sim P_\theta(y|x)}\left[-\nabla^2 \log P_\theta(y|x)\right] \tag{3}$$

Further, by using the identity from [Kunstner et al., 2019],

$$\sum_{x\in\mathcal{D}} \mathbb{E}_{y\sim P_\theta(y|x)}\left[-\nabla^2 \log P_\theta(y|x)\right] = \sum_{x\in\mathcal{D}} \mathbb{E}_{y\sim P_\theta(y|x)}\left[\nabla \log P_\theta(y|x)\nabla \log P_\theta(y|x)^\top\right] \tag{4}$$

we derive a formula that avoids the need to compute second-order derivatives. This highlights a key advantage of the expected Fisher matrix: it is positive semi-definite by construction.

Diagonal Approximation While equation (4) is theoretically evaluable, the Hessian matrix's size is impractically large for even small-scale language models. Thus, we approximate $\hat{H}_l$ by computing only the diagonal entries of equation (4).

$$\text{diag}(\hat{H}_l) = \sum_{x\in\mathcal{D}} \mathbb{E}_{y\sim P_\theta(y|x)}\left[(\nabla \log P_\theta(y|x))^2\right] \tag{5}$$

This diagonal approximation corresponds to considering the Taylor expansion behind the Laplace approximation for each coordinate separately, resulting in a posterior that is a normal distribution with diagonal covariance. However, this distribution is not necessarily the optimal KL-projection of the Gaussian derived from equation (4) onto the space of diagonal Gaussians[3].

B Approximations of Expected Fisher Matrix

The computation of the empirical Hessian was performed on a subset of a public dataset rather than employing a more statistically principled approach due to practical constraints. In the context of Laplace approximation (LA), the most efficient method to compute an approximate Hessian is to utilize the following diagonal approximation, as outlined in [Kunstner et al., 2020], i.e by computing the right term of:

$$diag(H_l) = \sum_{x\in\mathcal{D}} \mathbb{E}_{y\sim p_\theta(y|x)}(\nabla \log p_\theta(y|x))^2$$

[2]The term "expected Fisher" is used to differentiate it from the empirical Fisher matrix [Kunstner et al., 2019].

[3]Such a projection could theoretically be obtained by inverting the formula in equation 4 and then taking the diagonal, which is intractable.

where $\mathcal{D}$ is the dataset on which we compute the (approximate) hessian, H_l is the expected Fisher matrix and p_θ is the parametric likelihood of the model.
However, practical limitations emerge within the context of (very) large transformer architectures, exemplified by those utilized in the most recent Large Language Models (LLMs). This arises primarily from the challenge posed by the large number of output classes onto which the last layer of the transformer map the resulting output probabilities. In this section, we provide a simple explanation as to why the expected Fisher matrix proves unfeasible in practical applications.

Consider a neural network model $f(\theta) = f_\theta(x)$, where θ denotes the model parameters. The softmax function associated to this model is given by:

$$\text{softmax}(f(\theta))_i = \frac{e^{f_i(\theta)}}{\sum_{j=1}^{K} e^{f_j(\theta)}},$$

The probability of observing the true class y^* given the model output can be modeled as:

$$p(y^*|f(\theta)) = \prod_{k=1}^{K} (\text{softmax}(f(\theta))_k)^{y_k^*}.$$

Remark: the density function is indeed a multinomial distribution, in particular

$$p(y^*|f(\theta)) \sim Multinomial(n = 1, k = K, p = (p_1, p_2, ..., p_k))$$

where $p_i = softmax(f(\theta))_i$.

The log likelihood of y^* given $f(\theta)$ is:

$$\log(p(y^*|f(\theta))) = \sum_{k=1}^{K} y_k^* \log(\text{softmax}(f(\theta))_k).$$

The derivative of the log likelihood with respect to θ involves the chain rule:

$$\frac{\partial \log(p(y^*|f(\theta)))}{\partial\theta} = \sum_{i=1}^{K} (y_i^* - \text{softmax}(f(\theta))_i) \frac{\partial f_i(\theta)}{\partial\theta}.$$

To compute the expected value of the squared gradient of this log likelihood, considering y^* distributed according to the softmax model $p(y^*|f(\theta))$, we evaluate:

$$\mathbb{E}\left[\left(\frac{\partial \log(p(y^*|f(\theta)))}{\partial\theta}\right)^2\right].$$

First, observe that:

$$\left(\sum_{i=1}^{K} (y_i^* - \text{softmax}(f(\theta))_i) \frac{\partial f_i(\theta)}{\partial\theta}\right)^2 =$$

$$\sum_{k=1}^{K} (y_k^* - \text{softmax}(f(\theta))_k)^2 \left(\frac{\partial f_k(\theta)}{\partial\theta}\right)^2 +$$

$$2 \cdot \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} (y_i^* - \text{softmax}(f(\theta))_k)(y_j^* - \text{softmax}(f(\theta))_j) \left(\frac{\partial f_i(\theta)}{\partial\theta}\right) \left(\frac{\partial f_j(\theta)}{\partial\theta}\right)$$

By observing that each component y_k^* of the one-hot encoded vector y^*, considered a random variable Y on the probability space $([K], \mathcal{P}([K]), \mathbb{P}^Y)$, where $[K] =$

$\{k_1, k_2, ..., k_K\}$, follows a Bernoulli distribution with mean $p_k = softmax(f(\theta))_k$, the expectation becomes:

$$\mathbb{E}_{Y\sim p_\theta(y|x)}\left[\left(\sum_{i=1}^{K}(y_i^* - \text{softmax}(f(\theta))_i)\,\frac{\partial f_i(\theta)}{\partial\theta}\right)^2\right] =$$

$$\sum_{k=1}^{K}\text{softmax}(f(\theta))_k \cdot (1 - \text{softmax}(f(\theta))_k)\left(\frac{\partial f_k(\theta)}{\partial\theta}\right)^2 -$$

$$2\cdot\sum_{i=1}^{K-1}\sum_{j=i+1}^{K}\text{softmax}(f(\theta))_i \cdot \text{softmax}(f(\theta))_j\left(\frac{\partial f_i(\theta)}{\partial\theta}\right)\left(\frac{\partial f_j(\theta)}{\partial\theta}\right)$$

Then, the computation of the expected Fisher matrix would take $\Theta(K^2)$, where K is the number of output classes of the model. Hence, it is infeasible for K sufficiently large.