NATIONAL RESEARCH UNIVERSITY HIGHER SCHOOL OF ECONOMICS

As a manuscript

Kirill Struminsky

LEARNING GAURANTEES AND EFFICIENT INFERENCE FOR STRUCTURED PREDICTION

PhD Dissertation
for the purpose of obtaining academic degree
Doctor of Philosophy in Computer Science

Academic Supervisor:
Candidate of Sciences,
Research Professor Vetrov Dmitry Petrovich

Moscow — 2023

Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

На правах рукописи

Струминский Кирилл Алексеевич

ГАРАНТИИ ОБУЧЕНИЯ И ЭФФЕКТИВНЫЙ ВЫВОД В ЗАДАЧАХ СТРУКТУРНОГО ПРЕДСКАЗАНИЯ

ДИССЕРТАЦИЯ
на соискание учёной степени
кандидата компьютерных наук

Научный руководитель:
кандидат физ.-мат. наук,
Ветров Дмитрий Петрович

Москва — 2023

Contents

1 Introduction

Machine learning attempts to recover and describe empirical relationships in data. Often the interest is in quantifying or attributing observed data to a predetermined set of categories. For example, how does the price of an apartment depend on its location and parameters? Will the user want to read this email? These questions can be answered based on historical data containing details of past transactions or the history of user interaction with previously received emails. Attribution can also be of interest when the attributes are not known in advance: is it possible, for example, to distinguish several distinctive categories in the data?

At the same time, in applications there are problems in which the desired dependencies fall outside the scope of the examples described above. If, for example, we are talking about a machine translation task, then each text in the source language must be matched with a text in the target language. In this case, it would be incorrect to represent the predicted translation as a number or an element of the set of all possible translations. On the contrary, it would be convenient to represent the text as a sequence of words, where the translation algorithm must predict each word, focusing both on the original sentence and on neighboring words of the translation.

Variables in the data, represented as a set of mutually dependent values, are usually called structured. The area of machine learning devoted to the prediction of structured variables is called structured prediction. A characteristic feature of structured variables is the combinatorial growth of the number of possible values (outcomes) depending on the parameters of the problem. Ignoring the nuances of the problem, in the machine translation example, with a dictionary size of w and a known translation length l, the algorithm must choose among w^l possible translations. This feature raises questions about learning guarantees and efficient inference. Namely, how many examples are enough to reliably restore the required dependence? How to quickly select an element from a possible set of outcomes? This work is devoted to the study of these issues.

1.1 Relevance

As the area of machine learning application keeps spreading[56], the variety of tasks and problem setups is also growing, making structured prediction more in demand. In particular, the deep learning developments have made it possible to bring algorithms for natural language processing and computer vision to a qualitatively new level. In supervised learning problems where the target variables are structured variables, learning is often reduced to minimizing the cross-entropy loss function. Such a loss function, in turns, requires defining the distribution of a target structured variable. For example, in natural language processing tasks, distribution over text outputs is introduced by factorizing the distribution into word-level distributions according to the chain rule (for more details, see [23, Chapter 10]). Another solution, common, for example, in the problem of semantic segmentation, is to assume that all elements of a structured variable are independent given the input image (as, for example, done in [54]). Recent studies are mostly devoted to the design of neural network architectures for parameterizing distributions in the described approach, as well as scaling the described approach [8, 30, 72]. Structured outputs prompted such key developments as recurrent [28, 62] and convolutional neural networks [22, 43], as well as transformers [67] for sequence processing, UNet architecture for image processing [54].

The disadvantage of the above approach to structured prediction and deep learning in general is the limited interpretability of the recovered dependencies. In the meantime, certain governmental regulators introduce the "right to explanation"[68], according to which a person can demand an explanation of how the machine learning system made a decision regarding him. Thus, the problem of interpreting machine learning algorithms becomes especially acute with the development of deep learning systems. As a result, a designated area of research has emerged, attempting to interpret specific architectures [69, 31, 52], as well as to develop interpretation recipes for arbitrary machine learning algorithms [37, 51, 11]. At the same time, the idea of using latent structured variables to increase the interpretability of machine learning algorithms has gained popularity [32, 39]. Next, we describe the idea in more detail. Deep neural networks comprise a sequence of elementary computing blocks, however the combined output of these blocks is difficult to interpret. On the other hand, network evaluation may be more transparent if some of these intermediate construction blocks have interpretable (structured) outputs, and the network architecture itself takes into account the problem specifics. For example, in a sentiment analysis task one can design a model that chooses a small subset of words, based on which the model will make a prediction. In practice, the words chosen by such a model help to interpret the output. Besides that,

neural networks with latent structured variables can be seen as an evolution of latent variable models such as hidden Markov chains [12] or probabilistic context-free grammars [55] for modeling languages by adding more expressive neural network models.

However, in the case of discrete latent variables, the standard training approaches based on back-propagation is not applicable due to the non-differentiability of the block that returns the latent variable. The solution to this problem usually comes down to heuristic gradient substitutes [4] or stochastic relaxation [29, 38, 5, 45]. One of the chapters of this work is devoted to the problem of learning with hidden permutations. Another problem related to latent structured variables, which does not lose its relevance to this day, is the design of architectures with latent variables and the choice of objective functions. As previous work indicates [33, 16], end-to-end learning in such models often leads to predictive models that ignore hidden variables, learning the dependence only on the basis of standard neural network components. The standard solution in this case is learning with partial labeling of latent variables: for a subset of training samples, an additional loss function is introduced to encourage the desired prediction. An alternative would be to choose an architecture that does not allow for sufficient prediction accuracy without using the hidden variable [11].

Along with the development of practical approaches and algorithms for working with structural variables, it is important to obtain guarantees on the quality of their work. In the context of structured prediction, the combinatorial growth in the number of possible predictions and the unequal contribution of erroneous predictions (not all inaccurate predictions are equally bad) are the two factors that distinguish structured prediction from the well-studied classification setup [44]. Generalization in the context of structured prediction is discussed in [17, 36]. In practice, target metric often does not coincide with the functional being optimized during training (a surrogate loss function); a number of results on relationship between target and surrogate losses have been obtained for structural prediction problems. In the paper [14], the authors showed the consistency of a class of quadratic surrogate loss functions, and the paper [44] obtained an estimate for the discrepancy between the accuracy of the prediction according to the target metric and the surrogate loss function. Later, [42] generalized these results to smooth convex surrogate loss functions. The above works assume that the surrogate loss function is consistent, although inconsistent surrogates are also often used in practice: for example, the multi-class support vector machine in the Crammer-Singer form [19], as well as its generalizations to structured variables [63, 65]. As part of the study of inconsistent loss functions, this dissertation generalized the results [44] by obtaining estimates for quadratic surrogate loss functions without the additional requirement of consistency.

1.2 Work Goals

As noted above, structured variables often arise in various machine learning applications. Prospective problem setups may include structured target variables in the case of supervised learning, as well as structured latent variables in both supervised and unsupervised setups. In addition to prediction quality metrics, inference speed becomes a critical performance aspect as we shift to structured variables with a combinatorial number of possible outcomes. The goal of this work was to develop structured prediction methods that meet the requirements arising in applications: to develop structured prediction methods for observed and latent structured variables, while emphasising algorithms with feasible inference time and the availability of learning guarantees for the proposed methods.

Within the framework of the goals described above, the following **tasks** were set:

1. development prediction methods for such structural variables as permutations and subsets of a given size,
2. study of consistency and derivation of learning guarantees for supervised learning tasks with a structured target variable,
3. development and empirical analysis of models with latent structural variables,
4. development of efficient inference methods for structured latent variables
5. the use of latent structured variables for data interpretation, as well as the construction of interpretable machine learning methods.

Contributions. When solving the tasks above, we obtained the following results.

1. We developed and evaluated a gradient-based method to optimize over a set of permutations or subsets.

2. In supervised structured prediction setup, we carried out analysis of quadratic surrogate loss functions and quantified surrogate consistency in a novel setting.

3. We proposed and studied several approaches to recovering latent structured variables based on maximum evidence principle and quadratic surrogate loss functions.

4. We proposed a number of efficient inference procedures for such latent structured variables as permutations and fixed-size subsets.

5. We developed methods for interpreting data based on latent structured variables.

1.3 Practical Applications

The developed approach to permutation optimization is applicable for restoring the structure of the relationship between variables in data, which, in particular, is in demand when interpreting machine learning models. The prior distribution for convolutional neural network parameters offers a method for rapidly adapting model parameters to a new adjacent data domain. The method for estimating the parameters of a multi-user communication channel finds application in modern cellular networks. A probabilistic model for preprocessing geophysical exploration data provides a convenient way to detect anomalies and recover gaps in historical data.

1.4 Methodology

Our theoretical analysis of structured prediction is based on sections of probability theory, statistical learning theory, and optimization. In a general structured prediction setup, we obtained a result applicable to a number of structured prediction problems. Other consideration are based on probabilistic machine learning formalism, as well as the Bayesian approach to machine learning. The proposed methods are based on the basic sections of probability theory and stochastic optimization. Besides a few rigorous proofs, this work mostly relies on the empirical evaluation methods. We implemented the proposed algorithms in Python, assessed their performance and compared with analogues on synthetic and real data sets.

1.5 Publications and Probation of the Work

First-tier publications:

1. **Struminsky K.**, Lacoste-Julien S., Osokin A. Quantifying Learning Guarantees for Convex but Inconsistent Surrogates //Advances in Neural Information Processing Systems. – 2018. – С. 669-677. *Contribution of the thesis author:* A general lower bound on the calibration function in structured prediction setup; calculation of the lower bound coefficients for hierarchical classification; calculation of the lower bound coefficients for ranking.

2. Gadetsky, A., **Struminsky, K.**, Robinson, C., Quadrianto, N., & Vetrov, D. P. (2020). Low-Variance Black-Box Gradient Estimates for the Plackett-Luce Distribution. In AAAI (pp. 10126-10135). *Contribution of the thesis author:* An approach to optimization over permutations and acyclic graphs based on variational optimization for Plackett-Luce distributions; generalization of the RELAX gradient estimator to the case of the Plackett-Luce distribution.

3. Atanov, A., Ashukha, A., **Struminsky, K.**, Vetrov, D., & Welling, M. (2018, September). The Deep Weight Prior. In International Conference on Learning Representations. *Contribution of the thesis author:* Adaptation of the variational auto-encoder to the problem of estimating the prior distribution on the parameters of the Bayesian neural network.

Standard-tier publications:

1. **Struminsky K.** et al. A new approach for sparse Bayesian channel estimation in SCMA uplink systems //2016 8th International Conference on Wireless Communications & Signal Processing (WCSP). – IEEE, 2016. – С. 1-5. *Contribution of the thesis author:* Probabilistic model for estimating the parameters of a multi-user communication channel; improved scheme for approximate inference of parameters of a multi-user communication channel and estimation of the channel configuration.

2. **Struminskiy K.** et al. Well Log Data Standardization, Imputation and Anomaly Detection Using Hidden Markov Models //Petroleum Geostatistics 2019. – European Association of Geoscientists & Engineers, 2019. – Т. 2019. – №. 1. – С. 1-5. *Contribution of the thesis author:* A probabilistic model for the preprocessing of geological and physical exploration data.

In all papers, with the exception of "The Deep Weight Prior" [1], the applicant is the main author.
Conference presentations and seminar talks:

1. Bayesian Deep Learning Workshop, NeurIPS 2019, Vancouver, Canada, 13 December, 2019.

   Topic: Low-variance Gradient Estimates for the Plackett-Luce Distribution (spotlight presentation, poster).

2. 8th International Conference on Wireless Communications and Signal Processing, Yangzhou, Chine, 13-15 Ocboter, 2016.

   Topic: A new approach for sparse Bayesian channel estimation in SCMA uplink systems (oral presentation).

3. Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20), New York, USA, 7-12 February, 2020.

   Topic: Low-Variance Black-Box Gradient Estimates for the Plackett-Luce Distribution (oral presentation, poster).

4. EAGE Conference on Petroleum Geostatistics, Florence, Italy, 2-6 September, 2019.

   Topic: Well Log Data Standardization, Imputation and Anomaly Detection Using Hidden Markov Models (oral presentation).

5. Thirty-second Annual Conference on Neural Information Processing Systems (NeurIPS 2018), Montral, Canada, 2-8 December, 2018.

   Topic: Quantifying Learning Guarantees for Convex but Inconsistent Surrogates (poster).

6. Thirty-fifth Annual Conference on Neural Information Processing Systems (NeurIPS 2021), online, 6-14 December, 2021.

   Topic: Leveraging Recursive Gumbel-Max Trick for Approximate Inference in Combinatorial Spaces (poster).

7. Seventh International Conference on Learning Representations (ICLR 2019), New Orlean, USA, 6-9 May, 2019.

   Topic: The Deep Weight Prior (poster).

8. Bayes Group Research Seminar, Moscow, Russia, 26 October, 2018.

   Topic: Quantifying Learning Guarantees for Convex but Inconsistent Surrogates (oral presentation).

9. Sberbank Data Science Journey, Moscow, Russia, 10 November, 2018.

   Topic: Quantifying Learning Guarantees for Convex but Inconsistent Surrogates (oral presentation, poster).

10. Machines Can See: Computer Vision and Deep Learning Summit, Moscow, Russia, 25 June, 2019.

    Topic: The Deep Weight Prior (poster).

11. International Conference on Analysis of Images, Social Networks and Texts, AIST 2019, Kazan, Russia, 17-19 Jule, 2019.

    Topic: A Simple Method to Evaluate Support Size and Non-uniformity of a Decoder-Based Generative Model (oral presentation).

12. Advances in Approximate Bayesian Inference, NIPS 2016 Workshop, Barcelona, Spain, 2016.

    Topic: Robust Variational Inference (poster).

2 Preliminaries

2.1 Structured Variables in Machine Learning

We start by introducing the concept of a structured variable. In machine learning, a structured variable is an umbrella term for random variables, united by the following characteristic properties. Firstly, a structured variable has a large number of possible values: the support of a random variable is typically a finite set that cannot be quickly enumerated on a computer. Secondly, these variables are presented as a set of mutually dependent random variables. The second property can act as a definition of a structured variable. For clarity, we turn to specific examples below.

In applications, structured variables can act as a target variable in supervised problems (structured prediction), and can also act as an auxiliary latent variable in models with latent variables.

One of the standard examples of a structured prediction problem is segmentation in computer vision [34]. In this case, the structured variable is the segmentation mask of an image. Segmentation mask components are mutually dependent, since close points of an image with high probability correspond to the same class. Other examples of structured prediction problems include ranking [10, 49], extreme classification [13]. Many natural language processing tasks are also structured prediction tasks. A model that produces a text output, whether it is a summation, a translation, or an answer to a question, must predict a sequence of interdependent random variables. In deep learning, such models are defined by the seq2seq architecture [62], and for prediction they use approximate search algorithms among all possible options [50].

Before the spread of deep learning methods, structured variables were also in demand in natural language processing tasks, often playing the role of auxiliary latent variables there [58]. For example, early machine translation algorithms could rely on the input sentence parse tree to better convey the sentence meaning. In this example, the structured variable is the sentence parse tree, and a separate auxiliary model trained on different data could be used to build the tree.

However, these days machine learning solutions rarely rely on pipelines built with auxiliary models and tasks. Instead, deep neural networks allow end-to-end learning, pre-training on unlabeled data [41, 20], and knowledge transfer to small datasets [74]. As a result, end-to-end learning in models with latent structured variables became a relevant research topic. Such models allow to take the best of both worlds: on the one hand, the flexibility of neural networks, on the other hand, reliance on prior knowledge through structured variables for better interpretability and more efficient use of data.

Popular models with latent structured variables include hidden Markov chains [48] with sequence markup as a structured variable, probabilistic context-free grammars [15] with a parse tree as a structured variable, and a temporal sequence classification model [25] with latent sequence segmentation mask. These models are based on limiting assumptions on the model variables that are necessary for efficient learning an inference. More recent approaches circumvent the limiting assumptions by relying on stochastic gradient descent for end-to-end learning and fast amortized inference [45]. Some of the examples include models with hidden parse trees [16], implicit feature subset selection [11], and hidden text generation order [27].

In the next section, we introduce a general supervised structured prediction setup.

2.2 Structured Prediction Basics

Let us first consider the standard structured prediction setup. Namely, consider a supervised learning problem with inputs $x \in \mathcal{X}$ from an arbitrary set $\mathcal{X}$, and the goal is to predict a structured variable $y \in \mathcal{Y}$, which takes values in a finite set $\mathcal{Y}$. The data is distributed according to law $\mathcal{D}$, and y is a realization of a random vector Y with support $\mathcal{Y} \subset \mathbb{R}^m$. In general, the label of a training sample lie in a $\hat{\mathcal{Y}}$, which can differ from $\mathcal{Y}$.

To define a prediction algorithm, we define a function $f : \mathcal{X} \to \mathbb{R}^{|\mathcal{Y}|}$ that assigns a score to each possible structure $y \in \mathcal{Y}$ and then chooses the optimal structure as a prediction

$$\text{pred}(f(x)) := \arg\max_{y \in Y} f_y(x). \tag{1}$$

The difference between structured prediction setup and supervised learning setup is that the set of possible outcomes $\mathcal{Y}$ is large due to the combinatorial growth of possible outcomes. For example, in ranking, the outcome can be a permutation of elements, and when segmenting, a sequence of class labels. Therefore, the model should offer a quick way to solve the problem 1 at inference stage. In addition, we need to store function f in memory. Typically, one resorts to a low-rank parameterization of the function $f(x) = Fg(x)$, where $F : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{Y}|}$ is a fixed linear operator and $g(x) : \mathcal{X} \to \mathbb{R}^d$ is a function with we construct during training. Such a parameterization allows to reduce the memory footprint as we only have to store a function with $d \ll \|\mathcal{Y}\|$ outputs and allows to design efficient algorithms for inference task 1 that rely on the choice of matrix F. At the same time, the parameterization limits the set of possible predictions, since the score vector $f(x)$ lies within the linear span of the columns of the matrix $\mathcal{F} = \text{span}\, F$. Below we refer to $\mathcal{F}$ as the set of feasible scores.

Given a loss function $L(\cdot, \cdot) : \mathcal{Y} \times \hat{\mathcal{Y}} \to \mathbb{R}$ quantifying prediction quality, the goal is to find f that is optimal in terms of risk (the expected value of the loss):

$$\mathcal{R}_L(f) := \mathbb{E}_{X,Y} L(\text{pred}(f(X)), Y). \tag{2}$$

Direct optimization of risk is often unfeasible (in particular, a finite sample approximation of $R_L(f)$ is not differentiable with respect to f outputs). For optimization, we introduce an auxiliary (surrogate) loss function $\Phi : \mathbb{R}^k \times \mathcal{Y} \to \mathbb{R}$ and define the surrogate risk

$$\mathcal{R}_\Phi(f) := \mathbb{E}_{X,Y} \Phi(f(x), y). \tag{3}$$

We emphasize that the function 3 takes the value of $f(x)$ as an argument and makes gradient optimization feasible, whereas the objective function 2 takes the predictions from a discrete set. Popular surrogate loss functions include quadratic functions [14, 7], likelihood-based functions [34] and the surrogates arising in variations of SVM [53, 65].

When replacing the objective function with a surrogate one, the question inevitably arises of the relationship between the optimum of the surrogate loss function and the solution of the original problem. To answer this question, the concept of consistency of a surrogate loss function was introduced [2]. The concept is closely related to the concept of Fisher consistency [18, p.287]. Intuitively, a surrogate loss function is consistent if the optimal f w.r.t. the surrogate risk is also optimal with respect to the original risk.

Let us define consistency in terms of the calibration function that connects the surrogate and target loss functions. For a score $f \in \mathcal{F} \subseteq \mathbb{R}^k$ and a distribution $q \in \Delta_k$ on a set of possible outcomes $Y \sim q$, we introduce the conditional risk $l(f, q) := \mathbb{E}_Y L(\text{pred}(f), Y)$ and the conditional surrogate risk $\phi(f, q) := \mathbb{E}_Y \Phi(f, Y)$. The excess (surrogate) risk is the deviation of $\delta l(f, q)$ ($\delta\phi(f, q)$) from the optimal risk

$$\delta l(f, q) := l(f, q) - \inf_{\hat{f} \in \mathcal{F}} l(\hat{f}, q) \tag{4}$$

$$\delta\phi(f, q) := \phi(f, q) - \inf_{\hat{f} \in \mathcal{F}} \phi(\hat{f}, q). \tag{5}$$

Using these auxiliary functions, we define the calibration function.

Definition 1. For a loss function L, a surrogate loss function Φ, and a set of feasible scores $\mathcal{F}$, the calibration function $H_{\Phi,L,\mathcal{F}}(\varepsilon)$ at the argument $\varepsilon \geq 0$ is equal to the surrogate risk infimum given the target risk is not less than ε:

$$H_{\Phi,L,\mathcal{F}}(\varepsilon) := \inf_{f \in \mathcal{F}, q \in \Delta_k} \delta\phi(f, q) \tag{6}$$

$$\text{s.t.} \delta l(f, q) \geq \varepsilon \tag{7}$$

Intuitively, the calibration function estimates how small the error of the surrogate loss function can be for a fixed loss function value. When loss is high due to prediction error, a consistent surrogate loss function should also return a high value. The following theorem connects the surrogate risk and the target risk using the calibration function.

Theorem 1 (Associating $\mathcal{R}_L$ with $\mathcal{R}_\Phi$ through the calibration function)**.** *Let $H_{\Phi,L,\mathcal{F}}$ be the calibration function for the loss function L and the surrogate loss function Φ, and let $\mathcal{F}$ be the set of feasible scores. Let $\Phi, \hat{L}, \mathcal{F}$ be a convex non-decreasing lower bound for the calibration function $H_{\Phi,L,\mathcal{F}}$. Assume additionally that Φ is a continuous function bounded from below. Then for any $\varepsilon > 0$ such that $\hat{H}_{\Phi,L,\mathcal{F}}$ is finite and any score $f \in \mathcal{F}$ holds*

$$\mathcal{R}_\Phi(f) < \inf_{\hat{f} \in \mathcal{F}} R_\phi(\hat{f}) + \hat{H}_{\Phi,L,\mathcal{F}}(\varepsilon) \Rightarrow \mathcal{R}_L(f) < \inf_{\hat{f} \in \mathcal{F}} \mathcal{R}_L(\hat{f}) + \varepsilon. \tag{8}$$

Next, we define η-consistency of a surrogate loss function, which is inspired by the above theorem.

Definition 2 (η-consistency of a surrogate loss function)**.** A surrogate loss function Φ is consistent up to the level $\eta \geq 0$ (η-consistent) for the objective function L and the set of feasible scores $\mathcal{F}$ if and only if the calibration function satisfies $H_{\Phi,L,\mathcal{F}}(\varepsilon) > 0$ for any $\varepsilon > \eta$ and there exists $\hat{\varepsilon} > \eta$ such that that $H_{\Phi,L,\mathcal{F}}(\hat{\varepsilon})$ is finite.

For $\eta = 0$, the above definition coincides with the notion of consistency common in the machine learning literature [35, 47]. Thus, one can validate consistency of a surrogate loss by showing that the calibration function is positive in the punctured neighborhood of $\varepsilon = 0$. However, in practice consistency may not be sufficient to build realistic learning guarantees. As Osoking et al. [44] showed, for various problems in structured prediction the theorem 1 delivers non-trivial guarantees only for practically unattainable surrogate risk values. It turns out that the scale of the calibration function plays an important role as well.

The notion of η-consistency is crucial for the results presented in this work. First, it allows to obtain learning guarantees in theorem 1 under a weaker assumption of inconsistent surrogate functions, that is, not falling under the definition of 0-consistency. Second, we construct a tighter calibration function lower bound for inconsistent surrogate losses and obtain a more optimistic learning guarantees.

2.3 Probabilistic Approach to Structured Prediction

The structured prediction setup in Section 2.2 used the language of probability to introduce assumptions about the data and reformulate learning as an optimization task. In addition, the language of probability is a convenient tool for defining non-deterministic prediction models and for modeling various modes of uncertainty such as uncertainty in the choice of model and uncertainty in the prediction of a particular model. *Probabilistic machine learning* is an approach to machine learning that relies on probability theory to formulate and solve machine learning tasks. Next, we describe this approach in more detail, starting from common examples of its application.

When formulating a problem within the framework of the probabilistic approach, the first step is to choose a set of random variables appearing in the problem, as well as their joint distribution. Treating the data as random variables, we describe the desired patterns by choosing the appropriate class of distributions.

So, for example, when building a logistic regression model, the class label $y \in \mathcal{Y} = \{-1, 1\}$ is represented as a random variable Y with a Bernoulli distribution depending on the input object $x \in \mathcal{X} = \mathbb{R}^d$, and probability $\mathbb{P}_Y(y \mid x; \theta) = \frac{1}{1+\exp(y\theta^T x)}$ with parameters $\theta \in \mathbb{R}^d$. Following the assumptions about the data, we assume that tuples of objects x and labels y are jointly independent (i.e., data is i.i.d.). Label distribution allows to model the uncertainty in the prediction of the model, which may be due to lack of data, the inflexibility of the model or the label being non-deterministic. If uncertainty also arises when estimating the model parameters, the probabilistic approach allows us to consider the model parameters as a random variable Θ as well. In the absence of any knowledge about the values of the Θ parameter, its distribution can be assumed to be normal $\Theta \sim \mathcal{N}(\mid 0, \operatorname{diag}\sigma), \sigma \in \mathbb{R}^d$ with a diagonal covariance matrix with parameters $\sigma \in \mathbb{R}^d$. Assuming $\Theta \perp\!\!\!\perp Y$, we get the joint distribution of Θ parameters and Y labels. The interpretation of model parameters as random variables underlies the Bayesian approach, allowing one to estimate the uncertainty in choosing model parameters using the posterior distribution $p_\Theta(\theta \mid (x^i, y^i)_{i=1}^n; \sigma)$ for a data set of n objects.

Note that the above example did not make any assumptions about the distribution of the input x, as they are not necessary when considering the classification problem. Such models are called discriminative. At the same time, the probabilistic approach makes it possible to oppose discriminate models to generative

ones, which also model the distribution of input objects. A classic example of a generative model is the naive Bayes classifier. It is based on the joint distribution $p(x, y \mid \theta) := p(x \mid y; \theta)p(y; \theta)$, and when classifying objects it relies on the conditional distribution $p(y \mid x, \theta)$.

In addition, the probabilistic approach allows you to introduce additional random variables, making it possible to simplify the description of the desired dependencies. For example, Latent Dirichlet Allocation[6] model for texts groups objects $x \in \mathcal{X}$ according to topics: for a text corpus, the model defines a set of $\tau \in \mathbb{N}$ topics, and then represents each individual text $p(x \mid t)$ based on a vector of topics $t \in \Delta^T$ that are reflected in the text. An auxiliary random variable in this case is a set of topics in the text t with a priori distribution $p_T(t)$. Since such auxiliary quantities are not reflected in the data, they are commonly referred to as *latent variables*.

The choice of a joint distribution often leads to the choice of a training method. Among the possible training methods, we distinguish two categories: in the case when the choice of model parameters is of interest, the parameters can be obtained by maximizing the likelihood:

$$\max_{\theta} \log p(\{x^i, y^i\}_{i=1}^n \mid \theta) \tag{9}$$

If there are latent variables in the model, it is natural to consider the marginal likelihood instead. In the literature, this approach is referred to as empirical Bayes or type-II maximum likelihood:

$$\max_{\theta} \log p(\{x^i, y^i\}_{i=1}^n \mid \theta) \tag{10}$$

$$\log p(\{x^i, y^i\}_{i=1}^n \mid \theta) = \log \mathbb{E}_T p(\{x^i, y^i\}_{i=1}^n, T \mid \theta). \tag{11}$$

In the case when one of the hidden quantities is of interest, one can restore their characteristic values based on the posterior distribution $p(T \mid \{x^i, y^i\}_{i=1}^n, \theta)$. In particular, the posterior distribution can be used to solve the problem 10. It is often impossible to calculate the posterior distribution explicitly in practice, and one of the common approaches to its approximation is the variational inference, which reduces the task to the optimization problem

$$\max_{\phi} \mathbb{E}_T \log \frac{p(\{x^i, y^i\}_{i=1}^n, T \mid \theta)}{q(T \mid \ phi)}, \tag{12}$$

where the expectation is taken over the random variable T with the distribution $q(\cdot \mid \phi)$, and the optimization is performed over the distribution parameters ϕ.

The objective functions described above can be interpreted as surrogate loss functions introduced using the probabilistic approach. Since surrogate functions eqs. (9), (10) and (12) do not depend on the loss function $L(\cdot, \cdot)$ in any way, these objectives may be inconsistent. Besides that, in comparison with the classical formulation of structured learning, the probabilistic approach allows to operate with latent structured variables. This, in turn, allows to design and train in an "end-to-end" fashion prediction models that involve auxiliary structured latent variables as intermediate components. For example, when solving a discriminative task for texts, the parse trees of sentences can be incorporated as a hidden auxiliary variable that provides additional information for the final prediction.

There are a number of general methods for solving the problems described above. In particular cases, there exists an analytical solution for problems eqs. (9), (10) and (12). In the general case, when an analytical solution is not available, approximate solution can be found using stochastic optimization methods. Problem 9 can be reduced to stochastic optimization in the case when the parameter θ ranges over a discrete structured set and we are unable to iterate through the whole domain containing θ. In problems eqs. (10) and (12), stochastic optimization allows you to optimize the mathematical expectation in the problem statement without resorting to its exact calculation. The two main approaches to constructing unbiased gradient estimates are the reparameterization trick and the REINFORCE algorithm. For structured variables, the first is rarely applicable, and the second often requires careful tuning for each problem.

Below we describe the two main approaches to estimating stochastic gradients. For the problem of a form

$$\max_{\theta} \mathbb{E}_T f(T), \tag{13}$$

where the random variable T has the distribution $q(\cdot \mid \theta)$, the REINFORCE algorithm constructs an unbiased gradient estimator by using the log-derivative trick

$$\nabla_\theta \mathbb{E}_T f(T) = \mathbb{E}_T f(T) \nabla_\theta \log q(T \mid \theta), \tag{14}$$

which allows us to construct an unbiased gradient estimate based on a sample t of the random variable T:

$$g(t, \theta) = f(t) \nabla_\theta \log q(t \mid \theta). \tag{15}$$

The estimate does not impose restrictions on the form of the function f, but requires an efficient algorithm for generating t and computing $\log q(t \mid \theta)$. The latter imposes additional restrictions on certain classes of discrete structured variables, such as distributions based on exponential families. In practice, the convergence of the algorithm can be hindered by the high variance of the estimate $g(t, \theta)$; as a result, the algorithm requires additional control variates to mitigate the gradient variance.

The reparameterization trick allows us to estimate the gradients in 13 under the assumption that the random variable T can be represented as $T = h(U, \theta)$ for a smooth f and a smooth with respect to the second argument h and some random variable U. As the name suggests, the gradient estimate is obtained by differentiating the expectation in a new parameterization

$$\nabla_\theta \mathbb{E}_T f(T) = \nabla_\theta \mathbb{E}_U f(h(U, \theta)) = \mathbb{E}_U \nabla_\theta f(h(U, \theta)), \tag{16}$$

giving an estimate that depends on the sample u of U as

$$g(u, \theta) = \nabla_\theta f(h(u, \theta)) = \left.\frac{\partial f}{\partial t}\right|_{t=h(u,\theta)} \frac{\partial h}{\partial \theta}. \tag{17}$$

Compared to estimate 15, the reparameterized estimate in practice has a lower variance, but imposes additional restrictions on f and T as it involves derivatives. In particular, the estimate 17 is not directly applicable to discrete variables, allowing estimation of gradients only for their continuous approximations.

3 Main Results

Below we cover the central results of the thesis.

3.1 General Methods

3.1.1 Permutation Prediction Based on Variational Relaxation

Our work [21] focuses on methods for approximate inference in the case when the structured hidden variable T is a random permutation. We consider variational distributions within the Plackett-Luce parametric distribution family.

Definition 3. A Plackett-Luce distribution with parameters $\theta_1, \ldots, \theta_n$ is a distribution on permutations with the probability of outcome $t \in S_n$ equal to

$$\mathbb{P}_T(T = t; \theta) = \prod_{i=1}^{n} \frac{\exp \theta_{t_i}}{\sum_{j=i}^{n} \exp \theta_{t_j}}. \tag{18}$$

Intuitively, the above formula corresponds to choosing n out of n elements without replacement, where the probability of choosing the i-th element is proportional to $\exp \theta_i$. The distribution is also of interest from the point of view of probabilistic relaxation of optimization problems. We replace the minimum over the function arguments with the minimum of the average function value over the distribution family parameters

$$\min_t f(t) \leq \min_\theta \mathbb{E}_T f(T), \tag{19}$$

where T has the Plackett-Luce distribution with parameters θ. This estimate smoothly depends on the distribution parameters. Importantly, it is possible to find a set of parameters that leads to an arbitrarily small gap in the above inequality. Indeed, when we scale the parameters $\theta' = \theta / \tau$ by the temperature τ approaching zero, the distribution tends to degenerate distribution. The distribution mode is the permutation that arranges θ in descending order, since such sorting delivers the maximum of each factor in formula 18. Therefore, if the sorting of the vector θ coincides with the optimal permutation of τ^*, temperature scaling lead to an arbitrarily small gap.

Explicit formula for the outcome probability 18 and generation using sampling without replacement allow using the REINFORCE [71] algorithm for approximate inference in the class of Plackett-Luce distributions. However, the default algorithm converges slowly due to high variance, so as part of our work, we adapted the RELAX [24] algorithm to obtain low variance gradient estimates.

Definition 4. Let a discrete random variable T be a function $T = H(Z)$ of a reparameterizable random variable Z with parameters θ. Then the estimate

$$g_{RELAX}(f) = [f(t) - c_\phi(\tilde{z})] \frac{\partial}{\partial \theta} \log \mathbb{P}_T(T = t; \theta) + \frac{\partial}{\partial \theta} c_\phi(z) - \frac{\partial}{\partial \theta} c_\phi(\tilde{z}) \tag{20}$$

for a realization z of a random variable Z, a discrete variable $t = H(z)$, and an independent realization $\tilde{z}$ of a conditional random variable $Z \mid T = t$ is an unbiased estimate of $\mathbb{E}_T f(T)$.

Initially, a similar estimate was proposed in [66], where $c_\phi(\cdot)$ was considered to be a smooth extension of function f to the domain of Z, containing the domain of T. In the paper [24], the authors proposed using an arbitrary $c_\phi(\cdot)$ (assuming differentiability with respect to the argument z and the parameters ϕ) while adjusting the parameters ϕ as it is optimized to reduce the variance. Both papers considered the case of a categorical distribution, while we generalized the method to the case of the Plackett-Luce distribution.

Our generalization is based on the equivalent definition of the Plackett-Luce distribution [73].

Definition 5. Let $Z_1, \ldots, Z_n$ be independent random variables with the Gumbel distribution with the corresponding parameters $\theta = (\theta_1, \ldots, \theta_n)$. Then the sorting of these random variables T has the Plackett-Luce distribution:

$$\mathbb{P}(z_{t_1} \geq \cdots \geq z_{t_n}; \theta) = \prod_{i=1}^{n} \frac{\exp \theta_{t_i}}{\sum_{j=i}^{n} \exp \theta_{t_j}}. \tag{21}$$

Thus, the random variable T can be represented as a deterministic function of Z, and in order to use the bound 20, it suffices to find a reparameterization for the conditional distribution $Z \mid T = t$. In our work, we propose an algorithm for reparameterization and efficient generation from this distribution:

Theorem 2. *Consider mutually independent realizations of the uniform distribution $v_1, \ldots, v_n \sim U[0, 1]$ and realizations of the Gumbel distribution $z_1, \ldots, z_n$ with parameters $\theta_1, \ldots,$ $theta_n$. Then for the permutation $t = \arg\operatorname{sort}(z_1, \ldots, z_n)$ the vector $\tilde{z} = (\tilde{z}_1, \ldots, \tilde{z}_n)$ defined as*

$$\tilde{z}_{t_i} = \begin{cases} -\log(-\log(v_i)) & i = 1 \\ -\log\left(\frac{\log v_i}{\sum_{j=i}^{n} \exp \theta_{t_j}} + \exp(-\tilde{z}_{t_{i-1}})\right) & i \geq 2 \end{cases} \tag{22}$$

is a realization of the conditional distribution $Z \mid T = t$.

We studied the performance of the proposed method on the problem of finding a causal data structure, considering several problem settings. First, we considered synthetic data generated from the Structured Equation Model, [46]. To generate data, we chose a random directed acyclic graph $G = (E, V)$ with a weighted adjacency matrix $W \in \mathbb{R}^{n \times n}$, and then generated data $X \in \mathbb{R}^{n \times N}$ satisfying the equation

$$X = W^T X + \varepsilon, \tag{23}$$

where ε is homoscedastic Gaussian noise. This equation describes a linear dependence in which each component of X_i depends on the parents of the vertex i in the graph G, as well as on the random noise. The task was to restore the structure of the graph G from the data X.

To reduce the problem to the problem of inferring a permutation, we parameterized the desired adjacency matrix W based on topological sorting: $W = PAP^T$, where $A \in \mathbb{R}^{n \times n}$ was a strictly upper triangular matrix, and P was the permutation matrix for the topological sorting of the graph. For the chosen parameterization, we solved the problem

$$\min_{P \in \mathbb{P}} \min_{A \in \mathbb{A}} \frac{1}{2N} \|X - PAP^T X\}_F^2 + \lambda \| \operatorname{vec}(A)\|_1 = Q(P, A), \tag{24}$$

where $\mathbb{P}$ is the set of permutation matrices and $\mathbb{A}$ is the set of strictly upper triangular matrices.

To optimize over the set of permutations, we switched to the probabilistic relaxation

$$\min_{\theta} \mathbb{E}_T \min_{A \in \mathbb{A}} Q(P(T), A). \tag{25}$$

We compared our method with the previously proposed Gumbel-Sinkhorn [40] and URS [26] algorithms based on P permutation matrix relaxation. The table 1 shows the results of experiments for four families

of graphs with 20 vertices. The algorithms proposed for comparison are significantly inferior to our approach both in terms of the quality of optimization of the objective function and in terms of the structural metrics SHD, SHD-CPDAG, and SID. We have also improved the Sinkhorn and URS algorithms by adding additional optimization constraints, obtaining comparable results. A full description of this experiment, as well as other experiments, can be found in the corresponding chapter of the thesis.

3.1.2 Learning Guarantees for Quadratic Surrogate Losses

In the paper [61] we analyzed surrogate loss functions with a specific focus on structured prediction. The main theoretical result of the work is a strengthened lower bound on the calibration function for a quadratic surrogate loss, which allows one to obtain non-trivial guarantees in the case when the surrogate loss is not consistent.

Following the notation introduced in Section 2.2, we introduce a quadratic surrogate loss function

$$\Phi_{quad}(f, y) := \frac{1}{2k}\|f + L(:, y)\|_2^2 = \frac{1}{2k}\sum_{\hat{y}\in\mathcal{Y}} \left(f_{\hat{y}}^2 + 2f_{\hat{y}}L(\hat{y}, y) + L(\hat{y}, y)^2 + L(\hat{y}, y)^2\right). \tag{26}$$

As noted above, the prediction f is often parameterized using an additional matrix $F : f(x) = Fg(x)$. Earlier Osokin et al. [44] obtained a calibration function lower bound under the assumption that the linear span $\mathcal{F}$ of the columns of the matrix F coincides with the linear span of the columns of the loss function.

Theorem 3. *For any loss matrix L, the corresponding quadratic surrogate Φ_{quad}, and the prediction space $\mathcal{F}$ containing the columns of the matrix L, the calibration function $H_{\Phi_{quad},L,\mathcal{F}}$ satisfies*

$$H_{\Phi_{quad},L,\mathcal{F}}(\varepsilon) \geq \frac{\varepsilon^2}{2k \max_{i\neq j}\|P_{\mathcal{F}}\Delta_{ij}\|_2^2} \geq \frac{\varepsilon^2}{4k}, \tag{27}$$

where $P_{\mathcal{F}}$ is the orthogonal projection operator onto $\mathcal{F}$ and the vector $\Delta_{ij} = e_i - e_j \in \mathbb{R}^k$, where e_c denotes c- th standard basis vector in $\mathbb{R}^k$.

The latter inequality is trivial and leads to the estimate obtained by Ciliberto et al. [14]. On the other hand, as $\mathcal{F} \subsetneq \mathbb{R}^k$ decreases, the projection norm $\|P_{\mathcal{F}}\Delta_{ij}\|_2^2$ drops, resulting in more accurate lower bounds for the calibration function. The minimum set of scores that satisfies the conditions of the theorem is $\mathcal{F} = \operatorname{span} L$.

In our work, we have relaxed the constraint $\operatorname{span} L \subset \mathcal{F}$, obtaining the following estimate.

Theorem 4. *For any loss matrix L, the corresponding quadratic surrogate Φ_{quad}, and the prediction space $\mathcal{F}$, the calibration function $H_{\Phi_{quad},L,\mathcal{F}}$ satisfies*

$$H_{\Phi_{quad},L,\mathcal{F}}(\varepsilon) \geq \min_{i\neq j}\max_{v\geq 0} \frac{(\varepsilon v - \xi_{ij}(v))_+^2}{2k\|P_{\mathcal{F}}\Delta_{ij}\|_2^2}, \text{ where } \xi_{ij}(v) := \|L^T(vI_k - P_{\mathcal{F}})\Delta_{ij}\|_\infty, \tag{28}$$

the operator $P_{\mathcal{F}}$ defines an orthogonal projection onto $\mathcal{F}$, the function $(x)_+^2 := [x > 0]x^2$ defines the right branch of the parabola and $\Delta_{ij} := e_i - e_j \in \mathbb{R}^k$, where e_c denotes the cth standard basis vector in $\mathbb{R}^k$.

Assuming $v = 1$ in the estimate introduced above, we can also obtain a simplified expression

$$H_{\Phi_{quad},L,\mathcal{F}}(\varepsilon) \geq \min_{i\neq j} \frac{(\varepsilon - \xi_{ij})_+^2}{2k\|P_{\mathcal{F}}\Delta_{ij}\|_2^2}, \text{ where } \xi_{ij} := \|L^T(I_k - P_{\mathcal{F}})\Delta_{ij}\|_\infty. \tag{29}$$

Importantly, for $\operatorname{span} L \subset \mathcal{F}$ the new lower bound 28 is at least as tight as the old one 27. Indeed, the expression inside coincides with the old estimate for $v = 1$, but can deliver a tighter bound when $v \neq 1$. In the case when $\mathcal{F}$ does not contain the columns of $\mathcal{F}$, the lower one will be the envelope of the family of curves with parameter v. Each of the curves is the right branch of a parabola shifted to the right. Near zero, the lower bound is zero due to the inconsistency of the surrogate when $\mathcal{F} \not\subset \operatorname{span} L$. The leftmost point with positive bound is equal to $\eta = \frac{\xi_{ij}(v)}{v}$ and determines the level of consistency of the surrogate in a broad sense.

In addition to deriving a general estimate, we calculate the constants in the inequality and analyze several popular loss functions. As an illustration, we present the loss function mAP (mean average

ER1				
	Val Q - Ql	SHD	SHD-CPDAG	SID
PL-RELAX	15.7±27.3	14.4±5.3	16.0±6.2	61.0±48.7
SINKHORN_{ECP}	10.4±8.7	15.8±4.7	17.0±6.0	84.8±56.3
URS_{ECP}	27.5±34.2	20.6±6.3	21.4±7.2	96.8±74.6
SINKHORN	1651.2±3050.4	24.0±6.1	25.0±6.7	131.2±76.5
GREEDY-SP	N/A	18.6±13.5	18.0±16.6	74.0±53.5
RANDOM	895.1±1270.3	37.8±5.2	38.8±4.9	146.8±79.9
SF1				
	Val Q - Q*	SHD	SHD-CPDAG	SID
PL-RELAX	-1.5±0.2	4.0±0.6	4.6±0.5	4.2±0.7
SINKHORN_{ECP}	1.9±4.3	6.6±2.2	6.6±2.4	10.4±5.0
URS_{ECP}	3.0±2.0	10.6±2.0	10.6±1.6	14.4±4.0
SINKHORN	38.3±26.2	19.0±0.0	19.0±0.0	35.0±2.4
URS	38.3±26.2	19.0±0.0	19.0±0.0	35.0±2.4
GREEDY-SP	N/A	2.0±1.4	0.0±0.0	7.0±5.1
RANDOM	94.0±36.4	36.2±2.6	36.6±2.3	48.6±14.7
ER4				
	Val Q - Q*	SHD	SHD-CPDAG	SID
PL-RELAX	468.8±208.4	71.0±5.9	72.6±3.9	289.6±9.1
SINKHORN_{ECP}	2519.0±3715.2	78.0±6.1	78.8±5.5	302.2±15.8
URS_{ECP}	1011.4±745.5	75.8±2.9	76.6±2.9	300.2±20.3
SINKHORN	126284.6±194386.3	88.8±6.0	91.0±5.7	330.0±14.1
GREEDY-SP	N/A	103.4±10.9	105.6±10.5	288.6±14.7
RANDOM	109891.2±74968.7	113.0±4.9	114.4±4.1	330.6±9.2
SF4				
	Val Q - Q	SHD	SHD-CPDAG	SID
PL-RELAX	-5.8±1.2	20.0±4.3	20.0±4.1	48.4±16.2
SINKHORN_{ECP}	-0.4±2.4	25.6±5.6	25.8±5.9	58.6±19.7
URS_{ECP}	8.5±11.8	30.2±5.8	30.6±5.2	72.2±25.0
SINKHORN	158.2±99.9	44.6±5.8	44.8±6.1	103.6±20.8
URS	140.7±140.6	42.0±5.4	42.8±5.1	89.8±20.4
GREEDY-SP	N/A	50.6±31.5	49.8±32.3	69.0±43.2
RANDOM	635.5±182.6	98.2±6.1	99.2±5.5	168.8±29.6

Table 1: Метрики для графов из 20 вершин

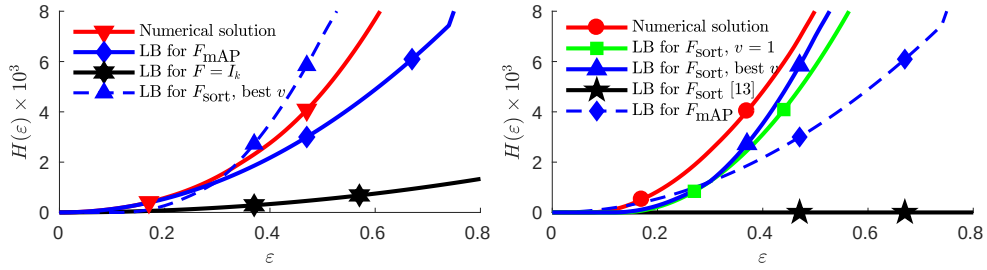


Figure 1: Left: consistent calibration function for F_{mAP}; right: inconsistent calibration function for F_{sort}

precision) used in ranking problems [10, 9, 49]. In this case, the model prediction $\sigma \in \hat{\mathcal{Y}} = S_r$ is a permutation of r elements, and the labels $y \in \mathcal{Y} = \{0,1\}^r$ are binary vectors of length r. The loss function $L_{mAP}(\sigma, y)$ averages the ranking accuracy for different recall levels:

$$L_{mAP}(\sigma, y) := 1 - \frac{1}{|y|} \sum_{p:y_o=1}^{r} \frac{1}{\sigma(p)} \sum_{q=1}^{\sigma(p)} y_{\sigma^{-1}(q)} = 1 - \sum_{p=1}^{r} \sum_{q=1}^{p} \frac{1}{\max(\sigma(p), \sigma(q))} \frac{y_p y_q}{|y|}. \tag{30}$$

Above, the norm of a binary vector is $|y| = \sum_{p=1}^{r} y_p$. The second expression for the loss matrix leads to two natural definitions of $\mathcal{F}$, which we present below. For the first parameterization, we define $\mathcal{F}_{mAP} = \operatorname{span} F_{mAP}$ in terms of the linear span of the columns of the matrix $F_{mAP} \in \mathbb{R}^{r! \times \frac{1}{2} r(r+1)}$ with elements $(F_{mAP})_{\sigma,pq} := \frac{1}{\max(\sigma(p),\sigma(q))}$. It follows from the definition of L_{mAP} that $\operatorname{span} L_{mAP} = \operatorname{span} F_{mAP}$, and the quadratic surrogate loss function is consistent. On the other hand, the derivation to this model reduces to the integer quadratic programming problem $\max_{\sigma \in S_r} (F_{mAP}\theta)_\sigma$, which is NP-hard. For the second parameterization, we define $\mathcal{F}_{sort} = \operatorname{span} F_{sort}$ as the linear span of the matrix $F_{sort} \in \mathbb{R}^{r! \times R}$ with elements $(F_{sort})_{\sigma,p} := \frac{1}{\sigma(p)}$. In this parameterization, inference task $\max_{\sigma \in S_r} (F_{sort}\theta)_\sigma$ is equivalent to sorting the elements of θ, which makes the second parameterization preferable to the first. On the other hand, $\mathcal{F}_{sort}$ does not contain the columns of the matrix L_{mAP}, which makes the quadratic surrogate inconsistent.

The figure 1 shows the graphs of the estimates described above for the loss function L_{mAP}. Due to the inconsistency of the surrogate loss function, the graph for F_{sort} is zero up to a certain $\varepsilon > 0$. At the same time, for some values of ε, the calibration function lower bound for F_{sort} turns out to be higher than the calibration function lower bound for F_{mAP}. In practice, this means that for lower optimization precision, our bound provides stronger learning guarantees for the parameterization F_{sort} with the efficient inference algorithm.

3.2 Applications

3.2.1 Structured Priors for Convolutional Neural Network Kernels

Our work [1] proposes to interpret the parameters of a convolutional neural network as a structured latent variable. Compared to basic Bayesian neural networks, the structured prior distribution of network parameters takes into account dependencies between individual weights of convolutional filters. In experiments, the proposed modification improved the classification quality in a setup with a limited training set, allowed to speed up network training, and allowed to extract low-dimensional data representations without additional training.

Bayesian neural network is a discriminative model at the intersection of Bayesian methods of machine learning and deep learning. The joint distribution

$$p(y^1, \ldots, y^n, \theta \mid x^1, \ldots x^n) = \left[\prod_{i=1}^{n} p(y^i \mid x^i, \theta) \right] p(\theta) \tag{31}$$

on class labels $y^1, \ldots, y^n$ and model weights θ typically consists of a set of independent distributions for each individual weight $p(\theta) = \prod_j p(\theta_j)$ and the likelihood of the label $p(y^i \mid x^i, \theta)$ given the input object x^i and the weights θ. For prediction the model combines the weights posterior distribution $p(\theta \mid \{x^i, y^i\}_{i=1}^n)$

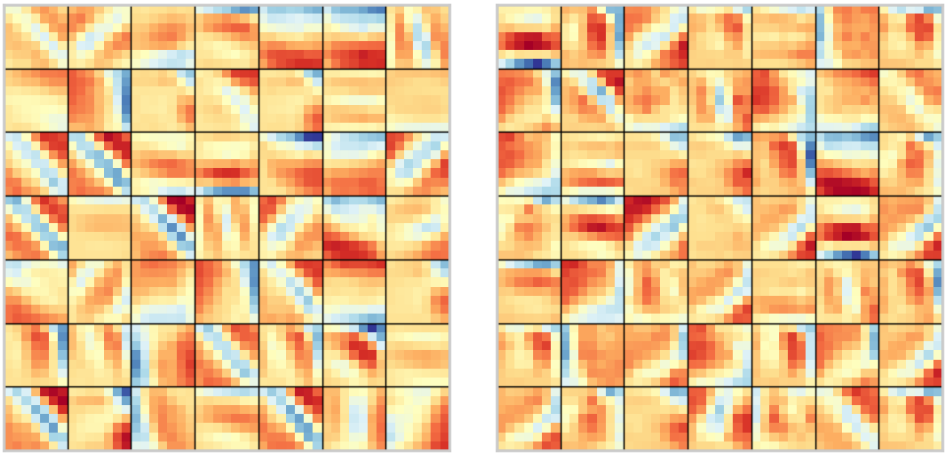


Figure 2: Left: Trained convolutional network filters. Right: filters obtained from approximation.

the label distribution $p(y^i \mid x^i, \theta)$ into a posterior predictive distribution:

$$p(y_{test} \mid \{x^i, y^i\}_{i=1}^n, x_{test}) = \int p(y_{test} \mid x_{test},\ theta) p(\theta \mid \{x^n, y^n\}_{n=1}^N) \mathrm{d}\theta. \tag{32}$$

In practice, the posterior distribution is approximated via variational inference, i.e. by solving the problem

$$\max_{\phi} \left[\mathbb{E}_{\Theta} \log p(y^1, \ldots, y^n \mid \Theta, x^1, \ldots, x^n) - KL(q(\Theta; \phi) || p(\Theta)) \right], \tag{33}$$

where the expectation is taken with respect to the variational distribution with density $q(\theta; \phi)$ and parameters ϕ. The assumption of the independence of the parameters in the prior distribution simplifies the parameterization of the model. However, the weights of the convolutional filters of a trained neural network do not behave like independent random variables. Qualitatively, the weights smoothly change depending on the location in the filter (Figure 2 illustrates the argument). Moreover, in applications, the trained parameters of the convolutional network can be used in a new task on a similar domain [74, 57]. For example, in the case of images, ImageNet-trained convolutional networks can be adapted to other computer vision tasks.

We consider the distribution of convolutional filters on a certain domain. We propose an empirical approximation to the distribution. In particular, we train several convolutional networks on an auxiliary task in the same domain. The auxiliary task must be representative of the given domain: training examples must be diverse, and the network representations must be sufficiently informative. In this work, we considered image classification, the auxiliary task was a classification task with a different training set and a different set of labels. Having trained several convolutional networks, we can build an empirical approximation of filter distribution. However, there are two problems with empirical approximation. First, to work with the distribution, it is necessary to store many convolutional networks in memory. Secondly, the density required to calculate the objective function of the Bayesian neural network is not available for the approximation. Therefore, we propose to approximate the distribution of filters using an auxiliary generative model based on a variational auto-encoder.

When training, we propose to replace the prior distribution of $p(W)$ with an estimate obtained on the basis of a variational auto-encoder. Thus, we arrive at a lower bound on the marginal likelihood

$$\log p(\{y^i\}_{i=1}^n \mid \{x^i\}_{i=1}^n) \geq \mathbb{E}_{\Theta} \left[\log p(\{y^i\}_{i=1}^n \mid \Theta, \{x^i\}_{i=1}^n) \right. \tag{34}$$

$$+ \mathbb{E}_Z \log \frac{p(\Theta \mid Z; \chi) p(Z)}{r(Z \mid \Theta; \psi)} \tag{35}$$

$$\left. - \log q(\Theta; \phi) \right], \tag{36}$$

where $q(\Theta; \phi)$ is a variational approximation of the network parameters, the distributions of $p(\Theta \mid Z; \chi)$ and $r(Z \mid \Theta; \psi)$ are determined by the variational auto-encoder , and the expectation with respect to the random vector Z is calculated with respect to the distribution $r(Z \mid \Theta; \psi)$. When training a Bayesian

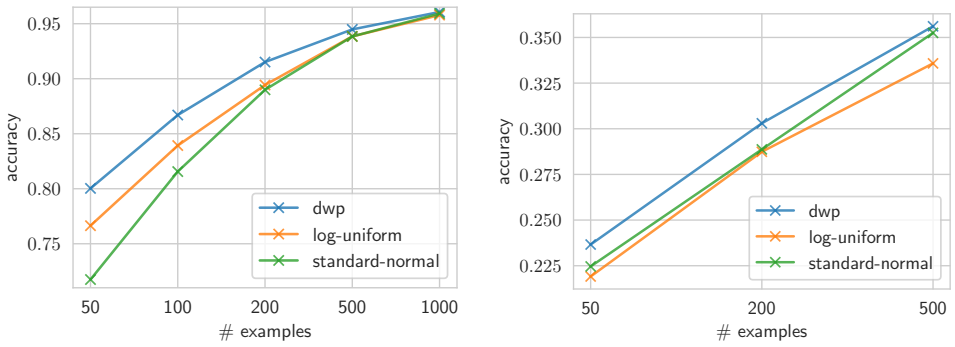


Figure 3: Classification quality depending on the size of the training sample. Left: MNIST, right: CIFAR-10.

neural network, we will use this estimate as an objective function. The first term in the estimate corresponds to the standard cross-entropy loss function, and the second term pulls the approximate posterior distribution $q(\Theta;\phi)$ towards the empirical prior distribution of the convolutional network parameters for the given domain $p(\Theta)$.

To evaluate the proposed approach, we conducted a series of experiments that evaluated the learning ability with limited training data, the representations the network obtains after the initialization from the prior distribution, and the learning time depending on the weight prior distribution. Here we restrict ourselves to the first experiment, a detailed description of the others can be found in the corresponding chapter.

While studying Bayesian network training with limited data, we considered the classification problem on MNIST and CIFAR-10. As a model, we took convolutional networks consisting of several convolutional layers along with several fully connected output layers. For convolutional layers, we trained the prior distribution on NotMNIST and CIFAR-100 data. We trained the fully connected layers with backpropagation without resorting to variational inference. For comparison, we considered the prior distributions for network parameters common in the literature: the Gaussian distribution and the log-uniform distribution, among which the latter guarantees the invariance of the prior distribution to the scale of the parameters.

For the three prior distribution families we trained classifiers with varying sizes of training data. As Figure 3 shows, the network with the proposed prior distribution performs better. On the MNIST data, the difference in quality disappears when the training sample size is sufficient. On CIFAR-10 data, the quality is uniformly higher. We assume that the difference can be explained by the simplicity of the classification task on the MNIST data: thousands of examples are enough to extract the necessary information from the data.

3.2.2 Bayesian Estimation of Multiple Access Channel Configuration

Probabilistic Model. In the paper [60], we consider the problem of estimating the parameters of a multi-user communication channel. To establish a connection on dedicated frequencies, users send special code signals, which are then received and processed by a cellular communication station. The communication channel is multi-user, so some frequencies can be occupied by several users and the system must be able to detect users by receiving a superposition of the sent code signals. In practice, to simplify the task, it is assumed that there are quite a few users.

In fact, the problem can be interpreted as a structured prediction problem, where, based on the received signals, it is necessary to choose a sparse binary vector with a block structure. Standard solutions are based on modifications of compressed sensing algorithms. The work [70] proposed an approach based on Bayesian linear regression. Bayesian linear regression allows finding sparse solutions to linear systems of equations, which is required in this problem. In our work, we adapted the standard Bayesian linear regression model to the specifics of the problem, taking into account the block structure of the desired solution, and also proposed a faster algorithm for solving the task.

Mathematical model of the communication process is a system of linear equations

$$y = \kappa\theta + z, \tag{37}$$

where the vector y corresponds to the received signal, the matrix κ is fixed and specified by the communication protocol, the vector θ is unknown and describes the channel configuration, and z models the noise that occurs during signal transmission. As a noise model, we used a Gaussian distribution with a known variance ρ. In addition, the vector θ has a block structure

$$\theta = (c_{11}t_1, \ldots, c_{1Q}t_1, c_{21}t_2, \ldots, c_{2Q}t_2, \ldots, c_{N1}t_N, \ldots, c_{NQ}t_N), \tag{38}$$

where the binary variables $t_1, \ldots, t_N \in \{0, 1\}$ are equal to one if the user is active, and the values $c_{11}, \ldots, c_{NQ} \in \mathbb{R}$ reflect the physical parameters of the communication channel. Within this model, we are primarily interested in recovering the $t_1, \ldots, t_N$ values that indicate active users in the channel. In addition, we are also interested in estimating the vector θ, since it contains signal fading parameter.

To solve the problem, we consider a Bayesian linear regression model with the following joint distribution

$$p(y, \theta; \rho, \gamma) = p(y \mid \theta; \rho)p(\theta; \gamma) \tag{39}$$

$$p(Y = y \mid \Theta = \theta; \rho) = \mathcal{N}(y \mid \kappa\theta; \rho I) \tag{40}$$

$$p(\Theta = \theta; \gamma) = \mathcal{N}(\theta \mid 0, \mathrm{diag}(\underbrace{\gamma_1, \ldots, \gamma_1}_{Q}, \ldots,\ underbrace\gamma_N, \ldots, \gamma_{NQ})). \tag{41}$$

The density $p(Y = y \mid \Theta = \theta; \rho)$ specifies the observations likelihood, and $p(\Theta = \theta; \gamma)$ specifies the prior distribution with the block structure. Note that in the previously proposed works, the basic regression model was used, which did not take into account the block structure of the prior distribution.

To estimate the channel configuration, we maximize the evidence $p(y; \rho, \gamma)$ with respect to the prior distribution parameters γ. Similarly to [70], we use the EM-algorithm for the derivation, alternately estimating the posterior distribution $p(\theta \mid y, \rho, \gamma)$ at the E-step and maximizing the evidence estimate with respect to γ at the M-step. At the M-step, we use the iterative scheme proposed in [64]. For our problem, the scheme improved the inference speed in model experiments compared to the previously considered schemes [70].

Simulation results. We ran a simulation to evaluate the performance of the proposed scheme. We compared reconstruction error of a model with a custom probabilistic model and the improved iterative scheme against the solution proposed in [70]. We used the Rayleigh fading model to model the signal amplitude, considered a channel with 6 active users out of $N = 36$, each using $Q = 5$ frequencies. We used Zadov-Chu sequences of length 20 to construct the codebook matrix κ. The graph 4 shows the dependence of the average proportion of incorrectly identified users $\mathrm{UDER} = \mathbb{E}\left[\ frac\sum_{n=1}^{N}[\hat{a}_n \neq a_n N\right]$ on the number of iterations of the EM-algorithm for different signal-to-noise ratio levels in the communication channel. In all four cases, the proposed scheme converges faster than the original scheme. Moreover, for high noise levels, the original scheme does not converge on average. The graph 5 shows the dependence of the average Θ estimation error on the number of iterations of the EM-algorithm. As in the previous experiment, the proposed scheme shows the best convergence. It is noteworthy that in terms of the MSE metric, the original scheme achieves comparable results even for high noise levels.

3.3 Pre-processing of Geological Survey Data with Hidden Markov Chains

Probabilistic model. The last chapter focuses on the analysis of geological survey data. In our work [59], we adapt the hidden Markov chain to the task of pre-processing and imputation for missing data in well logging. The Hidden Markov Chain is one of the classic probabilistic models with a latent structured variable: the hidden variable is given by a Markov chain with discrete states, the observations are independent under given the hidden variable, and the EM-algorithm is used to tune the parameters and infer the hidden variable.

In the first stages, the goal of a geological survey is to build a model of the deposit. The model is based on number wells drilled on the territory of the field, the survey data is collected with a number of sensors that are lowered into each well. As you move deeper, the sensors read different physical characteristics of the well depending on the depth. The acquired data form sequences referred to as "logs". Then, based on this data, an expert petrophysicist labels segments of wells that are of interest from the viewpoint of field development. To produce the labels the expert also performs data alignment, additional calibration and anomaly search.

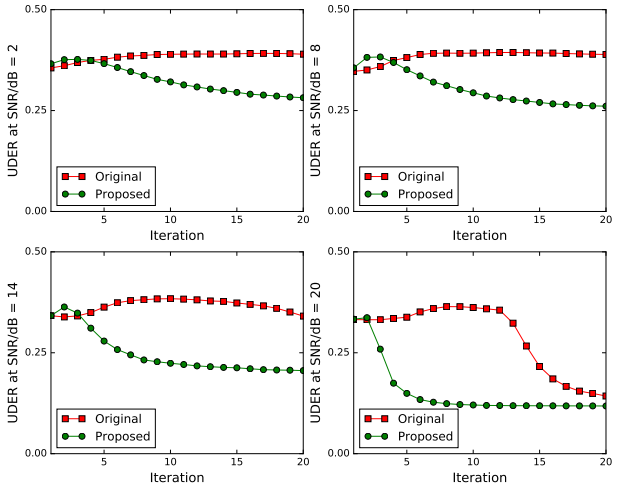


Figure 4: Dependence of the UDER (user detection error) on the number of iterations of the EM algorithm

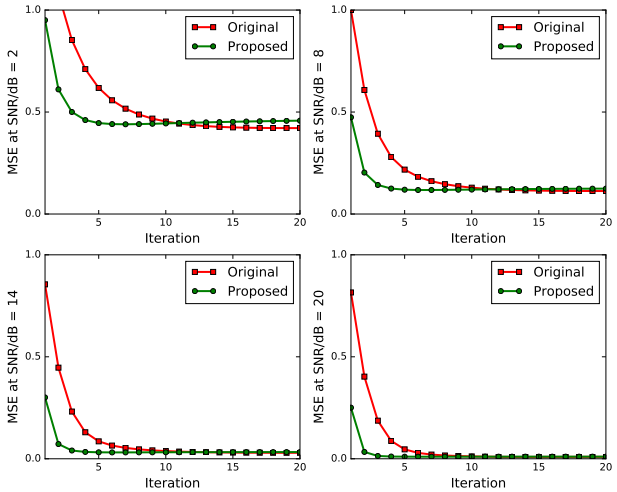


Figure 5: Dependence of the Θ communication channel parameters estimation error on the number of iterations of the EM algorithm

The goal of this study is to automate data processing steps of an expert petrophysicist. The results of the work of experts accumulated over many years make it possible to solve well labeling as a supervised learning task, however, the predictions of experts are subjective and may not provide an insufficiently reliable training signal. Therefore, we considered an unsupervised learning setup that could provide consistent predictions across all the wells.

Next we describe the proposed probabilistic model. Let $x^1, \ldots, x^K$ be logs for for K wells, $x^k \in \mathbb{R}^{l_k \times d}$. For each well at a given depth level, the sensor reading is primarily determined by the soil characteristics. We assume that soil characteristics can be described by a sequence of m states of a Markov chain. To define the Markov chain we introduce random vectors $T^1, \ldots, T^K$, $T^k_l \in \{1, \ldots, m\}, l = 1, \ldots, L^k k$, initial distribution $P(T^k_1 = t; \pi) \propto \pi_k, \pi \in \mathbb{R}^m_+$ and consecutive pair distributions $P(T^k_l = t \mid T^k_{l-1} = s; \tau) \propto \tau_{ts}, \tau \in \mathbb{R}^{m^2}_+$. The dependence of the elements of the chain allows to promote identical states for adjacent segments. Continuous segments of the chain with a constant latent state correspond to homogeneous sections of the well, along which soil characteristics do not change. In practice, soil characteristics are unknown, so the Markov chain acts as a latent variable in the model. However, we know the sensor readings in the logs. We assumed that for each type of soil, the sensor readings follows the multivariate Gaussian distribution $p(x^k_l \mid T^k_l = t; \mu, \Sigma) = \mathcal{N}(x^k_l \mid \mu_t, \Sigma_t), \mu \in \mathbb{R}^{m \times d}, \Sigma \in \mathbb{R}^{m \times d^2}$.

Besides that, sensor readings are affected by instrument calibration prior to recording. Assuming that sensor calibration can be represented as a linear transformation of the readings $x^k_l = \alpha_k \odot \hat{x}^k_l + \beta_k$ for a given observation x^k_l and a calibrated observation $\hat{x}^k_l$, we introduce additional calibration parameters $\alpha \in \mathbb{R}^{K \times d}_*, \beta \in \mathbb{R}^{K \times d}$ for each well. For the parameters $\Theta = (\pi, \tau, \alpha, \beta)$, the final observational model is

$$p(x^i, t^{i\,K}_{i=1}; \Theta) = \tag{42}$$

$$\prod_{k=1}^{K} [(\pi_{t^k_1} \prod_{l=2}^{L_k} \tau_{t^k_l t^k_{l-1}}) \times \tag{43}$$

$$\prod_{l=1}^{L_k} \mathcal{N}(x^k_l \mid \alpha_k \odot \mu_{t^k_l} + \beta_k, \operatorname{diag}(\alpha_k) \Sigma_{t^k_l} \operatorname{diag}(\alpha_k))]. \tag{44}$$

To tune the model parameters, we use the Baum-Welch algorithm to maximize the evidence lower bound of the model. We maximize the lower bound with stochastic gradient descent, starting from a hand-crafted initialization to avoid local optima. Since the observations in the model follow the Gaussian distribution, we could incorporate data with gaps using marginal distributions as a likelihood, without taking into account the missing data. We used the Viterbi algorithm to predict the hidden states of the circuit. Below we present the results of the model operation on synthetic fields, as well as on the Priobskoye field [3].

Empirical results. We started with a synthetic field, for which we both have measurements for wells, and the ground truth labels for soil types. As a result, we were able to qualitatively compare the hidden states of the Markov chain with the ground truth labels, thus eliminating the factor of subjective data interpretation by an expert. On the left, the graph 6 contains the logs (blue lines) as well as the predictions of our model (green line). While we were able to accurately replicate the behavior of the logs, we did not get a one-to-one correspondence between latent states and soil types. The ground truth labels and hidden states of the well is shown in the graph 6 on the right. The selected number of latent states exceeded the number of soil types: increasing the number of latent states improves the approximation of logs, but makes the latent states less interpretable. On the graph 7 we have shown the correspondence between latent states and soil types throughout the field. Most of the latent states correspond to the argillite prevailing in the deposit. The model was also able to separate tight rocks and sandstones, but none of the hidden states correspond to siltstone.

We then applied the model to pre-process the data at the Priobskoye field. The base model predicted reservoir layers (layers of interest in terms of oil production) using a binary classification based on a recurrent neural network [3]. In the base model, instrument readings were standardized to account for miscalibration. We, in turn, calibrated the reading using the calibration parameters α, β obtained with a hidden Markov model instead of standardizing the data. The new pre-processing algorithm did not give a significant improvement in the quality of the prediction, increasing the F1 score from 0.72 to 0.74.

Next, we used the model to fill the gaps in the data. We considered test wells for which there are no ILD (deep induction log) and LLD (lateral log) log values in the sample. We then compared two gap recovery strategies: replacing the log with the average across the field, and our approach of restoring

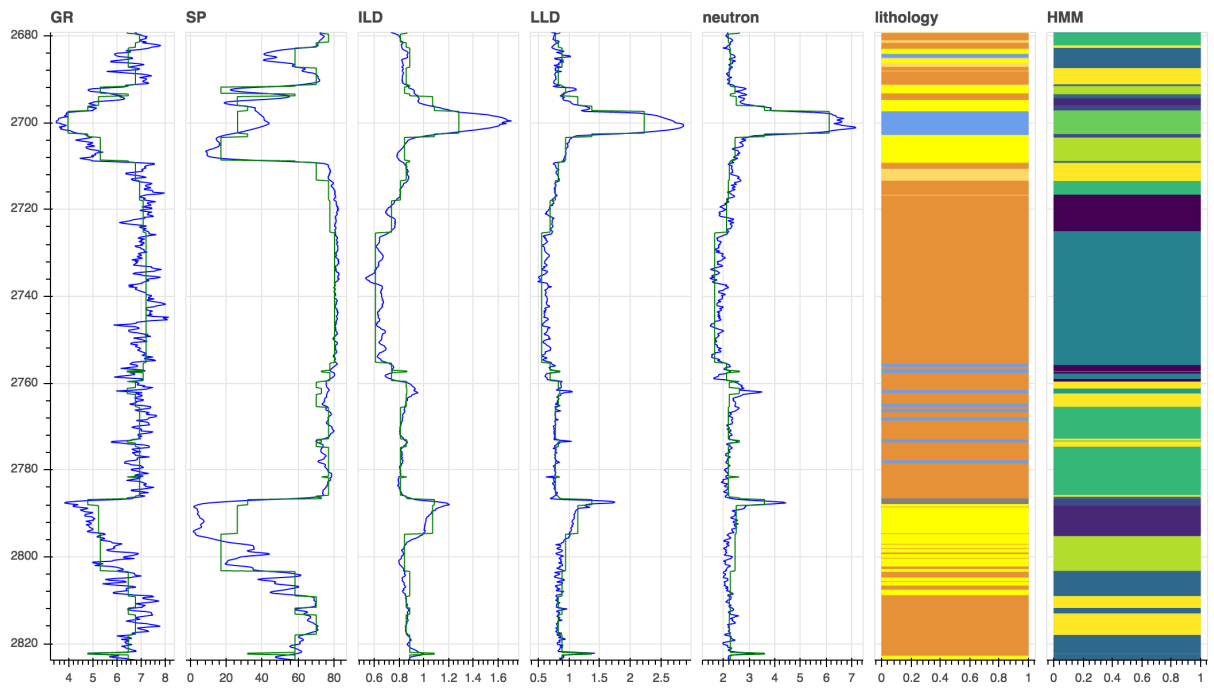


Figure 6: An illustration of how the model works for one well. Left: Example of observed logs (blue) and their approximation (green); right: expert labels compared to the found states of the Markov chain.

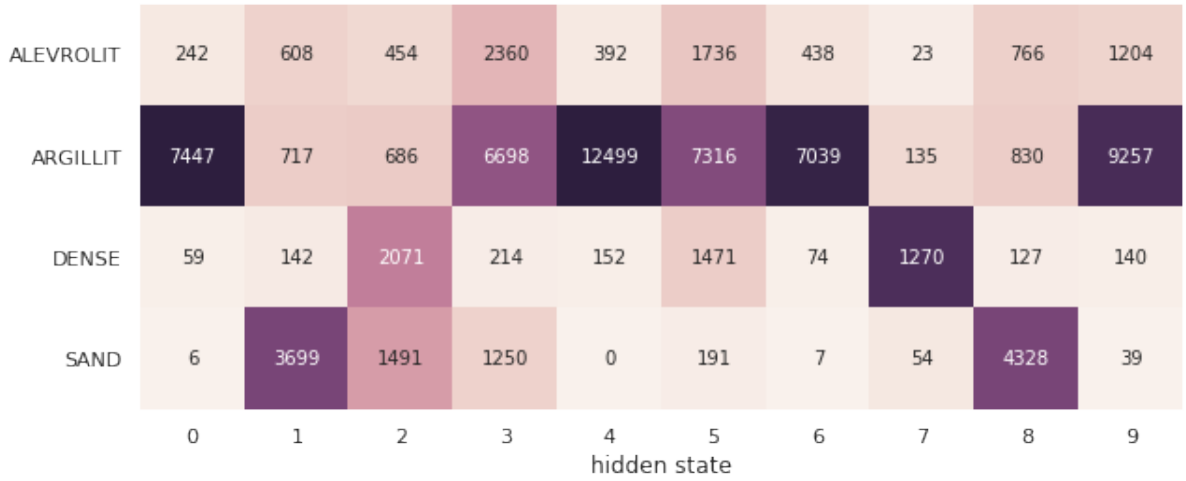


Figure 7: Correspondence between soil types and hidden states of the Markov chain

the log from the rest of the logs using a Markov chain. The proposed solution improved the prediction quality for the considered wells from F1=0.37 to F1=0.56. Thus, the proposed approach allows us to improve the quality of finding reservoir layers due to joint calibration and recovery of gaps in the data.

4 Conclusion

The results described above cover various aspects of structured prediction, including theoretical analysis of the standard structured prediction setup, models with latent structured variables based on a probabilistic approach, as well as applications of the described solutions to real problems. In conclusion, we briefly summarize the presented results.

1. We proposed a permutation optimization method based on probabilistic relaxation and the REINFORCE algorithm; we developed control variates to improve the convergence of the method. We evaluated the method on the problem of identifying causal links in data, where the topological sorting of a directed acyclic link graph acts as a structured variable. The proposed method significantly improved structure reconstruction metrics in comparison with relaxation-based gradient optimization methods. Since the considered optimization method does not introduce additional assumptions about the objective function and is actually a zero-order optimization method, in the future it can also be used for direct optimization of the objective function in structured prediction problems (without using auxiliary surrogate loss functions), as well as for amortized inference of permutations.

2. In a supervised structured prediction setup, we analyzed a training approach based on quadratic surrogate loss functions. In particular, we considered the case of inconsistent surrogate loss function, for which we obtained guarantees for the accuracy of the expected risk optimization. Assuming a fixed number of training samples and early optimization stopping, the analysis delivers tighter upper bounds on the expected risk values. From a practical point of view, the above formulation also leads to more efficient inference algorithms.

3. We considered a number of applications based on a probabilistic approach to structured prediction. First, we applied the variational auto-encoder model to infer the parameters of a convolutional neural network based on a priori knowledge about the network parameter distribution for a given domain. In this case, the parameters of the convolutional filters act as a latent structured variable, and the proposed approach improves the prediction accuracy of Bayesian neural networks for similar domains. Second, we considered the task of estimating the parameters of a multi-user communication channel, where the subset of active users acts as a hidden structured variable. In this case, we proposed an improved probabilistic model to estimate the structured variable, and accelerated the inference algorithm. Finally, we proposed a probabilistic model based on hidden Markov chains to model and interpret geophysical survey data. The proposed model uses structured variables, in this case the Markov chain hidden states, to infer and cluster the physical characteristics of the wells while modeling the joint distribution of these characteristics. Based on the reconstructed hidden states, we proposed an approach to data imputation and anomaly detection.

References

[1] Andrei Atanov, Arsenii Ashukha, Kirill Struminsky, Dmitriy Vetrov, and Max Welling. The deep weight prior. In *International Conference on Learning Representations*, 2018.

[2] Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[3] Boris Belozerov, Nikita Bukhanov, Dmitry Egorov, Adel Zakirov, Oksana Osmonalieva, Maria Golitsyna, Alexander Reshytko, Artyom Semenikhin, Evgeny Shindin, and Vladimir Lipets. Automatic well log analysis across priobskoe field using machine learning methods. In *SPE Russian Petroleum Technology Conference*. OnePetro, 2018.

[4] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[5] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable pertubed optimizers. *Advances in neural information processing systems*, 33:9508–9519, 2020.

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[7] Céline Brouard, Marie Szafranski, and Florence d'Alché Buc. Input output kernel regression: Supervised and semi-supervised structured output prediction with operator-valued kernels. *Journal of Machine Learning Research*, 17:np, 2016.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[9] David Buffoni, Clément Calauzenes, Patrick Gallinari, and Nicolas Usunier. Learning scoring functions with order-preserving losses and standardized supervision. In *ICML*, 2011.

[10] Clément Calauzènes, Nicolas Usunier, and Patrick Gallinari. On the (non-) existence of convex, calibrated surrogate losses for ranking. In *NIPS*, 2012.

[11] Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pages 883–892. PMLR, 2018.

[12] Justin Chiu and Alexander M Rush. Scaling hidden markov language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1341–1349, 2020.

[13] Anna Choromanska, Alekh Agarwal, and John Langford. Extreme multi class classification. In *NIPS Workshop: eXtreme Classification, submitted*, volume 1, pages 2–1, 2013.

[14] Carlo Ciliberto, Lorenzo Rosasco, and Alessandro Rudi. A consistent regularization approach for structured prediction. *Advances in neural information processing systems*, 29:4412–4420, 2016.

[15] Shay B Cohen and Noah A Smith. Joint morphological and syntactic disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 208–217, 2007.

[16] Caio Corro and Ivan Titov. Differentiable perturb-and-parse: Semi-supervised parsing with a structured variational autoencoder. *arXiv preprint arXiv:1807.09875*, 2018.

[17] Corinna Cortes, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Structured prediction theory based on factor graph complexity. *Advances in Neural Information Processing Systems*, 29, 2016.

[18] David Roxbee Cox and David Victor Hinkley. *Theoretical statistics*. CRC Press, 1979.

[19] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292, 2001.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[21] Artyom Gadetsky, Kirill Struminsky, Christopher Robinson, Novi Quadrianto, and Dmitry Vetrov. Low-variance black-box gradient estimates for the plackett-luce distribution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10126–10135, 2020.

[22] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR, 2017.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[24] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

[25] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[26] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. In *International Conference on Learning Representations*, 2018.

[27] Jiatao Gu, Qi Liu, and Kyunghyun Cho. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676, 2019.

[28] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[29] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net, 2017.

[30] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[31] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

[32] Yoon Kim, Sam Wiseman, and Alexander M Rush. A tutorial on deep latent variable models of natural language. *arXiv preprint arXiv:1812.06834*, 2018.

[33] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

[34] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[35] Yi Lin. A note on margin-based loss functions in classification. *Statistics & probability letters*, 68(1):73–82, 2004.

[36] Ben London, Bert Huang, and Lise Getoor. Stability and generalization in structured prediction. *The Journal of Machine Learning Research*, 17(1):7808–7859, 2016.

[37] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[38] C Maddison, A Mnih, and Y Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the international conference on learning Representations*. International Conference on Learning Representations, 2017.

[39] André FT Martins, Tsvetomila Mihaylova, Nikita Nangia, and Vlad Niculae. Latent structure models for natural language processing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 1–5, 2019.

[40] Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations*, 2018.

[41] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[42] Alex Nowak-Vila, Francis Bach, and Alessandro Rudi. A general theory for structured prediction with smooth convex surrogates. *arXiv preprint arXiv:1902.01958*, 2019.

[43] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[44] Anton Osokin, Francis R Bach, and Simon Lacoste-Julien. On structured prediction theory with calibrated convex surrogate losses. In *NIPS*, 2017.

[45] Max Benedikt Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. Gradient estimation with stochastic softmax tricks. In *NeurIPS 2020*, 2020.

[46] Judea Pearl. *Causality*. Cambridge university press, 2009.

[47] Fabian Pedregosa, Francis Bach, and Alexandre Gramfort. On the consistency of ordinal regression methods. *Journal of Machine Learning Research*, 18:1–35, 2017.

[48] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[49] Harish G Ramaswamy, Shivani Agarwal, and Ambuj Tewari. Convex calibrated surrogates for low-rank loss matrices with applications to subset ranking losses. In *NIPS*, 2013.

[50] D Raj Reddy et al. Speech understanding systems: A summary of results of the five-year research effort. *Department of Computer Science. Camegie-Mell University, Pittsburgh, PA*, 17:138, 1977.

[51] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[52] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

[53] BTCGD Roller, C Taskar, and D Guestrin. Max-margin markov networks. *Advances in neural information processing systems*, 16:25, 2004.

[54] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[55] Yasubumi Sakakibara, Michael Brown, Richard Hughey, I Saira Mian, Kimmen Sjölander, Rebecca C Underwood, and David Haussler. Stochastic context-free grammers for trna modeling. *Nucleic acids research*, 22(23):5112–5120, 1994.

[56] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):1–21, 2021.

[57] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[58] Noah A Smith. Linguistic structure prediction. *Synthesis lectures on human language technologies*, 4(2):1–274, 2011.

[59] K Struminskiy, A Klenitskiy, A Reshytko, D Egorov, A Shchepetnov, A Sabirov, D Vetrov, A Semenikhin, O Osmonalieva, and B Belozerov. Well log data standardization, imputation and anomaly detection using hidden markov models. In *Petroleum Geostatistics 2019*, volume 2019, pages 1–5. European Association of Geoscientists & Engineers, 2019.

[60] Kirill Struminsky, Stanislav Kruglik, Dmitry Vetrov, and Ivan Oseledets. A new approach for sparse bayesian channel estimation in scma uplink systems. In *2016 8th International Conference on Wireless Communications & Signal Processing (WCSP)*, pages 1–5. IEEE, 2016.

[61] Kirill Struminsky, Simon Lacoste-Julien, and Anton Osokin. Quantifying learning guarantees for convex but inconsistent surrogates. In *NeurIPS*, 2018.

[62] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[63] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. *Advances in neural information processing systems*, 16, 2003.

[64] Michael E Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244, 2001.

[65] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, Yasemin Altun, and Yoram Singer. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(9), 2005.

[66] George Tucker, Andriy Mnih, Chris J Maddison, Dieterich Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *arXiv preprint arXiv:1703.07370*, 2017.

[67] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[68] Paul Voigt and Axel Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

[69] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

[70] Yufeng Wang, Shidong Zhou, Limin Xiao, Xiujun Zhang, and Jin Lian. Sparse bayesian learning based user detection and channel estimation for scma uplink systems. In *2015 International Conference on Wireless Communications & Signal Processing (WCSP)*, pages 1–5. IEEE, 2015.

[71] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[72] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[73] John I Yellott Jr. The relationship between luce's choice axiom, thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology*, 15(2):109–144, 1977.

[74] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, 2014.

Appendix A Article. Low-variance Black-box Gradient Estimates for the Plackett-Luce Distribution

Authors. Artyom Gadetsky, Kirill Struminsky, Christopher Robinson, Novi Quadrianto, Dmitry Vetrov
Published in: Proceedings of the AAAI Conference on Artificial Intelligence. – 2020. – Т. 34. – №. 06. – С. 10126-10135

Abstract. Learning models with discrete latent variables using stochastic gradient descent remains a challenge due to the high variance of gradient estimates. Modern variance reduction techniques mostly consider categorical distributions and have limited applicability when the number of possible outcomes becomes large. In this work, we consider models with latent permutations and propose control variates for the Plackett-Luce distribution. In particular, the control variates allow us to optimize black-box functions over permutations using stochastic gradient descent. To illustrate the approach, we consider a variety of causal structure learning tasks for continuous and discrete data. We show that our method outperforms competitive relaxation-based optimization methods and is also applicable to non-differentiable score functions.

Low-variance Black-box Gradient Estimates for the Plackett-Luce Distribution

Artyom Gadetsky,[1*†] **Kirill Struminsky,**[1*] **Christopher Robinson,**[2]
Novi Quadrianto,[1, 2] **Dmitry Vetrov**[1‡]
[1] National Research University Higher School of Economics
[2] Predictive Analytics Lab (PAL), University of Sussex

Abstract

Learning models with discrete latent variables using stochastic gradient descent remains a challenge due to the high variance of gradient estimates. Modern variance reduction techniques mostly consider categorical distributions and have limited applicability when the number of possible outcomes becomes large. In this work, we consider models with latent permutations and propose control variates for the Plackett-Luce distribution. In particular, the control variates allow us to optimize black-box functions over permutations using stochastic gradient descent. To illustrate the approach, we consider a variety of causal structure learning tasks for continuous and discrete data. We show that our method outperforms competitive relaxation-based optimization methods and is also applicable to non-differentiable score functions.

Introduction

The vast majority of modern machine learning advancements share one central method - gradient-based optimization. Stochastic gradients give a scalable solution for learning, applicable when the loss function is too slow to compute due to the size of data or even intractable. The latter is often the case when the loss function includes an expectation over random latent variables. The objectives of this kind naturally arise in multiple settings, including probabilistic latent variable models (Neal and Hinton 1998) and reinforcement learning (Williams 1992). Often the distribution of random variables also depends on the optimizable parameters of the loss function, which in turn makes gradient estimation harder and less reliable due to the high variance of stochastic gradients.

Despite the recent breakthroughs in gradient estimation for continuous latent variables (Kingma and Welling 2013; Rezende, Mohamed, and Wierstra 2014; Mohamed et al. 2019), gradient estimation for discrete latent variables remains a challenge. Currently, general-purpose estimators (Williams 1992; Mnih and Gregor 2014) remain unreliable and the state-of-the-art methods (Tucker et al. 2017; Grathwohl et al. 2018; Yin and Zhou 2018) exclusively consider the categorical distribution. Although the reduction to the categorical case allows benefiting from gradient estimators for continuous relaxations, such solutions are hard to translate to discrete distributions with large support.

In this work, we consider a gradient estimator for the Plackett-Luce distribution, a distribution over permutations. Permutations naturally occur in various setting, such as ranking problems (Guiver and Snelson 2009), optimal routing (Bello et al. 2016) and causal inference (Friedman and Koller 2003). However, the support of the distribution is superexponential in the number of items k, which makes representing a distribution as a categorical distribution intractable even for dozens of items. At the same time, the Plackett-Luce distribution has $O(k)$ parameters and allows sampling in $O(k \log k)$.

We translate the recent variance reduction techniques (Tucker et al. 2017; Grathwohl et al. 2018) to the case of Plackett-Luce distributions. Similarly to REBAR, we use the difference of the REINFORCE estimator and the reparametrized estimator for the relaxed model. In particular, we derive the conditional marginalization step (Tucker et al. 2017) for the Plackett-Luce case. In our experiments, we recast causal inference tasks as a variational optimization over permutations and solve it using a gradient optimization method. We show that our method outperforms competitive relaxation-based approaches for optimization over permutations (Grover et al. 2019; Mena et al. 2018) for differentiable score functions and is applicable in a wider range of scenarios.

Our main contributions are the following:

- We derive a low-variance gradient estimator for the Plackett-Luce distribution.
- We apply the gradient estimator to solve variational optimization tasks for black-box functions and concentrate primarily on causal inference tasks for continuous and discrete data.
- For differentiable functions, we show that relaxation-based gradient optimization does not work out-of-the-box

*Both authors contributed equally to this work.
†Corresponding author. E-mail: artygadetsky@yandex.ru
‡Samsung-HSE Laboratory

for causal inference tasks and propose additional constraints to achieve competitive results.

A Brief Tour of Gradient Estimation

We consider a general optimization task $\min_\theta \mathbb{E}_{p(b|\theta)}[f(b)]$, where b is a discrete random variable parametrized by θ. The expectation can be intractable, for instance when b is a vector of categorical variables and the support of b is exponential in the vector length. The standard solution is to construct a stochastic estimate for the gradient $\hat{g}(f) := \frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}[f(b)]$ without explicitly computing the expectation. In this section, we briefly review the gradient estimation algorithms.

REINFORCE

The REINFORCE estimator (Williams 1992) gives us a widely-applicable unbiased estimate for the gradient

$$\hat{g}_{\textit{REINFORCE}}(f) = f(b)\frac{\partial}{\partial\theta}\log p(b \mid \theta),\ \ b \sim p(b \mid \theta). \quad (1)$$

Although an unbiased gradient estimate is sufficient to guarantee convergence of stochastic gradient descent, in practice, the algorithm may not converge due to the high variance of the estimate (Tucker et al. 2017). The variance of the REINFORCE estimator can be reduced using control variates. A Control variate is a function $c(b)$ with a zero mean $\mathbb{E}_{p(b|\theta)}[c(b)] = 0$ that can be used to define another unbiased estimator

$$\hat{g}_{\textit{CV}}(f) = \hat{g}_{\textit{REINFORCE}}(f) - c(b). \quad (2)$$

The variance of the new estimator $\hat{g}_{\textit{CV}}(f)$ is lower than the variance of $\hat{g}_{\textit{REINFORCE}}(f)$ if $c(b)$ is positively correlated with the random variable $f(b)$. As an illustration, the gradient of probability $\frac{\partial}{\partial\theta}\log p(b \mid \theta)$ has zero mean, therefore it can be used as a control variate (Mnih and Gregor 2014).

Reparametrization Gradients for Continuous Relaxations

The reparametrization trick (Kingma and Welling 2013; Rezende, Mohamed, and Wierstra 2014) is an alternative unbiased low-variance gradient estimator, applicable when f is differentiable and the latent variable $b_{\textit{cont}}$ is continuous. The estimator represents the latent variable as a differentiable determinisitc transformation $b_{\textit{cont}} = T(v, \theta)$ of a fixed distribution sample v and parameters θ and estimates the gradient as

$$\hat{g}_{\textit{reparam}}(f) = \frac{\partial}{\partial\theta}f(b_{\textit{cont}}) = \frac{\partial f}{\partial T}\frac{\partial T}{\partial\theta}, \quad (3)$$

$$v_i \sim \mathrm{uniform}[0, 1],\ i = 1, \ldots, k. \quad (4)$$

Although the reparametrization trick is not applicable when the latent variable b is discrete, (Jang, Gu, and Poole 2016; Maddison, Mnih, and Teh 2016) proposed the Gumbel-softmax estimator, a modification of the reparametrization trick for the relaxed categorical distribution.

To sample from a relaxed categorical distribution $p(b \mid \theta)$ with probabilities $\frac{\exp\theta_i}{\sum_j \exp\theta_j}$, Gumbel-Softmax first samples a vector of independent Gumbel random variables $z_i \sim \mathcal{G}(\theta_i, 1), i = 1, \ldots, k$

$$z_i = T(\theta_i, v_i) = \theta_i - \log(-\log(v_i)) \quad (5)$$

$$v_i \sim \mathrm{uniform}[0, 1],\ i = 1, \ldots, k \quad (6)$$

with location parameter θ. According to the **Gumbel-max trick** (Maddison, Tarlow, and Minka 2014), the index of the maximal element $H(z) = \arg\max(z)$ is a categorical random variable with distribution $p(b \mid \theta)$. Then, to make the sampler differentiable, the Gumbel-softmax trick replaces $\arg\max(z)$ with a relaxation $\mathrm{soft}\max(z) = \frac{1}{\sum \exp z_i}(\exp z_1, \ldots, \exp z_k)$. The gradient estimate is the reparametrization gradient for the relaxed categorical distribution:

$$\hat{g}_{\textit{Gumbel}}(f) = \frac{\partial}{\partial\theta}f(b) = \frac{\partial f}{\partial b}\frac{\partial b}{\partial z}\frac{\partial z}{\partial\theta}, \quad (7)$$

$$b = \mathrm{soft}\max(z), \quad (8)$$

$$z_i \sim \mathcal{G}(\theta_i, 1), i = 1, \ldots, k. \quad (9)$$

The resulting reparametrization gradient $\hat{g}_{\textit{Gumbel}}(f)$ has much lower variance than $\hat{g}_{\textit{REINFORCE}}(f)$, but is generally biased due to the relaxation.

Relaxation-based Control Variates

Recently, Tucker et al. (2017) and Grathwohl et al. (2018) proposed control variates for REINFORCE estimator based on the relaxed conditional distribution. Both works use the REINFORCE gradient estimator for the relaxed categorical distribution as a control variate for the non-relaxed estimator. To eliminate the bias of the REINFORCE estimator, they subtract the low-variance reparametrization gradient estimator.

The key insight of Tucker et al. (2017) is the conditional marginalization step used to correlate the non-relaxed REINFORCE estimator and the control variate. Importantly, the conditional marginalization relies on reparametrization trick for the conditional distribution $p(z \mid b, \theta)$, obtained from the joint distribution $p(b, z \mid \theta) = p(b|z)p(z \mid \theta)$ of the Gumbel random vector z and the output of the Gumbel-max trick $b = H(z) = \arg\max(z)$. Tucker et al. (2017) derive a reparametrizable sampling scheme for $p(z \mid b, \theta)$

$$\tilde{z}_i = \begin{cases} -\log(-\log v_i) & i = b \\ -\log\left(-\frac{\log v_i}{\exp\theta_i} + \exp(-\tilde{z}_b)\right) & i \neq b \end{cases}, \quad (10)$$

where vector v is a uniform i.i.d. vector $v \sim \mathrm{uniform}[0, 1]^k$. This gives a two-step generative process for the distribution $p(z \mid b, \theta)$. On the first step we sample the maximum variable v_b from the Gumbel distribtuion and on the second step we sample the other variables $v_i, i \neq b$ from the Gumbel distribution trunctated at $\tilde{z}_b$ with location parameter θ_i.

The unbiased RELAX estimator from Grathwohl et al. (2018) is

$$\hat{g}_{\textit{RELAX}}(f) = [f(b) - c_\phi(\tilde{z})]\frac{\partial}{\partial\theta}\log p(b \mid \theta) + \frac{\partial}{\partial\theta}c_\phi(z) - \frac{\partial}{\partial\theta}c_\phi(\tilde{z}) \quad (11)$$

$$b = H(z),\ z \sim p(z \mid \theta),\ \tilde{z} \sim p(z \mid b, \theta) \quad (12)$$

where $c_\phi(z)$ is a parametric function optimized to reduce the variance of the estimator.

Similarly, for a differentiable function f the REBAR estimator by Tucker et al. (2017) uses the function f with the relaxed argument $\text{soft}\max(z)$ and tunes the scalar parameter η

$$\begin{aligned}\hat{g}_{\textit{REBAR}}(f) =&[f(b) - \eta f(\text{soft}\max(\tilde{z}))]\frac{\partial}{\partial\theta}\log p(b \mid \theta) \\ &+ \eta\frac{\partial}{\partial\theta}f(\text{soft}\max(z)) \\ &- \eta\frac{\partial}{\partial\theta}f(\text{soft}\max(\tilde{z}))\end{aligned} \tag{13}$$

$$b = H(z),\ z \sim p(z \mid \theta),\ \tilde{z} \sim p(z \mid b, \theta) \tag{14}$$

Constructing Control Variates for the Plackett-Luce Distribution

In this paper, we extend the stochastic gradient estimators $\hat{g}_{\textit{REBAR}}(f)$ and $\hat{g}_{\textit{RELAX}}(f)$ from the categorical distribution to the Plackett-Luce distribution. With a slight abuse of notation, below we use letter b to denote an integer vector $b = (b_1, \dots, b_k) \in S_k$ that represent a permutation, θ to denote the parameters of the Plackett-Luce distribution and $p(b \mid \theta)$ to denote the Plackett-Luce distribution.

The goal of this section is to define the two components required to apply the aforementioned gradient estimators: the mapping $b = H(z)$ and the two reparametrizable conditional distributions $p(z \mid \theta)$ and $p(z|b, \theta)$. After this we apply the estimators as defined in eq. 11 and eq. 13, but to emphasize the difference we refer to them as PL-RELAX and PL-REBAR.

Definition 1. *The Plackett-Luce distribution (Luce 2005; Plackett 1975) with scores $\theta = (\theta_1, \dots, \theta_k)$ is a distribution over permutations S_k with the probability of outcome $b \in S_k$*

$$p(b|\theta) = \prod_{j=1}^{k} \frac{\exp\theta_{b_j}}{\sum_{u=j}^{k}\exp\theta_{b_u}}. \tag{15}$$

Intuitively, a sample from the Plackett-Luce distribution $b = (b_1, \dots, b_k)$ is generated as a sequence of samples from categorical distributions. The first component b_1 comes from the categorical distribution with logits θ, then the second components b_2 comes from the categorical distribution with the logits θ without the component θ_{b_1} and so on.

The Plackett-Luce can be used for variational optimization (Staines and Barber 2012). Indeed, at the lower temperatures $\theta \to \frac{\theta}{T}, T \ll 1$ the distribution converges to a divergent distribution. The mode of the Plackett-Luce distribution is the descending order permutation of the scores $b^0 : \theta_{b_1^0} \geq \cdots \geq \theta_{b_k^0}$, because b^0 permutation maximizes each factor in the product in eq. 15.

Now we will give an alternative definition of the Plackett-Luce distribution.

Lemma 1. *(appears in (Grover et al. 2019; Yellott Jr 1977)) Let z be a vector of k independent Gumbel random variables with location parameters specified by score vector θ*

$$z_i = \theta_i - \log(-\log(v_i)),\ v_i \sim \text{uniform}[0, 1]. \tag{16}$$

Then for a permutation $b \in S_k$ the probability of event $\{z_{b_1} \geq \cdots \geq z_{b_k}\}$ is

$$p(z_{b_1} \geq \cdots \geq z_{b_k}) = \prod_{j=1}^{k} \frac{\exp\theta_{b_j}}{\sum_{u=j}^{k}\exp\theta_{b_u}}. \tag{17}$$

Similarly to the **Gumbel-max trick**, Lemma 1 shows that an order of a Gumbel-distributed vector is distributed according to the Plackett-Luce distribution. Following the lemma, for Plackett-Luce distributions we define $p(z \mid \theta)$ to be a Gumbel-distributed vector and $H(z)$ to be a sorting operation

$$z_i \sim \mathcal{G}(\theta_i, 1),\ i = 1, \dots, k \tag{18}$$

$$H(z) = \arg\text{sort}(z) \tag{19}$$

Our principal discovery is that, similarly to the categorical case, the conditional distribution $p(z|b, \theta)$ factorizes into a sequence of truncated Gumbel distributions. As a consequence, the distribution is reparametrizable and can be used to construct a control variate for a gradient estimator.

Proposition 1. *Let $p(b, z \mid \theta)$ be the joint distribution with $z_i \sim \mathcal{G}(\theta_i, 1)$, $b = \arg\text{sort}(z)$ and normalized parameters $\sum_{j=1}^{k}\exp\theta_j = 1$. Then for uniform i.i.d samples $v_i \sim \text{uniform}[0, 1]$ and $\Theta_i = \sum_{j=i}^{k}\exp\theta_{b_j}$ for $i = 1, \dots, k$ the vector $\tilde{z} = (\tilde{z}_1, \dots, \tilde{z}_k)$*

$$\tilde{z}_{b_i} = \begin{cases} -\log(-\log v_i) & i = 1 \\ -\log(-\frac{\log v_i}{\Theta_i} + \exp(-\tilde{z}_{b_{i-1}})) & i \geq 2, \end{cases} \tag{20}$$

is a sample from the conditional distriubtion $p(z \mid b, \theta)$.

The proof of the proposition is given in the appendix.

The sampling procedure from Proposition 1 has two principal differences from the sampling scheme for the categorical case (see eq. 10). First, the truncation parameter $\tilde{z}_{b_{i-1}}$ now depends on the previous component $i - 1$, while for the categorical case the truncation parameter is defined by the maximum component. Second, the location parameter is now a cumulative sum and depends on the previous scores.

Related Work

Jang, Gu, and Poole; Maddison, Mnih, and Teh (2016; 2016) use the Gumbel distribution and Gumbel-max trick to define continuous relaxations of discrete distributions, by providing a gradient estimator which replaces the sampling of a categorical distribution with a differentiable sample from a Gumbel-Softmax distribution.

The Gumbel-Softmax distribution does not scale to permutations, as distribution over k-dimensional permutations is equivalent to that over $k!$ categories. Recently, a line of work proposed various for optimization over permutations. Linderman et al. (2018) relaxes the discrete set of permutations to Birkhoff polytope, the set of doubly-stochastic matrices, and extend stick-breaking approach (Sethuraman

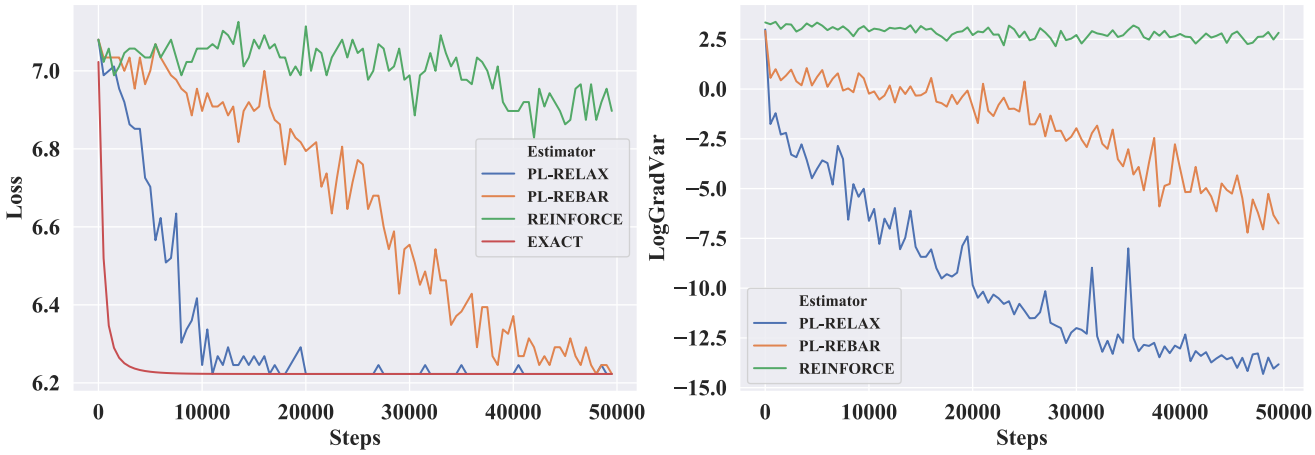


Figure 1: Training curves and log-variance of gradient estimators for different estimators on a toy problem: $\mathbb{E}_{p(b|\theta)}\|P_b - P_{0.05}\|_F^2$

1994) to satisfy polytope constraints. Mena et al. (2018) obtain doubly-stochastic matrices by applying the Sinkhorn operator. They use the Gumbel-Softmax distribution to define a distribution over latent matchings, the implicit Gumbel-Sinkhorn distribution. Grover et al. (2019) define new relaxation to the set of unimodal row-stochastic matrices, the set of matrices that have a unique maximal element in every row.

Grathwohl et al. (2018) extend Tucker et al. (2017) and derive control variate for black-box function optimization combining the REINFORCE estimator and reparametrization trick. Yin and Zhou (2018) propose gradient estimator that estimates the gradients of discrete distribution parameters in an augmented space.

For the special case of TSP, (Bello et al. 2016; Kool, van Hoof, and Welling 2018) introduce an amortized family of distributions over permutations using a deep autoregressive model and design control variates that exploit the structure of the loss function.

Experiments

We demonstrate the effectiveness of the proposed method with a simple toy task similar to Tucker et al. (2017) and then continue to the more challenging task of optimization over topological orderings for solving causal structure learning problems. Our PyTorch (Paszke et al. 2017) implementation of the gradient estimators is available at https://github.com/agadetsky/pytorch-pl-variance-reduction .

Toy Experiment

As a proof of concept we perform an experiment in minimizing $\mathbb{E}_{p(b|\theta)}\|P_b - P_t\|_F^2 = \mathbb{E}_{p(b|\theta)} f(P_b)$ as a function of θ where $p(b|\theta) = \text{Plackett-Luce}(b|\theta)$. P_b is permutation matrix with elements $p_{i,b_i} = 1$ and P_t is a matrix with $\frac{1}{k} + t$ on the main diagonal and $\frac{1}{k} - \frac{t}{k-1}$ in the remaining positions. This problem can be seen as linear sum assignment problem with specifically constructed doubly stochastic matrix P_t. It is easy to note that taking $k = 2$ and $t = 0.05$ leads to toy problem similar to that of Tucker et al. (2017). We focus on $t = 0.05$ and $k = 8$ to enable computation of exact gradients. For the PL-REBAR estimator we take $c_\phi(z) = \eta f(\sigma(z, \tau))$ where $\sigma(z, \tau)$ is the continuous relaxation of permutations described by Grover et al. (2019). For the PL-RELAX estimator we take $c_\phi(z) = f(\sigma(z, \tau)) + \rho_\phi(z)$ where $\rho_\phi(z)$ is a simple neural network with two linear layers and ReLU activation between them. Figure 1 shows the relative performance and gradient log-variance of REINFORCE, PL-REBAR and PL-RELAX. Although the REINFORCE estimator is unbiased, we can see that the variance of the estimator is too large even for the simple toy task, therefore the method is completely inapplicable for optimization over permutations. On the other hand, the proposed method significantly reduces variance of the gradient and thus converges to optimal. Also, similarly to the toy experiment from Grathwohl et al. (2018) paper, we observe better performance of the PL-RELAX estimator due to free-form control variate parameterized by a neural network.

Causal Structure Learning Through Order Search

Directed acyclic graph (DAG) models are popular tools for describing causal relationships and for guiding attempts to learn them from data. Learning the structure of a DAG remains challenging because of the combinatorial acyclicity constraint. A common way to model causal relations is a structural equation model (SEM). Let X be k-dimensional random variable, then relations are described as follows:

$$X_i = f_i(X_{pa(i)}, \varepsilon_i), \tag{21}$$

where $pa(i)$ is the set of parent vertices of variable X_i and ε_i is independent noise. Edge set $\{\cup_{i=1}^k \cup_{j \in pa(i)} j \to i\}$ describes DAG G on k vertices associated with joint distribution $\mathbb{P}_G(X) = \prod_{i=1}^k \mathbb{P}(X_i|pa(X_i))$. The basic structure learning problem can therefore be formulated as follows: let $\boldsymbol{X}$ be data matrix consisting of n i.i.d. samples of random

Table 1: Results for ER and SF graphs of 10 nodes

	ER1				ER4			
	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID
PL-RELAX	-0.2±1.7	5.2±2.5	5.8±3.2	13.0±9.6	12.2±26.3	29.4±1.9	35.0±4.4	67.0±1.8
SINKHORN$_{ECP}$	1.8±5.3	5.6±2.7	6.4±2.9	14.2±10.2	4.8±10.4	31.2±2.6	33.6±2.7	69.6±2.3
URS$_{ECP}$	13.5±26.9	7.4±3.7	7.4±3.6	16.0±8.9	12.4±6.2	29.8±3.6	32.8±4.9	67.4±2.1
SINKHORN	85.9±101.2	12.0±3.7	12.0±3.7	29.4±17.3	4019.6±3138.0	36.6±2.4	37.8±1.7	79.8±6.9
URS	71.4±128.9	10.8±2.9	11.0±3.2	26.0±10.5	1894.9±1704.8	34.6±2.2	36.8±2.8	74.4±2.7
GREEDY-SP	N/A	2.2±2.9	2.4±3.9	8.8±15.4	N/A	29.8±1.1	35.4±5.0	71.6±3.8
RANDOM	122.3±184.3	18.8±2.5	18.8±2.6	27.2±14.3	10078.2±10770.5	25.4±3.2	33.0±4.5	65.8±5.9
	SF1				SF4			
	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID
PL-RELAX	-0.7±0.3	2.2±1.5	2.4±1.5	2.6±2.2	-1.3±1.4	8.2±3.1	8.8±3.3	15.4±5.9
SINKHORN$_{ECP}$	0.6±2.9	2.8±3.2	3.0±3.2	7.6±12.3	-0.3±4.0	6.6±1.5	7.0±1.8	11.8±4.0
URS$_{ECP}$	1.6±1.8	5.0±1.7	5.4±2.2	7.0±2.2	2.1±2.3	12.8±2.5	13.4±2.2	24.8±5.6
SINKHORN	22.6±22.4	9.0±0.0	9.2±0.4	17.4±3.8	232.4±251.8	17.2±2.8	17.6±3.4	34.4±9.9
URS	10.1±5.2	9.6±1.2	9.6±1.2	14.6±2.1	69.6±81.6	14.6±1.4	14.6±1.2	29.2±5.8
GREEDY-SP	N/A	0.8±0.4	0.0±0.0	2.8±1.6	N/A	5.0±9.5	4.2±9.4	11.0±12.7
RANDOM	35.4±22.4	17.2±2.6	18.0±2.6	23.6±6.4	240.3±251.0	34.4±2.6	35.6±2.2	31.4±11.2

Table 2: Results for ER and SF graphs of 20 nodes

	ER1				ER4			
	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID
PL-RELAX	15.7±27.3	14.4±5.3	16.0±6.2	61.0±48.7	468.8±208.4	71.0±5.9	72.6±3.9	289.6±9.1
SINKHORN$_{ECP}$	10.4±8.7	15.8±4.7	17.0±6.0	84.8±56.3	2519.0±3715.2	78.0±6.1	78.8±5.5	302.2±15.8
URS$_{ECP}$	27.5±34.2	20.6±6.3	21.4±7.2	96.8±74.6	1011.4±745.5	75.8±2.9	76.6±2.9	300.2±20.3
SINKHORN	1651.2±3050.4	24.0±6.1	25.0±6.7	131.2±76.5	126284.6±194386.3	88.8±6.0	91.0±5.7	330.0±14.1
URS	1189.4±1815.5	26.4±8.4	26.6±8.6	134.2±75.0	7179677.6±7874489.3	93.0±3.8	94.4±4.5	328.0±11.5
GREEDY-SP	N/A	18.6±13.5	18.0±16.6	74.0±53.5	N/A	103.4±10.9	105.6±10.5	288.6±14.7
RANDOM	895.1±1270.3	37.8±5.2	38.8±4.9	146.8±79.9	109891.2±74968.7	113.0±4.9	114.4±4.1	330.6±9.2
	SF1				SF4			
	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID
PL-RELAX	-1.5±0.2	4.0±0.6	4.6±0.5	4.2±0.7	-5.8±1.2	20.0±4.3	20.0±4.1	48.4±16.2
SINKHORN$_{ECP}$	1.9±4.3	6.6±2.2	6.6±2.4	10.4±5.0	-0.4±2.4	25.6±5.6	25.8±5.9	58.6±19.7
URS$_{ECP}$	3.0±2.0	10.6±2.0	10.6±1.6	14.4±4.0	8.5±11.8	30.2±5.8	30.6±5.2	72.2±25.0
SINKHORN	38.3±26.2	19.0±0.0	19.0±0.0	35.0±2.4	158.2±99.9	44.6±5.8	44.8±6.1	103.6±20.8
URS	38.3±26.2	19.0±0.0	19.0±0.0	35.0±2.4	140.7±140.6	42.0±5.4	42.8±5.1	89.8±20.4
GREEDY-SP	N/A	2.0±1.4	0.0±0.0	7.0±5.1	N/A	50.6±31.5	49.8±32.3	69.0±43.2
RANDOM	94.0±36.4	36.2±2.6	36.6±2.3	48.6±14.7	635.5±182.6	98.2±6.1	99.2±5.5	168.8±29.6

Table 3: Results for ER and SF graphs of 50 nodes

	ER1				ER4			
	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID
PL-RELAX	-1.8±1.3	19.2±6.9	20.6±7.8	103.2±55.5	1863.1±1703.2	220.6±42.8	221.4±43.5	1779.6±193.1
SINKHORN$_{ECP}$	5.5±7.0	30.0±6.3	30.8±5.8	151.8±35.1	43463.9±70904.3	221.0±14.7	223.2±15.2	1846.4±158.3
URS$_{ECP}$	10.3±4.7	41.0±2.4	40.0±2.7	177.6±17.1	22997.9±38346.1	239.4±31.6	240.2±31.5	1789.8±154.4
SINKHORN	90.3±35.8	49.6±4.3	49.6±4.3	275.0±42.5	231304.8±290019.0	248.6±18.5	250.4±19.1	1966.8±135.5
URS	90.3±35.8	49.6±4.3	49.6±4.3	275.0±42.5	546793216.7±984510739.7	320.2±26.8	320.8±27.1	2119.0±130.5
GREEDY-SP	N/A	38.2±21.6	38.2±24.6	151.6±84.3	N/A	525.6±35.5	526.8±34.7	1951.4±50.3
RANDOM	271.0±71.6	99.4±9.3	99.8±9.5	301.2±60.4	477442.0±661243.9	360.8±23.5	361.0±23.2	2175.0±52.6
	SF1				SF4			
	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID
PL-RELAX	-3.9±0.5	11.4±3.3	11.8±2.9	14.4±2.7	-1.1±7.6	70.0±9.9	70.6±11.2	219.0±20.3
SINKHORN$_{ECP}$	25.1±18.2	28.6±6.5	28.4±6.1	58.4±12.1	124.3±126.0	94.4±22.7	95.6±23.0	257.2±25.8
URS$_{ECP}$	32.1±44.3	33.4±10.2	33.6±10.7	55.6±32.7	164.4±53.1	110.6±12.8	111.4±13.7	319.6±18.1
SINKHORN	138.2±68.2	49.0±0.0	49.0±0.0	110.6±5.5	10238.2±15850.1	139.0±8.3	139.6±8.1	387.0±37.2
URS	138.2±68.2	49.0±0.0	49.0±0.0	110.6±5.5	7966.9±4838.0	142.8±11.8	144.2±12.1	527.4±86.8
GREEDY-SP	N/A	38.8±39.3	35.4±39.6	54.8±20.6	N/A	381.2±76.2	384.2±77.0	963.0±475.7
RANDOM	380.1±207.8	97.8±7.3	97.8±7.3	155.4±31.2	10109.8±2027.0	312.0±14.9	312.4±15.0	807.0±101.7

variable X. Also let $\mathbb{D}$ be space of DAGs. Then, given observations $\boldsymbol{X}$ the task is to find DAG $G \in \mathbb{D}$ or so-called Bayesian Network for joint distribution $\mathbb{P}(X)$:

$$\min_{G \in \mathbb{D}} Q(G, \boldsymbol{X}) \tag{22}$$

where Q is function that scores DAG G given data.

To incorporate permutations in the objective (22) we consider parametrization of DAG adjacency matrix using nilpotent matrices which are upper triangular in basis induced by topological ordering, namely $W_G = PAP^T$ where A is strictly upper triangular adjacency matrix which describes parent sets of variables and permutation matrix P which describes topological ordering. Then optimization over DAGs can thus be seen as an optimization over topological orderings

$$\min_{P \in \mathcal{P}_k} \widehat{Q}(P, \boldsymbol{X}), \tag{23}$$

where $\widehat{Q}$ scores topological ordering P and $\mathcal{P}_k$ is the set of permutation matrices of size k. Optimization over A is usually hidden in the computation of $\widehat{Q}$. It is worth noting that this approach is similar to order MCMC (Friedman and Koller 2003), however our work considers gradient-based optimization over permutations matrices rather than discrete order changes.

Continuous data We consider linear additive noise SEMs:

$$X = W^T X + \varepsilon \tag{24}$$

where $W = PAP^T$ and non-zero elements of A describe linear coefficients and parent sets for each variable X_i.

As score function $\widehat{Q}$ we take regularized mean squared loss combined with sparsity-inducing $L1$ regularization term

$$\widehat{Q}(P, \boldsymbol{X}) = \min_{A \in \mathbb{A}} \frac{1}{2n} \|\boldsymbol{X} - PAP^T \boldsymbol{X}\|_F^2 + \lambda \|\operatorname{vec}(A)\|_1, \tag{25}$$

where $\mathbb{A}$ is the set of strictly upper triangular matrices. Computing $\widehat{Q}$ itself involves optimization problem, which can be efficiently solved using accelerated proximal gradient for convex composite function optimization (Nesterov 2013). To apply the proposed method, we reformulate (23) as variational optimization with respect to parameters of a Plackett-Luce distribution:

$$\min_{\theta} \mathop{\mathbb{E}}_{p(b|\theta)} \widehat{Q}(P_b, \boldsymbol{X}) \tag{26}$$

where $p(b|\theta) = \text{Plackett-Luce}(b|\theta)$, and P_b is a permutation matrix with $p_{i,b_i} = 1$. For variational optimization, we only apply PL-RELAX and treat $\widehat{Q}(P, \boldsymbol{X})$ as a black-box function to avoid unrolling the optimizer to compute gradients.

As a concurrent approach, we consider work by Mena et al. (2018) which proposes relaxing optimization over a set of permutations to a set of doubly-stochastic matrices using the Sinkhorn operator. Another recent work by Grover et al. (2019) proposes relaxation to the set of unimodal row-stochastic matrices (URS) which intersects the set of doubly-stochastic matrices and contains the set of all permutation matrices. Since these methods can't be used to optimize black-box functions we reformulate (26) as:

$$\min_{\phi} \min_{A \in \mathbb{A}} \frac{1}{2n} \|\boldsymbol{X} - P(\phi) A P(\phi)^T \boldsymbol{X}\|_F^2 + \lambda \|\operatorname{vec}(A)\|_1 \tag{27}$$

where ϕ are the parameters of the corresponding relaxation. We optimize (27) coordinate-wise using gradient descent with respect to ϕ and accelerated proximal gradient optimization with respect to A. We refer to the optimization of this objective as SINKHORN or URS according to the used relaxation.

We also try an alternative approach for the above relaxations. Since $P(\phi)$ is not a permutation matrix during training we extend (27) with an orthogonality constraint and replace $\|vec(A)\|$ with $H_\mu(vec(PAP^T))$ where H_μ is the Huber relaxation of $L1$ norm and μ is a hyperparameter controlling tightness of relaxation:

$$\begin{aligned} \min_{\phi} \min_{A \in \mathbb{A}} \quad & \frac{1}{2n} \|\boldsymbol{X} - P(\phi) A P(\phi)^T \boldsymbol{X}\|_F^2 + \\ & + \lambda H_\mu(\operatorname{vec}(P(\phi) A P^T(\phi))) \\ \text{s. t.} \quad & \|P(\phi) P^T(\phi) - I_k\|_F^2 = 0 \end{aligned} \tag{28}$$

We use an Augmented Lagrangian (Nemirovski 1999) to solve this equality constrained optimization problem (ECP) (28) and refer to the solutions as SINKHORN$_{ECP}$ or URS$_{ECP}$ correspondingly.

We simulated graphs from two well-known random graph models with different degree distributions: Erdos-Renyi random graphs and Scale-free networks with *k* and *4 k* expected number of edges, denoted by ER1, ER4, SF1, SF4 respectively. Given a random acyclic graph we assigned edge weights independently from $U([-2; -0.5] \cup [0.5; 2])$ to obtain weight matrix W. To generate data matrix $\boldsymbol{X}$ we follow generating process of linear SEM (24) with standard Gaussian noise.

As a sanity check, we also introduce a simple baseline. We generate Erdos-Renyi random graphs with the corresponding expected number of edges and refer to it as RANDOM baseline. For comparison, we also include the Greedy Sparse Permutation (Greedy-SP) algorithm (Solus et al. 2017). This algorithm casts DAG structure learning as a linear programming problem with graph sparsity as the linear objective function, and a sub-polytope of the permutohedron as the feasible region. Whilst this algorithm also searches permutations as a proxy to DAGs to reduce the size of the search space, it is in essence a constraint-based method - rather than optimising a DAG score function, it searches for the sparsest DAG which satisfies the conditional independence relations found. Conversely, our gradient-based method does not rely on these conditional independence tests, which typically require the simplifying assumptions of CI tests, and is able to use linear as well as non-linear objective functions (e.g. in the discrete data experiment, the quotient normalized maximum likelihood score is non-linear and non-differentiable).

For each method we report the score difference $\widehat{Q}$ (25) between learned and ground truth DAGs on additionally generated validation samples $\boldsymbol{X}_{val}$, as well as three DAG met-

rics from causal inference literature. The quoted score difference shows the effectiveness of our method for optimizing the chosen score function, while the DAG metrics show how well it performs on the problem itself. *Structural hamming distance* (SHD) is the number of edge additions, removals, and reversals required to get from the learned structure to the ground truth. Multiple DAGs can represent the same set of conditional independence relations, forming a Markov equivalence class; this can be represented by a completed partially directed acyclic graph (CPDAG). We also report SHD-CPDAG - the SHD between the CPDAG the learned structure belongs to and that of the true structure. *Structural interventional distance* (SID) (Peters and Bühlmann 2013) quantifies the distance between two DAGs in terms of their respective causal inference statements. This gives an indication of accuracy of computed interventions using the learned graph.

We consider graphs of 10, 20 and 50 nodes.

For PL-RELAX we take the mode of the distribution after training. For SINKHORN relaxation we apply the Hungarian algorithm to find the closest permutation matrix. For URS we use the argmax permutation property to obtain the permutation matrix. Regularization coefficient λ is set to 0.5 for all methods.

Tables 1-3 show the performance of all methods for varying number of nodes k averaged across 5 random seeds (the error ranges represent standard deviation). We can see that the proposed method outperforms baselines in the majority of settings. Also, it is worth mentioning that SINKHORN and URS perform poorly in terms of score function values due to the fact that the optimization is carried out over the set of relaxed matrices. This leads to deterioration in score value $\widehat{Q}$ when relaxation is transformed to permutation. As we can see there is no such problem with ECP versions of relaxations, though they perform worse than PL-RELAX and require additional constrained optimization techniques to be applied. Also, one more observation should be explained: PL-RELAX almost always ends up with better solutions in terms of score function than the ground truth DAG, therefore solves the optimization problem well. However, it is not ideal in terms of metrics. Peters and Bühlman (2014) proved that given enough data, it is possible to identify the ground truth DAG if data was generated from linear SEM with Gaussian homogeneous noise. Authors used $L0$-regularized mean squared error score function, but it is non-convex and hard to optimize, therefore $L1$-regularization is used in practice. Because of relaxation of the $L0$ norm and finite amount of data all guarantees vanish, and we observe inconsistency between the metrics of interest and values of the surrogate score function $\widehat{Q}$.

Discrete data Due to the discrete and nonlinear nature of categorical data, it cannot be modeled with the SEM defined previously. Discrete variable networks can however be modeled as generated by sampling each node's conditional probability table, depending only on the configuration of its parent nodes. In the standard general form this is $X_i = f_i(X_{pa(i)})$, where f_i is assumed to be multinomial, thus

$$f_i(X_{pa(i)}) \sim Multinomial(\Theta_{X_i}|Pa(X_i))$$

where $\Theta_{X_i}|Pa(X_i)$ are the conditional probabilities $\theta_{i,j,k} = P(X_i = k|Pa(X_i) = j)$.

Rather than learning the optimal A for a given P by minimising a training loss, we can therefore instead try to maximise the marginal likelihood based on the above model

$$Q(P, \mathbf{X}) = \max_{A \in \mathbb{A}} P(\mathbf{X}|A, P) \tag{29}$$

which can be found using the factorisation

$$P(\mathbf{X}|A, P) = \prod_{i=1}^{d} \prod_{j=1}^{q_i} P(\mathbf{X}_{i,pa(i)=j}; \alpha). \tag{30}$$

As a result of the decomposition of the score by node in equation (30), the *maximum a posteriori* (MAP) parent set can be selected from the set of parents permitted by the topological ordering for each node, independently of the rest. Due to the ordering, the graph resulting from combining each of these MAP parent connections is guaranteed to be acyclic, thus the exact MAP DAG for a given ordering can be found. Due to the combinatorial size of even this reduced search space, the set of permitted parents for a given node is reduced further, to only those that cannot be easily proven to be conditionally independent - as determined by a standard constraint-based method (in this case the PC-stable algorithm (Colombo and Maathuis 2014)). As this finds the exact solution for a reduced search space, the result is an approximation of the best score possible for the ordering. Whilst this provides an approximate score for any given order, it is a non-differentiable black-box function; therefore whilst our method can be applied to this permutation optimization, options are severely limited - the SINKHORN and URS methods used for continuous SEM graph benchmarks for example cannot be used. For a simple evaluation, Table 4 shows the result of our method on data sampled from the standard ALARM network compared against random orders, and permutations optimized by order MCMC (Friedman and Koller 2003), all using the same MAP DAG method described above, maximizing the quotient normalized maximum likelihood score (Silander et al. 2018). Higher Val $\widehat{Q} - \widehat{Q}^*$ is better, other metrics lower is better. Whilst Table 4 shows our algorithm to be less effective than MCMC for this task, the comparison is not particularly favorable - MCMC is performed directly on permutations, rather than attempting to learn the Plackett-Luce distribution over permutations - thus the MCMC simply attempts to find a good local minimum in the score space. To give a lower bound to performance, we also compare to the MAP DAGs of 1000 random permutations, computed in the same way as for MCMC and our algorithm, showing sampling the learned Plackett-Luce distribution gives permutations far better than random.

Conclusion

In this work we proposed a gradient-based optimization method, with unique capabilities for application to Plackett-Luce distributions over permutations. A proof of concept

Table 4: Results for ALARM graph (37 nodes)

	Val $\widehat{Q} - \widehat{Q}^*$	SHD	SHD-CPDAG	SID
PL-RELAX	-15645.2±3255.8	14.6±1.7	19.0±2.3	214.2±31.8
SINKHORN		N/A		
URS		N/A		
ORDER MCMC	-13404.7±2224.6	8.6±1.1	10.6±0.5	104.4±20.8
RANDOM	-75022.7±9647.7	25.8±3.7	30.0±3.9	478.8±70.8

experiment shows our method outperforms existing methods for differentiable objective functions, whilst also generalizing to non-differentiable black-box functions, and being applicable to permutation learning despite the factorial complexity. This allowed us to extend Plackett-Luce distribution based causal graphical model structure learning beyond the simple SEM based methods, to the more general case of DAGs of arbitrary variable types.

In future, our method could be combined with other standard scoring functions from Bayesian network literature - providing they decompose as described in equation (30) - for DAG structure learning of continuous data from different model types. Other potential applications include approximate inference for probabilistic models with latent permutations, routing problems and combinatorial problems for permutations.

Acknowledgements

This work was partly supported by Sberbank AI Lab and UK EPSRC project EP/P03442X/1. Kirill Struminsky proved proposition 1 and was supported by the Russian Science Foundation grant no. 19-71-30020. The authors thank NRU HSE for providing computational resources, NVIDIA for GPU donations, and Amazon for AWS Cloud Credits.

Appendix

We prove Proposition 1 in this section. We first discuss the properties of Gumbel distribution. Then we discuss the generative processes for the densities used for $p(z \mid b, \theta)$ in Eq. 20. Then we show that $p(b \mid z)p(z \mid \theta) = p(b \mid \theta)p(z \mid b, \theta)$ for the unconditional Gumbel density $p(z \mid \theta)$ and the Plackett-Luce distribution $p(b \mid \theta)$.

Density for the Gumbel distribution and the truncated Gumbel distribution

The density function of the Gumbel distribution with location parameter μ is

$$\phi_\mu(z) = \exp(-z + \mu)\exp(-\exp(-z + \mu)) \tag{31}$$

and the cumulative density function is

$$\Phi_\mu = \exp(-\exp(-z + \mu)). \tag{32}$$

Our derivation of the conditional distribution $p(b \mid z, \theta)$ relies on the additive property of the cumulative density function of the Gumbel distribution

$$\Phi_{\log(\exp\mu + \exp\nu)}(z) = \exp(-\exp(z)(\exp\mu + \exp\nu)) = \Phi_\mu(z)\Phi_\nu(z), \tag{33}$$

which we enfold in the following auxiliary claim.

Lemma 2. *For permutation $b \in S_k$, score vector $\theta \in \mathbb{R}^k$ and $i = 1, \dots, k$ and the argument vector $z \in \mathbb{R}^k$ we have*

$$\phi_{\theta_{b_i}}(z_{b_i})\Phi_{\log(\sum_{j=i+1}^k \exp\theta_{b_j})}(z_{b_i}) \tag{34}$$

$$= \frac{\exp\theta_{b_i}}{\sum_{j=i}^k \exp\theta_{b_j}}\phi_{\log(\sum_{j=i}^k \exp\theta_{b_j})}(z_{b_i}). \tag{35}$$

Proof. For brevity, we denote $\exp\theta_i$ as p_i. We then rewrite the density $\phi_{\log p_{b_i}}(z_{b_i})$ through the exponent $\exp(-z_{b_i} + \log p_{b_i})$ and c.d.f. $\Phi_{\log p_{b_i}}(z_{b_i})$ and apply the additive property in Eq. 38:

$$\phi_{\log p_{b_i}}(z_{b_i})\Phi_{\log(\sum_{j=i+1}^k p_{b_j})}(z_{b_i}) \tag{36}$$

$$= p_{b_i}\exp(-z_{b_i})\Phi_{\log p_{b_i}}(z_{b_i})\Phi_{\log(\sum_{j=i+1}^k p_{b_j})}(z_{b_i}) \tag{37}$$

$$= p_{b_i}\exp(-z_{b_i})\Phi_{\log(\sum_{j=i}^k p_{b_j})}(z_{b_i}) \tag{38}$$

$$= p_{b_i}\frac{\sum_{j=i}^k p_{b_j}}{\sum_{j=i}^k p_{b_j}}\exp(-z_{b_i})\Phi_{\log(\sum_{j=i}^k p_{b_j})}(z_{b_i}) \tag{39}$$

$$= \frac{p_{b_i}}{\sum_{j=1}^k p_{b_j}}\phi_{\log(\sum_{j=i}^k p_{b_j})}(z_{b_i}). \tag{40}$$

The last step collapses the exponent and the c.d.f. into the density function $\phi_{\log(\sum_{j=i}^k p_{b_j})}(z_{b_i})$. □

Finally, to define the density of conditional distribution $p(b \mid z, \theta)$ we define the density of the truncated Gumbel distribution $\phi_\mu^{z_0}(z) \propto \phi_\mu(z)I[z \le z_0]$:

$$\phi_\mu^{z_0}(z) = \frac{\phi_\mu(z)}{\Phi_\mu(z_0)}(z)I[z \le z_0], \tag{41}$$

where the superscript z_0 denotes the truncation parameter.

Reparametrization for the Gumbel distribution and the truncated Gumbel distribution

The reparametrization trick requires representing a draw from a distribution as a deterministic transformation of a fixed distribution sample and a distribution parameter. For a sample z from the Gumbel distribution $\mathcal{G}(\mu, 1)$ with location parameter μ the representation is

$$z = \mu - \log(-\log v),\ v \sim \text{uniform}[0, 1]. \tag{42}$$

For the Gumbel distribution truncated at z_0 (Maddison, Tarlow, and Minka 2014) proposed an analogous representation

$$z = \mu - \log(-\log v + \exp(-z_0 + \mu))$$

$$= -\log\left(-\frac{\log v}{\exp\mu} + \exp(-z_0)\right) \tag{43}$$

$$v \sim \text{uniform}[0, 1]. \tag{44}$$

In particular, the sampling schemes in Eq. 10 and Eq. 20 generate samples from the truncated Gumbel distribution.

The derivation of the conditional distribution

We now derive the conditional distribution and the sampling scheme defined in Proposition 1.

The joint distribution of the permutation b and the Gumbel samples z is

$$p(b, z \mid \theta) = p(b \mid z)p(z \mid \theta) \tag{45}$$

$$= \phi_{\theta_{b_1}}(z_{b_1}) \prod_{i=2}^{k} \left(\phi_{\theta_{b_i}}(z_{b_i}) I[z_{b_{i-1}} \geq z_{b_i}] \right) \tag{46}$$

We first multiply and divide the joint density by the c.d.f. $\Phi_{\log(\sum_{i=2}^{k} \exp \theta_{b_i})}(z_{b_1})$ and apply Lemma 2

$$\frac{\Phi_{\log(\sum_{i=2}^{k} \exp \theta_{b_i})}(z_{b_1})}{\Phi_{\log(\sum_{i=2}^{k} \exp \theta_{b_i})}(z_{b_1})} \phi_{\theta_{b_1}}(z_{b_1}) \prod_{i=2}^{k} \dots \tag{47}$$

$$= \frac{\exp \theta_{b_1}}{\sum_{i=1}^{k} \exp \theta_{b_i}} \frac{\phi_{\log(\sum_{i=1}^{k} \exp \theta_{b_i})}(z_{b_1})}{\Phi_{\log(\sum_{i=2}^{k} \exp \theta_{b_i})}(z_{b_1})} \prod_{i=2}^{k} \dots . \tag{48}$$

Next, we apply Lemma 2 to combine the c.d.f. in the denominator $\Phi_{\log(\sum_{j=i}^{k} \exp \theta_{b_j})}(z_{b_{i-1}})$ and the term $\phi_{\theta_{b_i}}(z_{b_i}) I[z_{b_{i-1}} \geq z_{b_i}]$ inside the product

$$\frac{\phi_{\theta_{b_i}}(z_{b_i}) I[z_{b_{i-1}} \geq z_{b_i}]}{\Phi_{\log(\sum_{j=i}^{k} \exp \theta_{b_j})}(z_{b_{i-1}})} \tag{49}$$

$$= \frac{\phi_{\theta_{b_i}}(z_{b_i}) I[z_{b_{i-1}} \geq z_{b_i}]}{\Phi_{\log(\sum_{j=i}^{k} \exp \theta_{b_j})}(z_{b_{i-1}})} \frac{\Phi_{\log(\sum_{j=i+1}^{k} \exp \theta_{b_j})}(z_{b_i})}{\Phi_{\log(\sum_{j=i+1}^{k} \exp \theta_{b_j})}(z_{b_i})} \tag{50}$$

$$= \frac{\exp \theta_{b_i}}{\sum_{j=i}^{k} \exp \theta_{b_j}} \frac{\phi^{z_{b_{i-1}}}_{\log(\sum_{j=i}^{k} \exp \theta_{b_j})}(z_{b_i})}{\Phi_{\log(\sum_{j=i+1}^{k} \exp \theta_{b_j})}(z_{b_i})} \tag{51}$$

and obtain the truncated distribution $\phi^{z_{b_{i-1}}}_{\log(\sum_{j=i}^{k} \exp \theta_{b_j})}(z_{b_i})$ along with one factor of the Plackett-Luce probability $\frac{\exp \theta_{b_i}}{\sum_{j=i}^{k} \exp \theta_{b_j}}$. Also, after the transformation the summation index in the denominator c.d.f. changes from i to $i+1$. This gives us an induction step that we apply sequentially for $i = 2, \dots, k-1$. For $i = k$ the denominator c.d.f. $\Phi_{\log \exp \theta_k}(z_{b_{k-1}})$ and the product term $\phi_{\log \exp \theta_k}(z_{b_k}) I[z_{k-1} \geq z_k]$ combine into the truncated Gumbel distribution with density $\phi^{z_{b_{k-1}}}_{\log \exp \theta_k}(z_{b_k})$.

As a result, we rearrange $p(b, z \mid \theta)$ into the product of the truncated Gumbel distribution densities $p(z \mid b, \theta)$ and the probability of the Plackett-Luce distribution $p(b \mid \theta)$:

$$\prod_{i=1}^{k} \frac{\exp \theta_{b_i}}{\sum_{j=i}^{k} \exp \theta_{b_j}} \left(\phi_0(z_{b_1}) \prod_{i=2}^{k} \phi^{z_{b_{i-1}}}_{\log \sum_{j=i}^{k} \exp \theta_j}(z_{b_i}) \right). \tag{52}$$

Finally, to obtain the claim of Proposition 1 we apply the reparametrized sampling scheme defined in Eq. 43.

References

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.

Colombo, D., and Maathuis, M. H. 2014. Order-independent constraint-based causal structure learning. *The Journal of Machine Learning Research* 15(1):3741–3782.

Friedman, N., and Koller, D. 2003. Being bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine learning* 50(1-2):95–125.

Grathwohl, W.; Choi, D.; Wu, Y.; Roeder, G.; and Duvenaud, D. 2018. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*.

Grover, A.; Wang, E.; Zweig, A.; and Ermon, S. 2019. Stochastic optimization of sorting networks via continuous relaxations. In *International Conference on Learning Representations*.

Guiver, J., and Snelson, E. 2009. Bayesian inference for plackett-luce ranking models. In *proceedings of the 26th annual international conference on machine learning*, 377–384. ACM.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kool, W.; van Hoof, H.; and Welling, M. 2018. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*.

Linderman, S.; Mena, G.; Cooper, H.; Paninski, L.; and Cunningham, J. 2018. Reparameterizing the birkhoff polytope for variational permutation inference. In Storkey, A., and Perez-Cruz, F., eds., *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, 1618–1627. Playa Blanca, Lanzarote, Canary Islands: PMLR.

Luce, R. D. 2005. *Individual Choice Behavior: A Theoretical Analysis*. Courier Corporation.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Maddison, C. J.; Tarlow, D.; and Minka, T. 2014. A* sampling. In *Advances in Neural Information Processing Systems*, 3086–3094.

Mena, G.; Belanger, D.; Linderman, S.; and Snoek, J. 2018. Learning latent permutations with gumbel-sinkhorn networks. In *International Conference on Learning Representations*.

Mnih, A., and Gregor, K. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.

Mohamed, S.; Rosca, M.; Figurnov, M.; and Mnih, A. 2019. Monte carlo gradient estimation in machine learning. *arXiv preprint arXiv:1906.10652*.

Neal, R. M., and Hinton, G. E. 1998. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*. Springer. 355–368.

Nemirovski, A. 1999. Optimization ii: Standard numerical methods for nonlinear continuous optimization. *Lecture notes*.

Nesterov, Y. 2013. Gradient methods for minimizing composite functions. *Mathematical Programming* 140(1):125–161.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*.

Peters, J., and Bühlman, P. 2014. Identifiability of Gaussian structural equation models with equal error variances. *Biometrika* 101(1):219–228.

Peters, J., and Bühlmann, P. 2013. Structural intervention distance (sid) for evaluating causal graphs. *arXiv preprint arXiv:1306.1043*.

Plackett, R. L. 1975. The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 24(2):193–202.

Rezende, D. J.; Mohamed, S.; and Wierstra, D. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.

Sethuraman, J. 1994. A constructive definition of dirichlet priors. *Statistica sinica* 639–650.

Silander, T.; Leppä-aho, J.; Jääsaari, E.; and Roos, T. 2018. Quotient normalized maximum likelihood criterion for learning bayesian network structures. In Storkey, A., and Perez-Cruz, F., eds., *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, 948–957. Playa Blanca, Lanzarote, Canary Islands: PMLR.

Solus, L.; Wang, Y.; Matejovicova, L.; and Uhler, C. 2017. Consistency guarantees for permutation-based causal inference algorithms.

Staines, J., and Barber, D. 2012. Variational optimization. *arXiv preprint arXiv:1212.4507*.

Tucker, G.; Mnih, A.; Maddison, C. J.; Lawson, J.; and Sohl-Dickstein, J. 2017. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 2627–2636.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Yellott Jr, J. I. 1977. The relationship between luce's choice axiom, thurstone's theory of comparative judgment, and the double exponential distribution. *Journal of Mathematical Psychology* 15(2):109–144.

Yin, M., and Zhou, M. 2018. Arm: Augment-reinforce-merge gradient for discrete latent variable models. *arXiv preprint arXiv:1807.11143*.

Appendix B Article. Quantifying learning guarantees for convex but inconsistent surrogates

Authors. Kirill Struminsky, Simon Lacoste-Julien, Anton Osokin
Published in: Advances in Neural Information Processing Systems. – 2018. – C. 669-677
Abstract. We study consistency properties of machine learning methods based on minimizing convex surrogates. We extend the recent framework of Osokin et al.(2017) for the quantitative analysis of consistency properties to the case of inconsistent surrogates. Our key technical contribution consists in a new lower bound on the calibration function for the quadratic surrogate, which is non-trivial (not always zero) for inconsistent cases. The new bound allows to quantify the level of inconsistency of the setting and shows how learning with inconsistent surrogates can have guarantees on sample complexity and optimization difficulty. We apply our theory to two concrete cases: multi-class classification with the tree-structured loss and ranking with the mean average precision loss. The results show the approximation-computation trade-offs caused by inconsistent surrogates and their potential benefits.

Quantifying Learning Guarantees for Convex but Inconsistent Surrogates

Kirill Struminsky
NRU HSE,* Moscow, Russia

Simon Lacoste-Julien†
MILA and DIRO
Université de Montréal, Canada

Anton Osokin
NRU HSE,*‡ Moscow, Russia
Skoltech,§ Moscow, Russia

Abstract

We study consistency properties of machine learning methods based on minimizing convex surrogates. We extend the recent framework of Osokin et al. [14] for the quantitative analysis of consistency properties to the case of inconsistent surrogates. Our key technical contribution consists in a new lower bound on the calibration function for the quadratic surrogate, which is non-trivial (not always zero) for inconsistent cases. The new bound allows to quantify the level of inconsistency of the setting and shows how learning with inconsistent surrogates can have guarantees on sample complexity and optimization difficulty. We apply our theory to two concrete cases: multi-class classification with the tree-structured loss and ranking with the mean average precision loss. The results show the approximation-computation trade-offs caused by inconsistent surrogates and their potential benefits.

1 Introduction

Consistency is a desirable property of any statistical estimator, which informally means that in the limit of infinite data, the estimator converges to the correct quantity. In the context of machine learning algorithms based on surrogate loss minimization, we usually use the notion of Fisher consistency, which means that the exact minimization of the expected surrogate loss leads to the exact minimization of the actual task loss. It can be shown that Fisher consistency is closely related to the question of infinite-sample consistency (a.k.a. classification calibration) of the surrogate loss with respect to the task loss (see [2, 17] for a detailed review).

The property of infinite-sample consistency (which we will refer to as simply consistency) shows that the minimization of a particular surrogate is the right problem to solve, but it becomes especially attractive when one can actually minimize the surrogate, which is the case, e.g, when the surrogate is convex. Consistency of convex surrogates has been the central question of many studies for such problems as binary classification [2, 24, 19], multi-class classification [23, 21, 1, 17], ranking [11, 4, 5, 18, 15] and, more recently, structured prediction [7, 14].

Recently, Osokin et al. [14] have pinpointed that in some cases minimizing a consistent convex surrogate might be not sufficient for efficient learning. In particular, when the number of possible predictions is large (which is typically the case in the settings of structured prediction and ranking) reaching adequately small value of the expected task loss can be practically impossible, because one would need to optimize the surrogate to high accuracy, which requires an intractable number of iterations of the optimization algorithm.

*National Research University Higher School of Economics
†CIFAR Fellow
‡Samsung-HSE Joint Lab
§Skolkovo Institute of Science and Technology

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

It also turns out [14] that the possibility of efficient learning is related to the structure of the task loss. The 0-1 loss, which does not make distinction between different kinds of errors, shows the worst case behavior. However, more structured losses, e.g., the Hamming distance between sequence labelings, allow efficient learning if the score vector is designed appropriately (for the Hamming distance, the score for a complete configuration should be decomposable into the sum of scores for individual elements).

However, the analysis of Osokin et al. [14] gives non-trivial conclusions only for consistent surrogates. At the same time it is known that inconsistent surrogates often work well in practice (for example, the Crammer-Singer formulation of multi-class SVM [8], or its generalization structured SVM [20, 22]). There have indeed been several works to analyze inconsistent surrogates [12, 18, 5, 14], but they usually end the story with proving that some surrogate (or a family or surrogates) is not consistent.

Contributions. In this work, we look at the problem from a more quantitative angle and analyze to which extent inconsistent surrogates can be useful for learning. We focus on the same setting as [14] and generalize their results to the case of inconsistent surrogates (their bounds are trivial for these cases) to be able to draw non-trivial conclusions. The main technical contribution consists in a tighter lower bound on the calibration function (Theorem 3), which is strictly more general than the bound of [14]. Notably, our bound is non-trivial in the case when the surrogate is not consistent and quantifies to which degree learning with inconsistent surrogates is possible. We further study the behavior of our bound in two practical scenarios: multi-class classification with a tree-structured loss and ranking with the mean average precision (mAP) loss. For the tree-structured loss, our bound shows that there can be a trade-off between the best achievable accuracy and the speed of convergence. For the mAP loss, we use our tools to study the (non-)existence of consistent convex surrogates of a particular dimension (an important issue for the task of ranking [11, 4, 5, 18, 17]) and quantify to which extent our quadratic surrogate with the score vector of insufficient dimension is consistent.

This paper is organized as follows. First, we introduce the setting we work with in Section 2 and review the key results of [14] in Section 3. In Section 4, we prove our main theoretical result, which is a new lower bound on the calibration function. In Section 5, we analyze the behavior of our bound for the two different settings: multi-class classification and ranking (the mean average precision loss). Finally, we review the related works and conclude in Section 6.

2 Notation and Preliminaries

In this section, we introduce our setting, which closely follows [14]. We denote the input features by $\boldsymbol{x} \in \mathcal{X}$ where $\mathcal{X}$ is the input domain. The particular structure of $\mathcal{X}$ is not of the key importance for this study. The output variables, that are in the center of our analysis, will be denoted by $\hat{\boldsymbol{y}} \in \hat{\mathcal{Y}}$ with $\hat{\mathcal{Y}}$ being the set of possible predictions or the output domain.[5] In such settings as structured prediction or ranking, the predictions are very high-dimensional and with some structure that is useful to model explicitly (for example, a sequence, permutation or image).

The central object of our study is the *loss function* $L(\hat{\boldsymbol{y}}, \boldsymbol{y}) \geq 0$ that represents the cost of making the prediction $\hat{\boldsymbol{y}} \in \hat{\mathcal{Y}}$ when the ground-truth label is $\boldsymbol{y} \in \mathcal{Y}$. Note that in some applications of interest the sets $\hat{\mathcal{Y}}$ and $\mathcal{Y}$ are different. For example, in ranking with the mean average precision (mAP) loss function (see Section 5.2 and, e.g., [18] for the details), the set $\hat{\mathcal{Y}}$ consists of all the permutations of the items (to represent the ranking itself), but the set $\mathcal{Y}$ consists of all the subsets of items (to represent the set of relevant items, which is the ground-truth annotation in this setting). In this paper, we only study the case when both $\hat{\mathcal{Y}}$ and $\mathcal{Y}$ are finite. We denote the cardinality of $\hat{\mathcal{Y}}$ by k, and the cardinality of $\mathcal{Y}$ by m. In this case, the loss function can be encoded as a matrix L of size $k \times m$.

In many applications of interest, both quantities k and m are exponentially large in the size of the natural dimension of the input $\boldsymbol{x}$. For example, in the task of sequence labeling, both k and m are equal to the number of all possible sequences of symbols from a finite alphabet. In the task of ranking (the mAP formulation), k is equal to the number of permutations of items and m is equal to the number of item subsets.

[5] The output domain $\hat{\mathcal{Y}}$ itself can depend on the vector of input features $\boldsymbol{x}$ (for example, if $\boldsymbol{x}$ can represent sequences of different lengths and the length of the output sequence has to equal the length of the input), but we will not use this dependency and omit it for brevity.

Following usual practices, we work with the prediction model defined by a (learned) vector-valued *score function* $\mathfrak{f} : \mathcal{X} \to \mathbb{R}^k$, which defines a scalar score $\mathfrak{f}_{\hat{\boldsymbol{y}}}(\boldsymbol{x})$ for each possible output $\hat{\boldsymbol{y}} \in \hat{\mathcal{Y}}$. The final prediction is then chosen as an output configuration with the maximal score:

$$\operatorname{pred}(\mathfrak{f}(\boldsymbol{x})) := \operatorname*{argmax}_{\hat{\boldsymbol{y}} \in \hat{\mathcal{Y}}} \mathfrak{f}_{\hat{\boldsymbol{y}}}(\boldsymbol{x}). \tag{1}$$

If the maximal score is given by multiple outputs $\hat{\boldsymbol{y}}$ (so-called *ties*), the predictor follows a simple deterministic tie-breaking rule and picks the output appearing first in some predefined ordering on $\hat{\mathcal{Y}}$.

In this setup, learning consists in finding a *score function* $\mathfrak{f}$ for which the predictor gives the smallest expected loss with features $\boldsymbol{x}$ and labels $\boldsymbol{y}$ coming from an unknown data-generating distribution $\mathcal{D}$:

$$\mathcal{R}_L(\mathfrak{f}) := \mathbf{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}\, L\big(\operatorname{pred}(\mathfrak{f}(\boldsymbol{x})), \boldsymbol{y}\big). \tag{2}$$

The quantity $\mathcal{R}_L(\mathfrak{f})$ is usually referred to as the actual (or population) *risk* based on the loss L. Minimizing the actual risk directly is usually difficult (because of non-convexity and non-continuity of the predictor (1)). The standard approach is to substitute (2) with another objective, a *surrogate risk* (or the Φ-risk), which is easier for optimization (in this paper, we only consider convex surrogates):

$$\mathcal{R}_\Phi(\mathfrak{f}) := \mathbf{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}\, \Phi(\mathfrak{f}(\boldsymbol{x}), \boldsymbol{y}), \tag{3}$$

where we will refer to the function $\Phi : \mathbb{R}^k \times \mathcal{Y} \to \mathbb{R}$ as the *surrogate loss*. To make the minimization of (3) well-defined, we will always assume the surrogate loss Φ to be bounded from below and continuous.

The surrogate loss should be chosen in such a way that the minimization of (3) also leads to the minimization of (2), i.e., to the solution of the original problem. The property of consistency of the surrogate loss is an approach to formalize this intuition, i.e., to guarantee that no matter the data-generating distribution, minimizing (3) w.r.t. $\mathfrak{f}$ implies minimizing (2) w.r.t. $\mathfrak{f}$ as well (both of these are possible only in the limit of infinite data and computational budget). Osokin et al. [14] quantified what happens if the surrogate risk is minimized approximately by translating the optimization error of (3) to the optimization error of (2). The main goal of this paper is to generalize this analysis to the cases when the surrogate is not consistent and to show that there can be trade-offs between the minimum value of the actual risk that can be achieved by minimizing an inconsistent surrogate and the speed with which this minimum can be achieved.

3 Calibration Functions and Consistency

In this section, we review the approach of Osokin et al. [14] for studying consistency in the context of structured prediction. The first part of the analysis establishes the connection between the minimization of the actual risk $\mathcal{R}_L$ (2) and the surrogate risk $\mathcal{R}_\Phi$ (3) via the so-called *calibration function* (see Definition 1 [14, and references therein]). This step is usually called *non-parametric* (or pointwise) because it does not explicitly model the dependency of the scores $\boldsymbol{f} := \mathfrak{f}(\boldsymbol{x})$ on the input variables $\boldsymbol{x}$. The second part of the analysis establishes the connection with an optimization algorithm allowing to make a statement about how many iterations would be enough to find a predictor that is (in expectation) within ε of the global minimum of the actual risk $\mathcal{R}_L$.

Non-parametric analysis. The standard non-parametric setting considers all measurable score functions $\mathfrak{f}$ to effectively ignore the dependency on the features $\boldsymbol{x}$. As noted by [14], it is beneficial to consider a restricted set of the score functions $\mathfrak{F}_\mathcal{F}$ that consists of all vector-valued Borel measurable functions $\mathfrak{f} : \mathcal{X} \to \mathcal{F}$ where $\mathcal{F} \subseteq \mathbb{R}^k$ is a subspace of allowed score vectors. Compatibility of the subspace $\mathcal{F}$ and the loss function L will be a crucial point of this paper. Note that the analysis is still non-parametric because the dependence on $\boldsymbol{x}$ is not explicitly modeled.

Within the analysis, we will use the *conditional* actual and surrogate risks defined as the expectations of the corresponding losses w.r.t. a categorical distribution $\boldsymbol{q}$ on the set of annotations $\mathcal{Y}$, $m := |\mathcal{Y}|$:

$$\ell(\boldsymbol{f}, \boldsymbol{q}) := \sum\nolimits_{\boldsymbol{y}=1}^{m} q_{\boldsymbol{y}} L(\operatorname{pred}(\boldsymbol{f}), \boldsymbol{y}), \quad \phi(\boldsymbol{f}, \boldsymbol{q}) := \sum\nolimits_{\boldsymbol{y}=1}^{m} q_{\boldsymbol{y}} \Phi(\boldsymbol{f}, \boldsymbol{y}). \tag{4}$$

Hereinafter, we represent an m-dimensional categorical distribution $\boldsymbol{q}$ as a point in the probability simplex Δ_m and use the symbol $q_{\boldsymbol{y}}$ to denote the probability of the $\boldsymbol{y}$-th outcome. Using this notation, we can rewrite the risk $\mathcal{R}_L$ and surrogate risk $\mathcal{R}_\Phi$ as

$$\mathcal{R}_L(\mathfrak{f}) = \mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}_\mathcal{X}}\, \ell(\mathfrak{f}(\boldsymbol{x}), \mathbf{P}_\mathcal{D}(\cdot \mid \boldsymbol{x})), \quad \mathcal{R}_\Phi(\mathfrak{f}) = \mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}_\mathcal{X}}\, \phi(\mathfrak{f}(\boldsymbol{x}), \mathbf{P}_\mathcal{D}(\cdot \mid \boldsymbol{x})), \tag{5}$$

where $\mathcal{D}_{\mathcal{X}}$ is the marginal distribution of $\boldsymbol{x}$ and $\mathbf{P}_{\mathcal{D}}(\cdot \mid \boldsymbol{x})$ denotes the conditional distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ (both defined for the joint data-generating distribution $\mathcal{D}$).

For each score vector $\boldsymbol{f} \in \mathcal{F}$ and a distribution $\boldsymbol{q} \in \Delta_m$ over ground-truth labels, we now define the *excess* actual and surrogate risks

$$\delta\phi(\boldsymbol{f}, \boldsymbol{q}) = \phi(\boldsymbol{f}, \boldsymbol{q}) - \inf_{\hat{\boldsymbol{f}} \in \mathcal{F}} \phi(\hat{\boldsymbol{f}}, \boldsymbol{q}), \quad \delta\ell(\boldsymbol{f}, \boldsymbol{q}) = \ell(\boldsymbol{f}, \boldsymbol{q}) - \inf_{\hat{\boldsymbol{f}} \in \mathbb{R}^k} \ell(\hat{\boldsymbol{f}}, \boldsymbol{q}), \tag{6}$$

which show how close the current conditional actual and surrogate risks are to the corresponding minimal achievable conditional risks (depending only on the distribution $\boldsymbol{q}$). Note that the two infima in (6) are defined w.r.t. different sets of score vectors. For the surrogate risk, the infimum is taken w.r.t. the set of allowed scores $\mathcal{F}$ capturing only the scores obtainable by the learning process. For the actual risk, the infimum is taken w.r.t. the set of all possible scores $\mathbb{R}^k$ including score vectors that cannot be learned. This distinction is important when analyzing inconsistent surrogates and allows to characterize the *approximation error* of the selected function class.[6]

We are now ready to define the *calibration function*, which is the final object of the non-parametric part of the analysis. Calibration functions directly show how well one needs to minimize the surrogate risk to guarantee that the excess of the actual risk is smaller than ε.

Definition 1 (Calibration function, [14])**.** *For a task loss L, a surrogate loss Φ, a set of feasible scores $\mathcal{F}$, the* calibration function $H_{\Phi,L,\mathcal{F}}(\varepsilon)$ *is defined as:*

$$H_{\Phi,L,\mathcal{F}}(\varepsilon) := \inf_{\boldsymbol{f} \in \mathcal{F},\ \boldsymbol{q} \in \Delta_m} \delta\phi(\boldsymbol{f}, \boldsymbol{q}) \tag{7}$$

$$\text{s.t.} \quad \delta\ell(\boldsymbol{f}, \boldsymbol{q}) \geq \varepsilon, \tag{8}$$

where $\varepsilon \geq 0$ is the target accuracy. We set $H_{\Phi,L,\mathcal{F}}(\varepsilon)$ to $+\infty$ when the feasible set is empty.

By construction, $H_{\Phi,L,\mathcal{F}}$ is non-decreasing on $[0, +\infty)$, $H_{\Phi,L,\mathcal{F}}(\varepsilon) \geq 0$ and $H_{\Phi,L,\mathcal{F}}(0) = 0$. The calibration function also provides the so-called *excess risk bound*

$$H_{\Phi,L,\mathcal{F}}(\delta\ell(\boldsymbol{f}, \boldsymbol{q})) \leq \delta\phi(\boldsymbol{f}, \boldsymbol{q}),\ \forall \boldsymbol{f} \in \mathcal{F},\ \forall \boldsymbol{q} \in \Delta_m, \tag{9}$$

which implies the formal connection between the surrogate and task risks [14, Theorem 2].

The calibration function can fully characterize consistency of the setting defined by the surrogate loss, the subspace of scores and the task loss. The maximal value of ε at which the calibration function $H_{\Phi,L,\mathcal{F}}(\varepsilon)$ equals zero shows the best accuracy on the actual loss that can be obtained [14, Theorem 6]. The notion of level-η consistency captures this effect.

Definition 2 (level-η consistency, [14])**.** *A surrogate loss Φ is* consistent up to level $\eta \geq 0$ *w.r.t. a task loss L and a set of scores $\mathcal{F}$ if and only if the calibration function satisfies $H_{\Phi,L,\mathcal{F}}(\varepsilon) > 0$ for all $\varepsilon > \eta$ and there exists $\hat{\varepsilon} > \eta$ such that $H_{\Phi,L,\mathcal{F}}(\hat{\varepsilon})$ is finite.*

The case of level-0 consistency corresponds to the classical consistent surrogate and Fisher consistency. When $\eta > 0$, the surrogate is not consistent, meaning that the actual risk cannot be minimized globally. However, Osokin et al. [14, Appendix E.4] give an example where even though constructing a consistent setting is possible (by the choice of the score subspace $\mathcal{F}$), it might still be beneficial to use only a level-η consistent setting because of the exponentially faster growth of the calibration function. The main contribution of this paper is a lower bound on the calibration function (Theorem 3), which is non-zero for $\eta > 0$ and thus can be used to obtain convergence rates in inconsistent settings.

Optimization and learning guarantees; normalizing the calibration function. Osokin et al. [14] note that the scale of the calibration function is not defined, i.e., if one multiplies the surrogate loss by some positive constant, the calibration function is multiplied by the same constant as well. One way to define a "natural normalization" is to use a scale-invariant convergence rate of a stochastic optimization algorithm. Osokin et al. [14, Section 3.3] applied the classical online ASGD [13] (under the well-specification assumption) and got the sample complexity (and the convergence rate of ASGD at the same time) result saying that N^* steps of ASGD are sufficient to get ε-accuracy on the *task loss* (in expectation), where N^* is computed as follows:

$$N^* := \frac{4D^2M^2}{\check{H}^2_{\Phi,L,\mathcal{F}}(\varepsilon)}. \tag{10}$$

[6] Note that Osokin et al. [14] define the excess risks by taking both infima w.r.t. the the set of allowed scores $\mathcal{F}$, which is subtly different from us. The results of the two setups are equivalent in the cases of consistent surrogates, which are the main focus of Osokin et al. [14], but can be different in inconsistent cases.

Here the quantity N^* depends on a convex lower bound $\check{H}_{\Phi,L,\mathcal{F}}(\varepsilon)$ on the calibration function $H_{\Phi,L,\mathcal{F}}(\varepsilon)$ and the constants D, M, which appear in the convergence rate of ASGD: D is an upper bound on the norm of an optimal solution and M^2 is an upper bound on the expected square norm of the stochastic gradient. Osokin et al. [14] show how to bound the constant DM for a very specific quadratic surrogate defined below (see Section 3.1).

3.1 Bounds for the Quadratic Surrogate

The major complication in applying and interpreting the theoretical results presented in Section 3 is the complexity of computing the calibration function. Osokin et al. [14] analyzed the calibration function only for the quadratic surrogate

$$\Phi_{\text{quad}}(\boldsymbol{f}, \hat{\boldsymbol{y}}) := \tfrac{1}{2k}\|\boldsymbol{f} + L(:, \boldsymbol{y})\|_2^2 = \tfrac{1}{2k}\sum\nolimits_{\hat{\boldsymbol{y}}\in\hat{\mathcal{Y}}}(f_{\hat{\boldsymbol{y}}}^2 + 2f_{\hat{\boldsymbol{y}}}L(\hat{\boldsymbol{y}}, \boldsymbol{y}) + L(\hat{\boldsymbol{y}}, \boldsymbol{y})^2). \tag{11}$$

For any task loss L, this surrogate is consistent whenever the subspace of allowed scores is rich enough, i.e., the subspace of scores $\mathcal{F}$ fully contains $\operatorname{span}(L)$. To connect with optimization, we assume a parametrization of the subspace $\mathcal{F}$ as a span of the columns of some matrix F, i.e., $\mathcal{F} = \operatorname{span}(F) = \{\boldsymbol{f} = F\boldsymbol{\theta} \mid \boldsymbol{\theta} \in \mathbb{R}^r\}$.[7] In the interesting settings, the dimension r is much smaller than both k and m. Note that to compute the gradient of the objective (11) w.r.t. the parameters $\boldsymbol{\theta}$, one needs to compute matrix products $F^\mathsf{T}F \in \mathbb{R}^{r\times r}$ and $F^\mathsf{T}L(:, \boldsymbol{y}) \in \mathbb{R}^r$, which are usually both of feasible sizes, but require exponentially big sum (k summands) inside. Computing these quantities can be seen as some form of inference required to run the learning process.

Osokin et al. [14] proved a lower bound on the calibration functions for the quadratic surrogate (11) [14, Theorem 7], which we now present to contrast our result presented in Section 4. When the subspace of scores $\mathcal{F}$ contains $\operatorname{span}(L)$, $\operatorname{span}(L) \subseteq \mathcal{F}$, implying that the setting is consistent, the calibration function is bounded from below by $\min_{i\neq j} \frac{\varepsilon^2}{2k\|P_\mathcal{F}\Delta_{ij}\|_2^2}$, where $P_\mathcal{F}$ is the orthogonal projection on the subspace $\mathcal{F}$ and $\Delta_{ij} := \mathbf{e}_i - \mathbf{e}_j \in \mathbb{R}^k$ with $\mathbf{e}_c$ being the c-th basis vector of the standard basis in $\mathbb{R}^k$. They also showed that for some very structured losses (Hamming and block 0-1 losses), the quantity $k\|P_\mathcal{F}\Delta_{ij}\|_2^2$ is not exponentially large and thus the calibration function suggests that efficient learning is possible. One interesting case not studied by Osokin et al. [14] is the situation where the subspace of scores $\mathcal{F}$ does not fully contain the subspace $\operatorname{span}(L)$. In this case, the surrogate might not be consistent but still lead to effective and efficient practical algorithms.

Normalizing the calibration function. The normalization constant DM appearing in (10) can also be computed for the quadratic surrogate (11) under the assumption of well-specification (see [14, Appendix F] for details). In particular, we have $DM = L_{\max}^2\xi(\kappa(F)\sqrt{r}RQ_{\max})$, $\xi(z) = z^2 + z$, where $L_{\max}$ denotes the maximal value of all elements in L, $\kappa(F)$ is the condition number of the matrix F and r in an upper bound on the rank of $\mathcal{F}$. The constants R and $Q_{\max}$ come from the kernel ASGD setup and, importantly, depend only on the data distribution, but not on the loss L or score matrix F. Note that for a given subspace $\mathcal{F}$, the choice of matrix F is arbitrary and it can always be chosen as an orthonormal basis of $\mathcal{F}$ giving a $\kappa(F)$ of one. However, such F can lead to inefficient prediction (1), which makes the whole framework less appealing. Another important observation coming from the value of DM is the justification of the $\frac{1}{k}$ scaling in front of the surrogate (11).

4 Calibration Function for Inconsistent Surrogates

Our main result generalizes the Theorem 7 of [14] to the case of inconsistent surrogates (the key difference consists in the absence of the assumption $\operatorname{span}(L) \subseteq \mathcal{F}$).

Theorem 3 (Lower bound on the calibration function $H_{\Phi_{\text{quad}},L,\mathcal{F}}(\varepsilon)$)**.** *For any task loss L, its quadratic surrogate Φ_{quad}, and a score subspace $\mathcal{F}$, the calibration function is bounded from below:*

$$H_{\Phi_{\text{quad}},L,\mathcal{F}}(\varepsilon) \geq \min_{i\neq j}\max_{v\geq 0}\frac{(\varepsilon v - \xi_{ij}(v))_+^2}{2k\|P_\mathcal{F}\Delta_{ij}\|_2^2}, \quad \textit{where} \quad \xi_{ij}(v) := \left\| L^\mathsf{T}(v\mathbf{I}_k - P_\mathcal{F})\Delta_{ij} \right\|_\infty, \tag{12}$$

where $P_\mathcal{F}$ is the orthogonal projection on the subspace $\mathcal{F}$, $(x)_+^2 := [x > 0]x^2$ is the truncation of the parabola to its right branch and $\Delta_{ij} := \mathbf{e}_i - \mathbf{e}_j \in \mathbb{R}^k$ with $\mathbf{e}_c \in \mathbb{R}^k$ being the c-th column of the

[7] We do a pointwise analysis in this section, so we are not modeling the dependence of $\boldsymbol{\theta}$ on the features $\boldsymbol{x}$. However, in an actual implementation, the vector $\boldsymbol{\theta}$ should be a function of the features $\boldsymbol{x}$ coming from some flexible family such as a RKHS or some neural networks.

identity matrix $\mathbf{I}_k$. By convention, if both numerator and denominator of (12) equal zero the whole bound equals zero. If only the denominator equals zero then the whole bound equals infinity (the particular pair of i and j is effectively not considered).

The proof of Theorem 3 starts with using the idea of [14] to compute the calibration function by solving a collection of convex quadratic programs (QPs). Then we diverge from the proof of [14] (because it leads to a non-informative bound in inconsistent settings). For each of the formulated QPs, we construct a dual by using the approach of Dorn [10]. The dual of Dorn is convenient for our needs because it does not require inverting the matrix defining the quadratic terms (compared to the standard Lagrangian dual). The complete proof is given in Appendix B.

Remark 4. *The numerator of the bound (12) explicitly specifies the point at which the bound becomes non-zero, implying level-η consistency with $\eta = \frac{\xi_{ij}(v)}{v}$ for the values of i, j, v that are active for a particular ε. The quantity $\frac{v^2}{2k\|P_{\mathcal{F}}\Delta_{ij}\|_2^2}$ bounds the weight of the ε^2 term in the calibration function after it leaves zero. Moving the quantity v defines the trade-off between the slope, which is related to the convergence speed of the algorithm, and the value of η defining the best achievable accuracy.*

Remark 5. *If we have conditions of Theorem 7 of [14] satisfied, i.e., $\operatorname{span}(L) \subseteq \mathcal{F}$, then the vector $L^\mathsf{T}(\mathbf{I}_k - P_{\mathcal{F}})\Delta_{ij}$ equals zero and $\xi_{ij}(v)$ becomes $|v-1|\,\|L^\mathsf{T}\Delta_{ij}\|_\infty$, which equals zero when $v = 1$. It might seem that having $v > 1$ can potentially give us a tighter lower bound than Theorem 7 [14] even in consistent cases. However, the quantity $\|L^\mathsf{T}\Delta_{ij}\|_\infty$ upper bounds the maximal possible (w.r.t. the conditional distribution $\mathbf{P}_{\mathcal{D}}(\cdot \mid \boldsymbol{x})$) value of the excess task loss for a fixed pair i, j leading to the identity $v\varepsilon - |v-1|\,\|L^\mathsf{T}\Delta_{ij}\|_\infty = \|L^\mathsf{T}\Delta_{ij}\|_\infty$ for $\varepsilon = \|L^\mathsf{T}\Delta_{ij}\|_\infty$ and $v \geq 1$. Together with the convexity of the function $(x)_+^2$, this implies that the best possible value of v in consistent settings equals one.*

Remark 6. *Setting v in (12) to any non-negative constant gives a valid lower bound. In particular, setting v to 1 (while potentially making the bound less tight) highlights the separation between the weight of the quadratic term and the best achievable accuracy η. The bound now reads as follows:*

$$H_{\Phi_{\text{quad}},L,\mathcal{F}}(\varepsilon) \geq \min_{i \neq j} \frac{(\varepsilon - \xi_{ij})_+^2}{2k\|P_{\mathcal{F}}\Delta_{ij}\|_2^2}, \quad \textit{where} \quad \xi_{ij} := \left\| L^\mathsf{T}(\mathbf{I}_k - P_{\mathcal{F}})\Delta_{ij} \right\|_\infty. \tag{13}$$

Note that the weight of the ε^2 term now equals the corresponding coefficient of the bound of Theorem 7 [14]. Notably, this weight depends only on the score subspace $\mathcal{F}$, but not on the loss L.

5 Bounds for Particular Losses

5.1 Multi-Class Classification with the Tree-Structured Loss

As an illustration of the obtained lower bound (12), we consider the task of multi-class classification and the *tree-structured loss*, which is defined for a weighted tree built on labels (such trees on labels often appear in settings with large number of labels, e.g., extreme classification [6]). Leaves in the tree correspond to the class labels $\hat{y} \in \hat{\mathcal{Y}} = \mathcal{Y}$ and the loss function is defined as the length of the path ρ between the leaves, i.e., $L_{\text{tree}}(y, \hat{y}) := \rho(y, \hat{y})$. To compute the lower bound exactly, we assume that the number of children d_s and the weights of the edges connecting a node with its children $\frac{\alpha_s}{2}$ are equal for all the nodes of the same depth level $s = 0, \ldots, D-1$ (see Figure 2 in Appendix C for an example of such a tree) and that $\sum_{s=0}^{D-1} \alpha_s = 1$, which normalizes $L_{\max}$ to one.

To define the score matrix $\mathcal{F}_{\text{tree},s_0}$, we set the *consistency depth* $s_0 \in \{1, \ldots, D\}$ and restrict the scores $\boldsymbol{f}$ to be equal for the groups (blocks) of leaves that have the same ancestor on the level s_0. Let $B(i)$ be the set of leaves that have the same ancestor as a leaf i at the depth s_0. With this notation, we have $\mathcal{F}_{\text{tree},s_0} = \operatorname{span}\{\sum_{i \in B(j)} \mathbf{e}_i \mid j = 1, \ldots, k\}$. Theorem 3 gives us the bound (see Appendix C):

$$H_{\Phi_{\text{quad}},L_{\text{tree}},\mathcal{F}_{\text{tree},s_0}}(\varepsilon) \geq [\varepsilon > \eta_{s_0}] \frac{(\eta_{s_0} - \bar{\rho}_{s_0} + \alpha_{s_0-1})^2}{(\frac{\eta_{s_0}}{2} + \alpha_{s_0-1})^2} \frac{(\varepsilon - \frac{\eta_{s_0}}{2})_+^2}{4b_{s_0}}, \tag{14}$$

where b_{s_0}, $\bar{\rho}_{s_0} := \frac{1}{|B(j)|}\sum_{i \in B(j)} \rho(i,j) = \sum_{s=s_0}^{D-1} \alpha_s \frac{(\prod_{s'=s_0}^{s} d_{s'}) - 1}{\prod_{s'=s_0}^{s} d_{s'}}$ and $\eta_{s_0} := \max_{i \in B(j)} \rho(i,j) = \sum_{s=s_0}^{D-1} \alpha_s$ are the number of blocks, the average and maximal distance within a block, respectively.

Now we discuss the behavior of the bound (14) when changing the truncation level s_0. With the growth of s_0, the level of consistency η_{s_0} goes to 0 indicating that more labels can be distinguished. At the same time, we have $\frac{\eta_{s_0}}{2} \leq \bar{\rho}_{s_0}$ for the trees we consider and thus the coefficient in front of the

ε^2 term can be bounded from above by $\frac{1}{4b_{s_0}}$, which means that the lower bound on the calibration function decreases at an exponential rate with the growth of s_0. These arguments show the trade-off between the level of consistency and the coefficient of ε^2 in the calibration function.

Finally, note that the mixture of 0-1 and block 0-1 losses considered in [14, Appendix E.4] is an instance of the tree-structured loss with $D = 2$. Their bound [14, Proposition 17] matches (14) up to the difference in the definition of the calibration function (they do not have the $[\varepsilon > \eta_{s_0}]$ multiplier because they do not consider pairs of labels that fall in the same block).

5.2 Mean Average Precision (mAP) Loss for Ranking

The mAP loss, which is a popular way of measuring the quality of ranking, has attracted significant attention from the consistency point of view [4, 5, 18]. In the mAP setting, the ground-truth labels are binary vectors $\boldsymbol{y} \in \mathcal{Y} = \{0, 1\}^r$ that indicate the items relevant for the query (a subset of r items-to-rank) and the prediction consists in producing a permutation of items $\sigma \in \hat{\mathcal{Y}}$, $\hat{\mathcal{Y}} = \mathcal{S}_r$. The mAP loss is based on averaging the precision at different levels of recall and is defined as follows:

$$L_{\text{mAP}}(\sigma, y) := 1 - \frac{1}{|\boldsymbol{y}|} \sum_{p:y_p=1}^{r} \frac{1}{\sigma(p)} \sum_{q=1}^{\sigma(p)} y_{\sigma^{-1}(q)} = 1 - \sum_{p=1}^{r} \sum_{q=1}^{p} \frac{1}{\max(\sigma(p), \sigma(q))} \frac{y_p y_q}{|\boldsymbol{y}|}, \tag{15}$$

where $\sigma(p)$ is the position of an item p in a permutation σ, σ^{-1} is the inverse permutation and $|\boldsymbol{y}| := \sum_{p=1}^{r} y_p$. The second identity provides a convenient form of writing the mAP loss [18] showing that the loss matrix L_{mAP} is of rank at most $\frac{1}{2}r(r+1)$.[8] The matrix $F_{\text{mAP}} \in \mathbb{R}^{r! \times \frac{1}{2}r(r+1)}$ such that $(F_{\text{mAP}})_{\sigma, pq} := \frac{1}{\max(\sigma(p), \sigma(q))}$ is a natural candidate to define the score subspace $\mathcal{F}$ to get the consistent setting with the quadratic surrogate (11) (Eq. (15) implies that $\operatorname{span}(L_{\text{mAP}}) = \operatorname{span}(F_{\text{mAP}})$).

However, as noted in Section 6 of [18], although the matrix F_{mAP} is convenient from the consistency point of view (in the setup of [18]), it leads to the prediction problem $\max_{\sigma \in \mathcal{S}_r} (F_{\text{mAP}} \boldsymbol{\theta})_\sigma$, which is a quadratic assignment problem (QAP), and most QAPs are NP-hard.

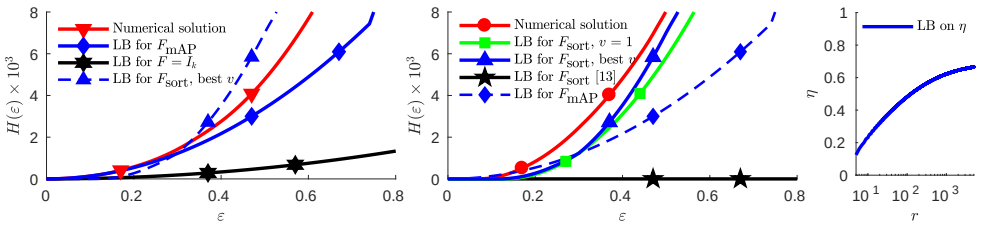
To be able to predict efficiently, it would be beneficial to have the matrix F with r columns such that sorting the r-dimensional $\boldsymbol{\theta}$ would give the desired permutation. It appears that it is possible to construct such a matrix by selecting a subset of columns of matrix F_{mAP}. We define $F_{\text{sort}} \in \mathbb{R}^{r! \times r}$ by $(F_{\text{sort}})_{\sigma, p} := \frac{1}{\sigma(p)}$. A solution of the prediction problem $\max_{\sigma \in \mathcal{S}_r} (F_{\text{sort}} \boldsymbol{\theta})_\sigma$ is simply a permutation that sorts the elements of $\boldsymbol{\theta} \in \mathbb{R}^r$ in the decreasing order (this statement follows from the fact that we can always increase the score $(F_{\text{sort}} \boldsymbol{\theta})_\sigma = \sum_{p=1}^{r} \frac{\theta_p}{\sigma(p)}$ by swapping a pair of non-aligned items).

Most importantly for our study, the columns of the matrix F_{sort} are a subset of the columns of the matrix F_{mAP}, which indicates that learning with the convenient matrix F_{sort} might be sufficient for the mAP loss. In what follows, we study the calibration functions for the loss matrix L_{mAP} and score matrices F_{mAP} and F_{sort}. In Figure 1a-b, we plot the calibration functions for both F_{mAP} and F_{sort} and the lower bounds given by Theorem 3. All the curves were obtained for $r = 5$ (computing the exact values of the calibration functions is exponential in r).

Next, we study the behavior of the lower bound (12) for large values of r. In Lemma 13 of Appendix D, we show that the denominator of the bound (12) is not exponential in r (we have $2r! \|P_{\mathcal{F}_{\text{sort}}} \Delta_{\pi\omega}\|_2^2 = O(r)$). We also know that $\|P_{\mathcal{F}_{\text{sort}}} \Delta_{\pi\omega}\|_2^2 \leq \|P_{\mathcal{F}_{\text{mAP}}} \Delta_{\pi\omega}\|_2^2$ (because $\mathcal{F}_{\text{sort}}$ is a subspace of $\mathcal{F}_{\text{mAP}}$), which implies that the calibration function of the consistent setting grows not faster than the one of the inconsistent setting. We can also numerically compute a lower bound on the point η until which the calibration function is guaranteed to be zero (for this we simply pick two permutations π, ω and a labeling $\boldsymbol{y}$ that delivers large values of $\left(L_{\text{mAP}}^{\mathsf{T}} (\mathbf{I}_k - P_{\mathcal{F}_{\text{sort}}}) \Delta_{ij}\right)_{\boldsymbol{y}} \leq \xi_{\pi,\omega}(1)$). Figure 1c shows that the level of inconsistency η grows with the growth of r, which makes the method less appealing for large-scale settings.

Finally, note that to run the ASGD algorithm for the quadratic surrogate (11), mAP loss and score matrix F_{sort}, we need to efficiently compute $F_{\text{sort}}^{\mathsf{T}} F_{\text{sort}}$ and $F_{\text{sort}}^{\mathsf{T}} L_{\text{mAP}}(:, \boldsymbol{y})$. Lemmas 11 and 12 (see Appendix D) provide linear in r time algorithms for doing this. The condition number of $\mathcal{F}_{\text{sort}}$ grows as $\Theta(\log r)$ keeping the sample complexity bound (10) well behaved.

[8] Ramaswamy & Agarwal [17, Proposition 21] showed that the rank of L_{mAP} is a least $\frac{1}{2}r(r+1) - 2$.



(a): Consistent surrogate with F_{mAP} (b): Inconsistent surrogate with F_{sort} (c): LB on η

Figure 1: **Plot (a)** shows the calibration function $H_{\Phi_{\text{quad}}, L_{\text{mAP}}, \mathcal{F}_{\text{mAP}}}(\varepsilon)$ for L_{mAP} (red line) obtained numerically. The solid blue line [14, Theorem 7] is its lower bound, LB, and the solid black line is the worst case bound obtained for $F = \mathbf{I}_{r!}$ (which means not constructing an appropriate low-dimension $\mathcal{F}$). Difference between the blue and the black lines is exponential (proportional to $r!$). The dashed blue line illustrates the inconsistent surrogate (note that it is zero for small $\varepsilon > 0$, but then grows faster than the solid blue line – the consistent setting). **Plot (b)** shows the calibration function $H_{\Phi_{\text{quad}}, L_{\text{mAP}}, \mathcal{F}_{\text{sort}}}(\varepsilon)$ (red line) obtained numerically (this setting is level-η consistent for $\eta \approx 0.08$). The blue line (Theorem 3) is its lower bound for the optimal value of v and the green line is the bound for $v = 1$ (easier to obtain). The black line shows the zero-valued trivial bound from [14]. The dashed blue line shows $H_{\Phi_{\text{quad}}, L_{\text{mAP}}, \mathcal{F}_{\text{mAP}}}(\varepsilon)$ for the consistent surrogate to compare the two settings. Note that in both plots (a) and (b) the solid blue lines are the lower bounds of the corresponding calibration functions (red lines), but the dashed blue lines are not (shown for comparison purposes). **Plot (c)** shows a lower bound on the point η where the exact calibration function $H_{\Phi_{\text{quad}}, L_{\text{mAP}}, \mathcal{F}_{\text{sort}}}(\varepsilon)$ stops being zero, indicating the level of consistency (Definition 2).

6 Discussion

Related works. Despite a large number of works studying consistency and calibration in the context of machine learning, there have been relatively few attempts to obtain guarantees for inconsistent surrogates. The most popular approach is to study consistency under so-called *low noise* conditions. Such works show that under certain assumptions on the data generating distribution $\mathcal{D}$ (usually these assumptions are on the conditional distribution of labels and are impossible to verify for real data) the surrogate of interest becomes consistent, whereas being inconsistent for general $\mathcal{D}$. Duchi et al. [11] established such a result for the value-regularized linear surrogate for ranking (which resembles the pairwise disagreement, PD, loss). Ramaswamy et al. [18] provided similar results for the mAP and PD losses for ranking and their quadratic surrogate. Similarly to our conclusions, the mAP surrogate of [18] is consistent with $\frac{1}{2}r(r+1)$ parameters learned and only low-noise consistent with r parameters learned. Long & Servedio [12] introduced a notion of realizable consistency w.r.t. a function class (they considered linear predictors), which is consistency w.r.t. the function class assuming the data distribution such that labels depend on features deterministically with this dependency being in the correct function class. Ben-David et al. [3] worked in the agnostic setting for binary classification (no assumptions on the underlying $\mathcal{D}$) and provided guarantees on the error of linear predictors when the margin was bounded by some constant (their work reduces to consistency in the limit case, but is more general).

Conclusion. Differently from the previous approaches, we do not put constraints on the data generating distribution, but instead study the connection between the surrogate and task losses by the means of the calibration function (following [14]), which represents the worst-case scenario. For the quadratic surrogate (11), we can bound the calibration function from below in such a way that the bound is non-trivial in inconsistent settings (differently from [14]). Our bound quantifies the level of inconsistency of a setting (defined by the used surrogate loss, task loss and parametrization of the scores) and allows to analyze when learning with inconsistent surrogates can be beneficial. We illustrate the behavior of our bound for two tasks (multi-class classification and ranking) and show examples of conclusions that our approach can give.

Future work. It would be interesting to combine our quantitative analysis with the constraints on the data distribution, which might give adaptive calibration functions (in analogy to adaptive convergence rates in convex optimization: for example, SAGA [9] has a linear convergence rate for strongly convex objectives and $1/t$ rate for non-strongly convex ones), and with the recent results of Pillaud-Vivien et al. [16] showing that under some low-noise assumptions even slow convergence of the surrogate objective can imply exponentially fast convergence of the task loss.

Acknowledgements

This work was partly supported by Samsung Research, by Samsung Electronics, by the Ministry of Education and Science of the Russian Federation (grant 14.756.31.0001) and by the NSERC Discovery Grant RGPIN-2017-06936.

References

[1] Ávila Pires, Bernardo, Ghavamzadeh, Mohammad, and Szepesvári, Csaba. Cost-sensitive multiclass classification risk bounds. In *ICML*, 2013.

[2] Bartlett, Peter L., Jordan, Michael I., and McAuliffe, Jon D. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

[3] Ben-David, Shai, Loker, David, Srebro, Nathan, and Sridharan, Karthik. Minimizing the misclassification error rate using a surrogate convex loss. 2012.

[4] Buffoni, David, Gallinari, Patrick, Usunier, Nicolas, and Calauzènes, Clément. Learning scoring functions with order-preserving losses and standardized supervision. In *ICML*, 2011.

[5] Calauzènes, Clément, Usunier, Nicolas, and Gallinari, Patrick. On the (non-)existence of convex, calibrated surrogate losses for ranking. In *NIPS*, 2012.

[6] Choromanska, Anna, Agarwal, Alekh, and Langford, John. Extreme multi class classification. In *NIPS Workshop: eXtreme Classification*, 2013.

[7] Ciliberto, Carlo, Rudi, Alessandro, and Rosasco, Lorenzo. A consistent regularization approach for structured prediction. In *NIPS*, 2016.

[8] Crammer, Koby and Singer, Yoram. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research (JMLR)*, 2:265–292, 2001.

[9] Defazio, Aaron, Bach, Francis, and Lacoste-Julien, Simon. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, 2014.

[10] Dorn, William S. Duality in quadratic programming. *Quarterly of Applied Mathematics*, 18(2):155–162, 1960.

[11] Duchi, John C., Mackey, Lester W., and Jordan, Michael I. On the consistency of ranking algorithms. In *ICML*, 2010.

[12] Long, Phil and Servedio, Rocco. Consistency versus realizable H-consistency for multiclass classification. In *ICML*, 2013.

[13] Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4):1574–1609, 2009.

[14] Osokin, Anton, Bach, Francis, and Lacoste-Julien, Simon. On structured prediction theory with calibrated convex surrogate losses. In *NIPS*, 2017.

[15] Pedregosa, Fabian, Bach, Francis, and Gramfort, Alexandre. On the consistency of ordinal regression methods. *Journal of Machine Learning Research (JMLR)*, 18(55):1–35, 2017.

[16] Pillaud-Vivien, Loucas, Rudi, Alessandro, and Bach, Francis. Exponential convergence of testing error for stochastic gradient methods. In *COLT*, 2018.

[17] Ramaswamy, Harish G. and Agarwal, Shivani. Convex calibration dimension for multiclass loss matrices. *Journal of Machine Learning Research (JMLR)*, 17(14):1–45, 2016.

[18] Ramaswamy, Harish G., Agarwal, Shivani, and Tewari, Ambuj. Convex calibrated surrogates for low-rank loss matrices with applications to subset ranking losses. In *NIPS*, 2013.

[19] Steinwart, Ingo. How to compare different loss functions and their risks. *Constructive Approximation*, 26 (2):225–287, 2007.

[20] Taskar, Ben, Guestrin, Carlos, and Koller, Daphne. Max-margin markov networks. In *NIPS*, 2003.

[21] Tewari, Ambuj and Bartlett, Peter L. On the consistency of multiclass classification methods. *Journal of Machine Learning Research (JMLR)*, 8:1007–1025, 2007.

[22] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, 2005.

[23] Zhang, Tong. Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research (JMLR)*, 5:1225–1251, 2004.

[24] Zhang, Tong. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32(1):56–134, 2004.

Supplementary Material (Appendix)

Quantifying Learning Guarantees for Convex but Inconsistent Surrogates

Outline

Section A: Proofs of the two technical lemmas used in Theorem 3.
Section B: Proof of Theorem 3, which is the main result of this paper.
Section C: Lower bound on the calibration function for the tree-structured loss.
Section D: Derivations for the mean average precision loss.

A Technical Lemmas

In this section, we prove two technical lemmas that are used in the proofs of the main theoretical claims of the paper. These two lemmas are the generalizations of the two corresponding lemmas of [14].

Lemma 7 computes the excess of the weighted surrogate risk $\delta\phi$ for the quadratic loss Φ_{quad} (11), which is central to our analysis presented in Section 4. Lemma 7 generalizes Lemma 9 of [14] by removing the assumption of $\operatorname{span}(L) \subseteq \mathcal{F}$. Analogously to Lemma 9 [14], the key property of this result is that the excess $\delta\phi$ is jointly convex w.r.t. the parameters $\boldsymbol{\theta}$ and conditional distribution $\boldsymbol{q}$, which allows further analysis.

Lemma 8 allows to cope with the combinatorial aspect of the calibration function computation. In particular, when the excess of the weighted surrogate risk is convex, Lemma 8 reduces the computation of the calibration function to a set of convex optimization problems, which often can be solved analytically. Note that our Lemma 8 is slightly different from Lemma 10 of Osokin et al. [14] to deal with the difference of the definition of the excess population risk (6).

Lemma 7. *Consider the quadratic surrogate Φ_{quad} (11) defined for a task loss L. Let a subspace of scores $\mathcal{F} \subseteq \mathbb{R}^k$ be parametrized by $\boldsymbol{\theta} \in \mathbb{R}^r$, i.e., $\boldsymbol{f} = F\boldsymbol{\theta} \in \mathcal{F}$ with $F \in \mathbb{R}^{k \times r}$. Then, the excess of the weighted surrogate loss can be expressed as*

$$\delta\phi_{\text{quad}}(F\boldsymbol{\theta}, \boldsymbol{q}) := \phi_{\text{quad}}(F\boldsymbol{\theta}, \boldsymbol{q}) - \inf_{\boldsymbol{\theta}' \in \mathbb{R}^r} \phi_{\text{quad}}(F\boldsymbol{\theta}', \boldsymbol{q}) = \tfrac{1}{2k}\|F\boldsymbol{\theta} + P_{\mathcal{F}} L\boldsymbol{q}\|_2^2,$$

where $P_{\mathcal{F}} := F(F^\mathsf{T}F)^\dagger F^\mathsf{T}$ is the orthogonal projection on the subspace $\mathcal{F} = \operatorname{span}(F)$.

Proof. The proof is almost identical to the proof of Lemma 9 of [14] generalizing it only in the last equality. By the definition of the quadratic surrogate Φ_{quad} (11), we have

$$\begin{aligned}
\phi(\boldsymbol{f}(\boldsymbol{\theta}), \boldsymbol{q}) &= \tfrac{1}{2k}(\boldsymbol{\theta}^\mathsf{T}F^\mathsf{T}F\boldsymbol{\theta} + 2\boldsymbol{\theta}^\mathsf{T}F^\mathsf{T}L\boldsymbol{q}) + r(\boldsymbol{q}),\\
\boldsymbol{\theta}^* &:= \operatorname{argmin}_{\boldsymbol{\theta}} \phi(\boldsymbol{f}(\boldsymbol{\theta}), \boldsymbol{q}) = -(F^\mathsf{T}F)^\dagger F^\mathsf{T}L\boldsymbol{q},\\
\delta\phi(\boldsymbol{f}(\boldsymbol{\theta}), \boldsymbol{q}) &= \tfrac{1}{2k}(\boldsymbol{\theta}^\mathsf{T}F^\mathsf{T}F\boldsymbol{\theta} + 2\boldsymbol{\theta}^\mathsf{T}F^\mathsf{T}L\boldsymbol{q} + \boldsymbol{q}^\mathsf{T}L^\mathsf{T}F(F^\mathsf{T}F)^\dagger F^\mathsf{T}L\boldsymbol{q})\\
&= \tfrac{1}{2k}\|F\boldsymbol{\theta} + P_{\mathcal{F}}L\boldsymbol{q}\|_2^2,
\end{aligned}$$

where $r(\boldsymbol{q})$ denotes the quantity independent of the parameters $\boldsymbol{\theta}$. Note that if the assumption $\operatorname{span}(L) \subseteq \operatorname{span}(F)$ holds we have $P_{\mathcal{F}}L = L$, which is the statement of Lemma 9 [14]. □

Lemma 8. *For any task loss L, a surrogate loss Φ that is continuous and bounded from below, and a set of scores $\mathcal{F}$, the calibration function can be lower bounded as*

$$H_{\Phi,L,\mathcal{F}}(\varepsilon) \geq \min_{i \neq j} H_{ij}(\varepsilon), \tag{16}$$

where H_{ij} is defined via minimization of the same objective as (7), but w.r.t. a smaller domain:

$$
\begin{aligned}
H_{ij}(\varepsilon) = \inf_{\boldsymbol{f},\boldsymbol{q}} \; & \delta\phi(\boldsymbol{f},\boldsymbol{q}), \\
\text{s.t. } & \ell_i(\boldsymbol{q}) \le \ell_j(\boldsymbol{q}) - \varepsilon, \\
& \ell_i(\boldsymbol{q}) \le \ell_c(\boldsymbol{q}), \quad \forall c \in \hat{\mathcal{Y}}, \\
& f_j \ge f_c, \quad \forall c \in \hat{\mathcal{Y}}, \\
& \boldsymbol{f} \in \mathcal{F}, \\
& \boldsymbol{q} \in \Delta_m.
\end{aligned}
\tag{17}
$$

Here $\ell_c(\boldsymbol{q}) := (L\boldsymbol{q})_c$ is the expected loss if predicting label c. The index i represents a label with the smallest expected loss while the index j represents a label with the largest score.

Proof. We use the notation $\mathcal{F}_j$ to define the set of score vectors $\boldsymbol{f}$ where the predictor $\text{pred}(\boldsymbol{f})$ takes the value j, i.e., $\mathcal{F}_j := \{\boldsymbol{f} \in \mathcal{F} \mid \text{pred}(\boldsymbol{f}) = j\}$. The union of the sets $\mathcal{F}_j$, $j \in \hat{\mathcal{Y}}$, equals the whole set $\mathcal{F}$. Sets $\mathcal{F}_j$ might not contain their boundaries because of the usage of a particular tie-breaking strategy, thus we consider the sets $\overline{\mathcal{F}}_j := \{\boldsymbol{f} \in \mathcal{F} \mid f_j \ge f_c, \forall c \in \hat{\mathcal{Y}}\}$, which are the closures of $\mathcal{F}_j$ if $\mathcal{F}_j$ are not empty. It also might happen that because of a particular tie-breaking strategy a set $\mathcal{F}_j$ is empty, while the corresponding $\overline{\mathcal{F}}_j$ is not.

If $\boldsymbol{f} \in \mathcal{F}_j$, i.e. $j = \text{pred}(\boldsymbol{f})$, then the feasible set of probability vectors $\boldsymbol{q}$ for which a label i is one of the best possible predictions (i.e. $\delta\ell(\boldsymbol{f},\boldsymbol{q}) = \ell_j(\boldsymbol{q}) - \ell_i(\boldsymbol{q}) \ge \varepsilon$) equals

$$
\Delta_{m,i,j,\varepsilon} := \{\boldsymbol{q} \in \Delta_m \mid \ell_i(\boldsymbol{q}) \le \ell_c(\boldsymbol{q}), \forall c \in \hat{\mathcal{Y}}; \ell_j(\boldsymbol{q}) - \ell_i(\boldsymbol{q}) \ge \varepsilon\},
$$

because $\inf_{\boldsymbol{f}' \in \mathbb{R}^k} \ell(\boldsymbol{f}', \boldsymbol{q}) = \min_{c \in \hat{\mathcal{Y}}} \ell_c(\boldsymbol{q})$.

The union of the sets $\{\mathcal{F}_j \times \Delta_{m,i,j,\varepsilon}\}_{i,j \in \hat{\mathcal{Y}}, i \ne j}$ exactly equals the feasibility set of the optimization problem (7)-(8) (note that this is not true for the union of the sets $\{\overline{\mathcal{F}}_j \times \Delta_{m,i,j,\varepsilon}\}_{i,j \in \hat{\mathcal{Y}}, i \ne j}$, which can be strictly larger), thus we can rewrite the definition of the calibration function as follows:

$$
H_{\Phi,L,\mathcal{F}}(\varepsilon) = \min_{\substack{i,j \in \hat{\mathcal{Y}} \\ i \ne j}} \inf_{\substack{\boldsymbol{f} \in \mathcal{F}_j, \\ \boldsymbol{q} \in \Delta_{m,i,j,\varepsilon}}} \delta\phi(\boldsymbol{f},\boldsymbol{q}) \ge \min_{\substack{i,j \in \hat{\mathcal{Y}} \\ i \ne j}} \inf_{\substack{\boldsymbol{f} \in \overline{\mathcal{F}}_j, \\ \boldsymbol{q} \in \Delta_{m,i,j,\varepsilon}}} \delta\phi(\boldsymbol{f},\boldsymbol{q}) = \min_{i \ne j} H_{ij}(\varepsilon), \tag{18}
$$

which finishes the proof. Note that the inequality of (18) can be not tight only if some $\mathcal{F}_j$ is empty, but the corresponding $\overline{\mathcal{F}}_j$ is not (due to continuity of the function $\delta\phi(\boldsymbol{f},\boldsymbol{q})$, which follows from Lemma 27 of [23]). □

B Proof of Theorem 3

Theorem 3 (Lower bound on the calibration function $H_{\Phi_{\text{quad}},L,\mathcal{F}}(\varepsilon)$)**.** *For any task loss L, its quadratic surrogate Φ_{quad}, and a score subspace $\mathcal{F}$, the calibration function is bounded from below:*

$$
H_{\Phi_{\text{quad}},L,\mathcal{F}}(\varepsilon) \ge \min_{i \ne j} \max_{v \ge 0} \frac{(\varepsilon v - \xi_{ij}(v))_+^2}{2k\|P_{\mathcal{F}}\Delta_{ij}\|_2^2}, \quad \textit{where} \quad \xi_{ij}(v) := \left\| L^\mathsf{T}(v\mathbf{I}_k - P_{\mathcal{F}})\Delta_{ij} \right\|_\infty, \tag{19}
$$

$P_{\mathcal{F}}$ is the orthogonal projection on the subspace $\mathcal{F}$, $(x)_+^2 := [x > 0]x^2$ is the truncation of the parabola to its right branch and $\Delta_{ij} := \mathbf{e}_i - \mathbf{e}_j \in \mathbb{R}^k$ with $\mathbf{e}_c \in \mathbb{R}^k$ being the c-th column of the identity matrix $\mathbf{I}_k$. By convention, if both numerator and denominator of (19) equal zero the whole bound equals zero. If only the denominator equals zero then the whole bound equals infinity (the particular pair of i and j is effectively not considered).

Proof. First, let us assume that the score subspace $\mathcal{F}$ is defined as the column space of a matrix $F \in \mathbb{R}^{k \times r}$, i.e., $\boldsymbol{f}(\boldsymbol{\theta}) = F\boldsymbol{\theta}$. For technical convenience, we can also assume that F is of the full rank, $\text{rank}(F) = r$. Lemma 7 gives us the expression $\delta\phi_{\text{quad}}(F\boldsymbol{\theta},\boldsymbol{q}) = \frac{1}{2k}\|F\boldsymbol{\theta} + P_{\mathcal{F}}L\boldsymbol{q}\|_2^2$ for the excess surrogate, which is jointly convex w.r.t. a conditional probability vector $\boldsymbol{q}$ and parameters $\boldsymbol{\theta}$.

The optimization problem (7)-(8) is non-convex because the constraint (8) on the excess risk depends of the predictor function $\text{pred}(\boldsymbol{f})$, see Eq. (1), containing the argmax operation. However, if we

constrain the predictor to output label j, i.e., $f_j \geq f_c$, $\forall c$, and the label delivering the smallest possible expected loss to be i, i.e., $(L\boldsymbol{q})_i \leq (L\boldsymbol{q})_c$, $\forall c$, the problem becomes convex because all the constraints are linear and the objective is convex. Lemma 8 in Appendix A allows to bound the calibration function with the minimal w.r.t. selected labels i and j optimal value of one of the convex problems, i.e., $H_{\Phi_{\text{quad}},L,\mathcal{F}}(\varepsilon) \geq \min_{i \neq j} H_{ij}(\varepsilon)$, where $H_{ij}(\varepsilon)$ is defined as follows:

$$
\begin{aligned}
H_{ij}(\varepsilon) = \min_{\boldsymbol{\theta},\boldsymbol{q}} \; & \tfrac{1}{2k}\|F\boldsymbol{\theta} + P_{\mathcal{F}}L\boldsymbol{q}\|_2^2, \\
\text{s.t. } & (L\boldsymbol{q})_i \leq (L\boldsymbol{q})_j - \varepsilon, \\
& (L\boldsymbol{q})_i \leq (L\boldsymbol{q})_c, \;\; \forall c \in \hat{\mathcal{Y}}, \\
& (F\boldsymbol{\theta})_j \geq (F\boldsymbol{\theta})_c, \; \forall c \in \hat{\mathcal{Y}}, \\
& \boldsymbol{q} \in \Delta_m.
\end{aligned} \tag{20}
$$

To obtain a lower bound, we relax (20) by removing some of the constraints and arrive at

$$
\begin{aligned}
kH_{ij}(\varepsilon) \geq \min_{\boldsymbol{\theta},\boldsymbol{q}} \; & \tfrac{1}{2}\|F\boldsymbol{\theta} + P_{\mathcal{F}}L\boldsymbol{q}\|_2^2, && (21) \\
\text{s.t. } & \Delta_{ij}^{\mathsf{T}} L\boldsymbol{q} \leq -\varepsilon, && (22) \\
& \Delta_{ij}^{\mathsf{T}} F\boldsymbol{\theta} \leq 0, && (23) \\
& \mathbf{1}_m^{\mathsf{T}}\boldsymbol{q} = 1, && (24) \\
& q_c \geq 0, \;\; c = 1, \ldots, m. && (25)
\end{aligned}
$$

where $\Delta_{ij}^{\mathsf{T}} L\boldsymbol{q} = (L\boldsymbol{q})_i - (L\boldsymbol{q})_j$, $\Delta_{ij}^{\mathsf{T}} F\boldsymbol{\theta} = (F\boldsymbol{\theta})_i - (F\boldsymbol{\theta})_j$, and $\Delta_{ij} = \mathbf{e}_i - \mathbf{e}_j \in \mathbb{R}^k$ with $\mathbf{e}_c \in \mathbb{R}^k$ being a vector with 1 at position c and zeros elsewhere. Note that the relaxation defined by the problem (21)-(25) is tighter than the one used in the proof of Theorem 7 [14, Eq. (25)-(27)], because the latter omitted the simplex constraints (24)-(25).

We now explicitly build a dual problem to the QP (21)-(25). If we used the standard Lagrangian approach we would have to invert the matrix defining the objective, which is difficult. Instead we use the dual formulation of Dorn [10, Page 160], which allows to build a dual without inverting any matrices.[9] For the problem (21)-(25), this dual can be written as follows:

$$
\begin{aligned}
kH_{ij}(\varepsilon) \geq \max_{\boldsymbol{\theta},\boldsymbol{q},v_F \geq 0, v_L \geq 0, u} \; & -\tfrac{1}{2}\|F\boldsymbol{\theta} + P_{\mathcal{F}}L\boldsymbol{q}\|_2^2 + v_L\varepsilon + u, && (26) \\
& - v_L L^{\mathsf{T}}\Delta_{ij} + u\mathbf{1}_m - L^{\mathsf{T}}P_{\mathcal{F}}L\boldsymbol{q} - L^{\mathsf{T}}F\boldsymbol{\theta} \leq \mathbf{0}_m, && (27) \\
& - v_F F^{\mathsf{T}}\Delta_{ij} - F^{\mathsf{T}}L\boldsymbol{q} - F^{\mathsf{T}}F\boldsymbol{\theta} = \mathbf{0}_r. && (28)
\end{aligned}
$$

From the equality (28), we can express $F^{\mathsf{T}}L\boldsymbol{q} = -v_F F^{\mathsf{T}}\Delta_{ij} - F^{\mathsf{T}}F\boldsymbol{\theta}$ and substitute it in the objective (26) and inequality (27). Using the identities $P_{\mathcal{F}} = F(F^{\mathsf{T}}F)^{-1}F^{\mathsf{T}}$ and $P_{\mathcal{F}}F = F$, we can exclude variables $\boldsymbol{\theta}$, $\boldsymbol{q}$ and get a simpler bound. Note that this step leads to a valid lower bound because for any $v_F \geq 0$ there exist feasible values of variables $\boldsymbol{q}$ and $\boldsymbol{\theta}$ (we can take simply $\boldsymbol{q} = \mathbf{0}$, $\boldsymbol{\theta} = -v_F(F^{\mathsf{T}}F)^{-1}F^{\mathsf{T}}\Delta_{ij}$). The new bound depends on the three variables only:

$$
\begin{aligned}
kH_{ij}(\varepsilon) \geq \max_{v_F \geq 0, v_L \geq 0, u} \; & -\tfrac{1}{2}v_F^2\Delta_{ij}^{\mathsf{T}}P_{\mathcal{F}}\Delta_{ij} + v_L\varepsilon + u, && (29) \\
& - v_L L^{\mathsf{T}}\Delta_{ij} + u\mathbf{1}_m + v_F L^{\mathsf{T}}P_{\mathcal{F}}\Delta_{ij} \leq \mathbf{0}_m. && (30)
\end{aligned}
$$

[9] Here we show the dual of Dorn [10] for the exact combination of constraints we are using. In the dual formulation, $\boldsymbol{v}$ and $\boldsymbol{u}$ are the extra variables corresponding to the inequality and equality constraints, respectively.

The primal problem

$$
\begin{aligned}
\min_{\boldsymbol{q} \geq 0,\, \boldsymbol{\theta}} \; & \tfrac{1}{2}\begin{pmatrix}\boldsymbol{q}^{\mathsf{T}} & \boldsymbol{\theta}^{\mathsf{T}}\end{pmatrix}\begin{pmatrix}H_{\boldsymbol{qq}} & H_{\boldsymbol{q\theta}} \\ H_{\boldsymbol{q\theta}}^{\mathsf{T}} & H_{\boldsymbol{\theta\theta}}\end{pmatrix}\begin{pmatrix}\boldsymbol{q} \\ \boldsymbol{\theta}\end{pmatrix}, \\
\text{s.t. } & A_{\boldsymbol{q}}\boldsymbol{q} + A_{\boldsymbol{\theta}}\boldsymbol{\theta} \geq \boldsymbol{b} \\
& C_{\boldsymbol{q}}\boldsymbol{q} + C_{\boldsymbol{\theta}}\boldsymbol{\theta} = \boldsymbol{d}
\end{aligned}
$$

The dual problem

$$
\begin{aligned}
\max_{\boldsymbol{q}\,\boldsymbol{\theta},\, \boldsymbol{v} \geq 0,\, \boldsymbol{u}} \; & -\tfrac{1}{2}\begin{pmatrix}\boldsymbol{q}^{\mathsf{T}} & \boldsymbol{\theta}^{\mathsf{T}}\end{pmatrix}\begin{pmatrix}H_{\boldsymbol{qq}} & H_{\boldsymbol{q\theta}} \\ H_{\boldsymbol{q\theta}}^{\mathsf{T}} & H_{\boldsymbol{\theta\theta}}\end{pmatrix}\begin{pmatrix}\boldsymbol{q} \\ \boldsymbol{\theta}\end{pmatrix} + \boldsymbol{b}^{\mathsf{T}}\boldsymbol{v} + \boldsymbol{d}^{\mathsf{T}}\boldsymbol{u}, \\
\text{s.t. } & A_{\boldsymbol{q}}^{\mathsf{T}}\boldsymbol{v} + C_{\boldsymbol{q}}^{\mathsf{T}}\boldsymbol{u} - H_{\boldsymbol{qq}}\boldsymbol{q} - H_{\boldsymbol{q\theta}}\boldsymbol{\theta} \leq \mathbf{0} \\
& A_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{v} + C_{\boldsymbol{\theta}}^{\mathsf{T}}\boldsymbol{u} - H_{\boldsymbol{q\theta}}^{\mathsf{T}}\boldsymbol{q} - H_{\boldsymbol{\theta\theta}}\boldsymbol{\theta} = \mathbf{0}
\end{aligned}
$$

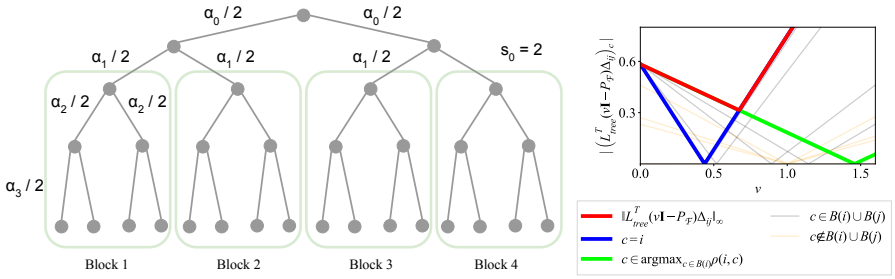


Figure 2: *Left:* An example of the tree-structured loss for the task of multi-class classification. *Right:* Illustration of the proof of Lemma 9 (best viewed in color). The thin gray and brown lines show the absolute values of the components of the vector $L_{\text{tree}}^{\mathsf{T}}(v\mathbf{I} - P_{\mathcal{F}_{\text{tree}}})\Delta_{ij}$ as functions of v. The bold blue and green lines correspond to the components at which the maximum value is achieved. The bold red line shows the resulting norm $\|L_{\text{tree}}^{\mathsf{T}}(v\mathbf{I} - P_{\mathcal{F}_{\text{tree}}})\Delta_{ij}\|_\infty$.

First, consider the case $\Delta_{ij}^{\mathsf{T}} P_{\mathcal{F}} \Delta_{ij} = \|P_{\mathcal{F}}\Delta_{ij}\|_2^2 \neq 0$. Given that $v_F \geq 0$ we can change the variables by introducing $\hat{v}_F := v_F \|P_{\mathcal{F}}\Delta_{ij}\|_2^2$, $v := v_L / v_F$, $\hat{u} := u / v_F$ after which we get

$$kH_{ij}(\varepsilon) \geq \frac{1}{\|P_{\mathcal{F}}\Delta_{ij}\|_2^2} \max_{\hat{v}_F \geq 0, \hat{v}_L \geq 0, \hat{u}} \; -\tfrac{1}{2}\hat{v}_F^2 + \hat{v}_F(v\varepsilon + \hat{u}), \tag{31}$$

$$-vL^{\mathsf{T}}\Delta_{ij} + \hat{u}\mathbf{1}_m + L^{\mathsf{T}} P_{\mathcal{F}} \Delta_{ij} \leq \mathbf{0}_m. \tag{32}$$

The global minimum of this function w.r.t. the variable $\hat{v}_F$ can be found analytically: if $v\varepsilon + \hat{u} \geq 0$ it equals $\frac{1}{2\|P_{\mathcal{F}}\Delta_{ij}\|_2^2}(v\varepsilon + \hat{u})^2$, and zero otherwise. The constraint (32) on $\hat{u}$ can be substituted with $\hat{u} = -\big\| L^{\mathsf{T}}(v\mathbf{I}_k - P_{\mathcal{F}})\Delta_{ij} \big\|_\infty =: -\xi_{ij}(v)$, because we always consider both $H_{ij}(\varepsilon)$ and $H_{ji}(\varepsilon)$ when bounding the calibration function.

Now, consider the boundary case of $\Delta_{ij}^{\mathsf{T}} P_{\mathcal{F}} \Delta_{ij} = \|P_{\mathcal{F}}\Delta_{ij}\|_2^2 = 0$. The problem (29)-(30) becomes $\frac{1}{2}\max_{v \geq 0} v(\varepsilon + \min L^{\mathsf{T}}\Delta_{ij})$ implying that the objective equals 0 if $\varepsilon + \min L^{\mathsf{T}}\Delta_{ij} \leq 0$. Otherwise, the objective equals $+\infty$, which corresponds to the in-feasibility of the constraint (22) of the primal problem. Note that because we always consider both $H_{ij}(\varepsilon)$ and $H_{ji}(\varepsilon)$ when bounding the calibration function we can substitute $v\min(L^{\mathsf{T}}\Delta_{ij})$ with $-\xi_{ij}(v)$. □

C Lower Bound on the Calibration Function for the Tree-Structured Loss

In this section, we compute the lower bound on the calibration function for the tree-structured loss defined in Section 5.1.

Lemma 9. *For a particular consistency depth s_0 and for the corresponding subspace $\mathcal{F}_{\text{tree},s_0}$, the projection operator $P_{\mathcal{F}_{\text{tree}},s_0}$ at Δ_{ij} is computed as*

$$P_{\mathcal{F}_{\text{tree}},s_0}\Delta_{ij} = \begin{cases} 0, & i \in B(j), \\ \frac{1}{|B(j)|}\left(\sum_{k \in B(i)} \mathbf{e}_k - \sum_{k \in B(j)} \mathbf{e}_k\right) & i \notin B(j). \end{cases} \tag{33}$$

The vectors $\xi_{ij}(v)$ are computed as

$$\xi_{ij}(v) = \begin{cases} v\rho(i,j) & i \in B(j), \\ \max\{|(v-1)\rho(i,j)|, |v(\rho(i,j) - \eta) - (\rho(i,j) - \bar{\rho})|\} & i \notin B(j), \end{cases} \tag{34}$$

for $\eta := \max_{c \in B(i)} \rho(i,c)$ and $\bar{\rho} := \frac{1}{|B(j)|}\sum_{c \in B(i)} \rho(i,c)$. Finally, the following lower bound of the calibration function for the loss L_{tree}, its quadratic surrogate Φ_{quad} and the score subspace $\mathcal{F}_{\text{tree},s_0}$ holds:

$$H_{\Phi_{\text{quad}},L_{\text{tree}},\mathcal{F}_{\text{tree},s_0}}(\varepsilon) \geq [\varepsilon > \eta]\frac{(\nu-\bar{\rho})^2}{(\nu-\frac{\eta}{2})^2}\frac{(\varepsilon-\frac{\eta}{2})_+^2}{4b}, \tag{35}$$

where $\nu := \min_{c \notin B(i)} \rho(i,c) > \eta$ and b is the number of blocks when the tree is cut at the depth s_0.

Proof. For brevity, we shortcut the notation F_{tree,s_0} to F, $\mathcal{F}_{\text{tree},s_0}$ to $\mathcal{F}$ and L_{tree} to L. First, we compute the projection operator $P_{\mathcal{F}}\mathbf{e}_i$ and the lower-bound denominator $2k\|P_{\mathcal{F}}\Delta_{ij}\|_2^2$. Recall, that the subspace of allowed scores $\mathcal{F}$ defined as $\operatorname{span}\{\sum_{l\in B(j)}\mathbf{e}_l | j=1,\dots,k\}$ is of dimension b. The vector $\mathbf{e}_i$ is orthogonal to the $b-1$ different vectors $\sum_{l\in B(j)}\mathbf{e}_l$, $j\notin B(i)$, thus the projection $P_{\mathcal{F}}\mathbf{e}_i$ equals the projection of $\mathbf{e}_i$ on the vector $\sum_{l\in B(i)}\mathbf{e}_l$, which equals $\frac{1}{s}\sum_{l\in B(i)}\mathbf{e}_l$ with $s:=B(i)=\frac{k}{b}$. The projection square norm $\|P_{\mathcal{F}}\Delta_{ij}\|_2^2$ equals $\frac{2}{s}=\frac{2b}{k}$ if $i\notin B(j)$ and 0 if $i\in B(j)$.

Next, we compute $\xi_{ij}(v)$ defined as $\|L^{\mathsf{T}}(vI-P_{\mathcal{F}})\Delta_{ij}\|_\infty$. By definition of the loss function, the element of the loss matrix $L_{ci}=(L\mathbf{e}_i)_c$ equals the tree distance from the leaf i to the leaf c. The projection operator $P_{\mathcal{F}}\mathbf{e}_i$ equals the vector $\frac{1}{s}\sum_{l\in B(i)}\mathbf{e}_l$, therefore $(LP_{\mathcal{F}}\mathbf{e}_i)_c$ is equal to the average tree distance from the elements of the block $B(i)$ to c: $\frac{1}{s}\sum_{l\in B(i)}L_{lc}$, which we denote by $\bar\rho(i,c)$. Note that the average distance $\bar\rho(i,c)$ is equal for all the leaves c that belong to the same block $B(c)$. With this notation, we have the following equality:

$$\|L^{\mathsf{T}}(vI-P_{\mathcal{F}})\Delta_{ij}\|_\infty=\max_{c\in\hat{\mathcal{Y}}}|v(\rho(i,c)-\rho(j,c))-(\bar\rho(i,c)-\bar\rho(j,c))|. \tag{36}$$

On the right-hand side, each component is the absolute value of a linear in v function, which equals zero at $v=\frac{\bar\rho(i,c)-\bar\rho(j,c)}{\rho(i,c)-\rho(j,c)}$ and the absolute value of the slope equals $|\rho(i,c)-\rho(j,c)|$. We consider the cases when the labels i and j are in the same and different blocks separately.

If i and j are in the same block we have that $\bar\rho(i,c)=\bar\rho(j,c)$ and, by the reverse triangle inequality, $|\rho(i,c)-\rho(j,c)|\le\rho(i,j)$ with the equality holding for $c=i$ or $c=j$, which implies that $\xi_{ij}(v)=\|(L^{\mathsf{T}}(vI-P_{\mathcal{F}})\Delta_{ij}\|_\infty=v\rho(i,j)$ for $i\in B(j)$.

Now, we study the second case where i and j are in different blocks. We first show that

$$|\bar\rho(i,c_1)-\bar\rho(j,c_1)|\le|\bar\rho(i,c_2)-\bar\rho(j,c_2)|\quad\text{if } c_1\notin B(i)\cup B(j) \text{ and } c_2\in B(i)\cup B(j). \tag{37}$$

This inequality is crucial for the proof and holds due to the restriction on tree weights and node degrees.

In this paragraph, we will show that the left-hand side of the inequality (37) achieves its maximum when $c_1\notin B(i)\cup B(j)$ is in the block closest to $B(j)$ (or, due to the loss symmetries, in the block closest to $B(i)$). If the lowest common ancestor of i and j is not an ancestor of c_1 the difference of the average distances equals zero due to equality of the paths from the lowest common ancestor to i and j. Otherwise, there exists $c_1\notin B(i)\cup B(j)$ such that the lowest common ancestor of i and j is an ancestor of c_1. Then, $\bar\rho(j,c_1)$ is minimized and $\bar\rho(i,c_1)$ is simultaneously maximized for a component c_1 closest to the block $B(j)$. In this case, the left-hand side maximum value equals $\bar\rho(i,j)-\min_{c\notin B(j)}\bar\rho(j,c)$ because $\bar\rho(i,j)=\bar\rho(i,c_1)$.

The right-hand side of the inequality (37) is the same for any choice of $c_2\in B(i)\cup B(j)$ and is equal to $\bar\rho(i,j)-\bar\rho(j,c_2)$ for some $c_2\in B(j)$. Since the average distance within the block is smaller than the average distance to any node outside of the block, i.e., $\min_{c\notin B(j)}\bar\rho(j,c)\ge\bar\rho(j,c_2)$ for $c_2\in B(j)$, the inequality (37) holds. The same arguments also show that

$$|\rho(i,c_1)-\rho(j,c_1)|\le|\rho(i,c_2)-\rho(j,c_2)|\quad\text{if } c_1\notin B(i)\cup B(j) \text{ and } c_2\in B(i)\cup B(j). \tag{38}$$

Recall that in our case the infinity norm in (36) equals the component-wise maximum of the absolute values of the linear functions of v. We will show below that for a small enough v the maximum is achieved at the components that have the smallest slope $|\rho(i,c)-\rho(j,c)|$ among the ones with the largest offset $|\bar\rho(i,c)-\bar\rho(j,c)|$ and from some point for larger values of v the maximum is achieved at the components with the steepest slope (see Figure 2 right for the illustration).

Consider a leaf $c_2\in B(i)$ farthest from the leaf i, i.e., $c_2\in\operatorname{argmax}_{c\in B(i)}\rho(i,c)$ (defines the green line in Figure 2 right). The offset $|\bar\rho(i,c_2)-\bar\rho(j,c_2)|$ is the same for all $c_2\in B(i)$ and, by (37), is larger than the offsets of the components $c_1\notin B(i)\cup B(j)$. The slope $|\rho(i,c_2)-\rho(j,c_2)|$ is the smallest among the components in $B(i)\cup B(j)$. The component c_2 of $L^{\mathsf{T}}(v\mathbf{I}-P_{\mathcal{F}})\Delta_{ij}$ equals zero for $v^*_{c_2}:=\frac{\bar\rho(j,c_2)-\bar\rho(i,c_2)}{\rho(j,c_2)-\rho(i,c_2)}=\frac{\rho(i,j)-\bar\rho(i,c_2)}{\rho(i,j)-\rho(i,c_2)}$, where $v^*_{c_2}>1$ by definition of c_2, i.e., because $\rho(i,c_2)$ is the maximal distance, which is not smaller than the average distance $\bar\rho(i,c_2)$. Finally, for $v\le1$ this component has higher values than the values of the components $c\notin B(i)\cup B(j)$. Indeed, the latter

are equal to zero at $v = 1$ and have smaller offset at $v = 0$ (thin brown lines in Figure 2 right for $v < 1$).

The component i of $L^\mathsf{T}(v\mathbf{I} - P_\mathcal{F})\Delta_{ij}$ has the steepest slope $|\rho(i,j) - \rho(i,i)| = \rho(i,j)$ and the same offset as in the previous paragraph $|\bar\rho(i,i) - \bar\rho(i,j)| = |\bar\rho(i,c_2) - \bar\rho(j,c_2)|$ (defines the blue line in Figure 2 right). The component equals zero for $v_i^* := \frac{\bar\rho(j,i)-\bar\rho(i,i)}{\rho(j,i)-\rho(i,i)} = \frac{\rho(j,i)-\bar\rho(i,i)}{\rho(j,i)}$, where $v_i^* \leq 1$. As a result, the component i has higher values than the components $c \notin B(i) \cup B(j)$, since they have smaller slope $|\rho(i,c) - \rho(i,c)|$ (due to the inequality (38)) and equal zero for $v = 1$ (thin brown lines in Figure 2 right for $v > 1$).

Since all the components $c \in B(i) \cup B(j)$ have the same offset, the maximum is achieved either at c_2 or at i:

$$\|L^\mathsf{T}(v\mathbf{I} - P_\mathcal{F})\Delta_{ij}\|_\infty = \max\{|v\rho(i,j) - (\rho(i,j) - \bar\rho)|, |v(\rho(i,j) - \eta) - (\rho(i,j) - \bar\rho)|\}, \quad (39)$$

where $\eta := \rho(i,c_2)$ is the maximal distance within a block and $\bar\rho := \bar\rho(i,c_2)$ is the average distance within a block.

Next, we compute $\max_{v\geq 0}(\varepsilon v - \xi_{ij}(v))_+^2$. If i and j are in the same block we have $(\varepsilon v - \xi_{ij}(v))_+^2 = (v(\varepsilon - \rho(i,j)))_+^2$, which equals zero for $\varepsilon \leq \rho(i,j)$ and $+\infty$ otherwise. If i and j are in different blocks we have the maximum value of $(\varepsilon v - \xi_{ij}(v))_+^2$ equal to $+\infty$ when $\varepsilon > \rho(i,j)$. In the case when $\varepsilon \leq \rho(i,j)$, the maximum is achieved at the intersection point $v\rho(i,j) - (\rho(i,j) - \bar\rho) = -v(\rho(i,j) - \eta) + (\rho(i,j) - \bar\rho)$, $v = \frac{2(\rho(i,j)-\bar\rho)}{2\rho(i,j)-\eta}$. The maximum value is positive if and only if $\varepsilon > \frac{\eta}{2}$, so for $\varepsilon \leq \rho(i,j)$ we obtain

$$\max_{v\geq 0}(\varepsilon v - \xi_{ij}(v))_+^2 = \tfrac{(\rho(i,j)-\bar\rho)^2}{(\rho(i,j)-\frac{\eta}{2})^2}(\varepsilon - \tfrac{\eta}{2})_+^2 \quad (40)$$

and $+\infty$ otherwise.

Finally, to get the actual lower bound on the calibration function, we compute the minimum with respect to all labels $\min_{i\neq j}\max_{v\geq 0}(\varepsilon v - \xi_{ij}(v))_+^2$. When i and j are in the same block, they deliver minimum value 0 for $\varepsilon \leq \rho(i,j)$ and the maximum value of $\rho(i,j)$ within a block equals η by definition of η. For $\varepsilon > \eta$, the minimum is delivered by i and j in different blocks. For the average distance within the block, we have $\bar\rho \geq \frac{\eta}{2}$ for the trees with the number of children and the weights of edges equal at the same depth level, therefore the outer minimum w.r.t. i and j is achieved at the smallest distance between two blocks $\nu := \min_{i\notin B(j)}\rho(i,j) > \eta$. As a result, we obtain the bound

$$H_{\Phi_{\text{quad}},L_{\text{tree}},\mathcal{F}_{\text{tree},s_0}}(\varepsilon) \geq [\varepsilon > \eta]\tfrac{(\nu-\bar\rho)^2}{(\nu-\frac{\eta}{2})^2}\tfrac{(\varepsilon-\frac{\eta}{2})_+^2}{4b}, \quad (41)$$

which completes the proof. □

In the next lemma, we compute the quantities η, $\bar\rho$, ν using the tree weights $\{\frac{1}{2}\alpha_s\}_{s=0}^{D-1}$ to finish the computation of the bound (14) of the main paper.

Lemma 10. *For a particular consistency depth s_0 and the corresponding subspace $\mathcal{F}_{\text{tree},s_0}$, the maximum distance within an arbitrary block η_{s_0}, the minimum distance between a leaf in a block and a leaf outside the block ν_{s_0} and the average distance within a block $\bar\rho_{s_0}$ can be computed as follows:*

$$\eta_{s_0} = \max_{i\in B(j)}\rho(i,j) = \sum_{s=s_0}^{D-1}\alpha_s \quad (42)$$

$$\nu_{s_0} = \min_{i\notin B(j)}\rho(i,j) = \sum_{s=s_0-1}^{D-1}\alpha_s \quad (43)$$

$$\bar\rho_{s_0} = \tfrac{1}{|B(j)|}\sum_{i\in B(j)}\rho(i,j) = \sum_{s=s_0}^{D-1}\alpha_s\tfrac{(\prod_{s'=s_0}^{s}d_{s'})-1}{\prod_{s'=s_0}^{s}d_{s'}}. \quad (44)$$

Proof. The expressions for η_{s_0} and ν_{s_0} immediately follow from the definition of the distance $\rho(i,j)$.

To obtain the expression for $\bar{\rho}_{s_0}$, we rewrite the distance $\rho(i,j)$ between leaves i and j in the same block $B(j)$ as the weighted sum of indicators:

$$\rho(i,j) = \sum_{s=s_0}^{D-1} \alpha_s [\text{path from } i \text{ to } j \text{ contains an edge of depth } s]. \tag{45}$$

Then, we fix a leaf j and compute the number of paths from j to the leaves in $B(j)$ that contain an edge of depth s. Such paths go through the same node (the ancestor of j at the depth s) on the way up from node j and go through one of $\prod_{s'=s_0}^{s} d_{s'} - 1$ possible nodes at the depth s on the way down. From each node of depth s, the path can further go to one of $\prod_{s'=s+1}^{D-1} d_{s'}$ leaves on the way down. Therefore, there are $\left(\prod_{s'=s_0}^{s} d_{s'} - 1\right)\left(\prod_{s'=s+1}^{D-1} d_{s'}\right)$ paths that contain an edge of depth s.

Next, we rewrite $\sum_{i \in B(j)} \rho(i,j)$ using the indicator notation and compute the sum:

$$\sum_{i \in B(j)} \rho(i,j) = \sum_{s=s_0}^{D-1} \sum_{i \in B(j)} \alpha_s [\text{path from } j \text{ to } i \text{ contains an edge of depth } s] \tag{46}$$

$$= \sum_{s=s_0}^{D-1} \alpha_s \left(\prod_{s'=s_0}^{s} d_{s'} - 1\right)\left(\prod_{s'=s+1}^{D-1} d_{s'}\right). \tag{47}$$

Since the number of leaves in a block is $\prod_{s'=s_0}^{D-1} d_{s'}$, we have

$$\bar{\rho}_{s_0} = \tfrac{1}{|B(j)|} \sum_{i \in B(j)} \rho(i,j) = \sum_{s=s_0}^{D-1} \alpha_s \tfrac{(\prod_{s'=s_0}^{s} d_{s'}) - 1}{\prod_{s'=s_0}^{s} d_{s'}}, \tag{48}$$

which finishes the proof. □

Note that for tree-depth $D = 2$ the minimum ν_1 equals $\alpha_0 + \alpha_1 = 1$. As a result, our calibration function lower bound coincides with the exact calibration function from [14].

D Derivations for the Mean Average Precision Loss

In this section, we prove several statements about F_{sort} and L_{mAP}, which are used in Section 5.2.

Lemma 11. *The matrix $F_{\text{sort}}^{\mathsf{T}} F_{\text{sort}}$ has the following form:*

$$(F_{\text{sort}}^{\mathsf{T}} F_{\text{sort}})_{pq} = \begin{cases} (r-1)! H_{r,2}, & p = q, \\ (r-2)!(H_{r,1}^2 - H_{r,2}), & p \neq q, \end{cases} \tag{49}$$

where $H_{n,m} := \sum_{k=1}^{n} \frac{1}{k^m}$ is the generalized harmonic number of order m of n. As a result, for distinct permutations π and ω, the square norm of the projection is equal to

$$\|P_{\mathcal{F}_{\text{sort}}} \Delta_{\pi\omega}\|_2^2 = \tfrac{1}{(r-2)!(rH_{r,2} - H_{r,1}^2)} \sum_{p=1}^{r} \left(\tfrac{1}{\pi(p)} - \tfrac{1}{\omega(p)}\right)^2. \tag{50}$$

The condition number $\kappa(F_{\text{sort}})$ equals $\frac{\sqrt{r-1} H_{r,1}}{\sqrt{rH_{r,2} - H_{r,1}^2}}$.

Proof. By definition, $(F_{\text{sort}}^{\mathsf{T}} F_{\text{sort}})_{pq} = \sum_{\sigma \in S_r} \frac{1}{\sigma(p)\sigma(q)}$. We can rewrite the sum as the sum over the permutations with fixed values $\sigma(p)$ and $\sigma(q)$ and then sum over the fixed values. Therefore, the sum is equal to $(r-1)! H_{r,2}$ when $p = q$ and is equal to $(r-2)!(H_{r,1}^2 - H_{r,2})$ otherwise.

We now have $F_{\text{sort}}^{\mathsf{T}} F_{\text{sort}} = (r-2)!(rH_{r,2} - H_{r,1}^2)\mathbf{I}_r + (r-2)!(H_{r,1}^2 - H_{r,2})\mathbf{1}\mathbf{1}^{\mathsf{T}}$. The Sherman-Woodbury formula for the matrix inversion gives us the sum of the scalar matrix $\frac{1}{(r-2)!(rH_{r,2} - H_{r,1})}\mathbf{I}_r$ and the constant matrix. Since $\mathbf{1}^{\mathsf{T}} F_{\text{sort}}^{\mathsf{T}} \Delta_{\pi\omega} = \sum_{p=1}^{r} \left(\frac{1}{\pi(p)} - \frac{1}{\omega(p)}\right) = 0$, the square norm of the projection equals $\frac{1}{(r-2)!(rH_{r,2} - H_{r,1}^2)} \Delta_{\pi\omega}^{\mathsf{T}} F_{\text{sort}} F_{\text{sort}}^{\mathsf{T}} \Delta_{\pi\omega} = \frac{1}{(r-2)!(rH_{r,2} - H_{r,1}^2)} \sum_{p=1}^{r} \left(\frac{1}{\pi(p)} - \frac{1}{\omega(p)}\right)^2$.

The condition number of F_{sort} equals the square root of the ratio between the maximal and minimal eigenvalues of $F_{\text{sort}}^{\mathsf{T}}F_{\text{sort}}$. Subtracting $(r-2)!(rH_{r,2}-H_{r,1}^2)\mathbf{I}_r$ from $F_{\text{sort}}^{\mathsf{T}}F_{\text{sort}}$ we get a matrix of rank 1, which means that $r-1$ eigenvalues of $F_{\text{sort}}^{\mathsf{T}}F_{\text{sort}}$ equal $(r-2)!(rH_{r,2}-H_{r,1}^2)$. The remaining eigenvalue corresponds to the eigenvector $\mathbf{1}$ and equals $(r-1)!H_{r,1}^2$. With these eigenvalues we get the condition number $\kappa(F_{\text{sort}})=\frac{\sqrt{r-1}H_{r,1}}{\sqrt{rH_{r,2}-H_{r,1}^2}}$. □

Lemma 12. *The matrix $L_{\text{mAP}}^{\mathsf{T}}F_{\text{sort}}$ has the following form:*

$$\left(L_{\text{mAP}}^{\mathsf{T}}F_{\text{sort}}\right)_{y,p}=\begin{cases}\alpha(|\boldsymbol{y}|), & y_p=1,\\ \beta(|\boldsymbol{y}|), & y_p=0.\end{cases} \tag{51}$$

That is, for each ground-truth value $\boldsymbol{y}$ the matrix row components have only two values that depend on the Hamming norm $|\boldsymbol{y}|:=\sum_{p=1}^{r}y_p$, specifically:

$$\alpha(|\boldsymbol{y}|)=\mathfrak{A}_r\left(1-\tfrac{|\boldsymbol{y}|-1}{r-2}\left(1-\tfrac{r}{|\boldsymbol{y}|(r-1)}\right)\right)-\mathfrak{B}_r\left(\tfrac{3}{2}\tfrac{|\boldsymbol{y}|-1}{r-2}\tfrac{r-|\boldsymbol{y}|}{|\boldsymbol{y}|}\right)-\mathfrak{C}_r\left(\tfrac{r-|\boldsymbol{y}|}{|\boldsymbol{y}|(r-1)}\right), \tag{52}$$

$$\beta(|\boldsymbol{y}|)=\mathfrak{A}_r\left(1-\tfrac{|\boldsymbol{y}|-1}{r-2}\right)-\mathfrak{B}_r\left(1-\tfrac{3}{2}\tfrac{|\boldsymbol{y}|-1}{r-2}\right). \tag{53}$$

Here $\mathfrak{A}_r=(r-1)!H_{r,1}$, $\mathfrak{B}_r=(r-2)!(H_{r,1}^2-H_{r,2})$, $\mathfrak{C}_r=(r-1)!H_{r,2}$. As a result, for permutations π and ω, we obtain

$$\left(L_{\text{mAP}}^{\mathsf{T}}P_{\mathcal{F}_{\text{sort}}}\Delta_{\pi\omega}\right)_{\boldsymbol{y}}=\gamma(|\boldsymbol{y}|)\left((F_{\text{sort}}\boldsymbol{y})_\pi-(F_{\text{sort}}\boldsymbol{y})_\omega\right), \tag{54}$$

where $\gamma(p)=\frac{\alpha(p)-\beta(p)}{(r-2)!\left(rH_{r,2}-H_{r,1}^2\right)}$.

Proof. For brevity, here we denote F_{sort} by F, $\mathcal{F}_{\text{sort}}$ by $\mathcal{F}$ and L_{mAP} by L. Following the definitions of L and F, we explicitly compute the components of $L^{\mathsf{T}}F$:

$$\left(L^{\mathsf{T}}F\right)_{\boldsymbol{y},s}=\sum_{\sigma\in S_r}\left(1-\tfrac{1}{\boldsymbol{y}}\sum_{p=1}^{r}\sum_{q=1}^{p}\tfrac{y_{\sigma^{-1}(p)}y_{\sigma^{-1}(q)}}{p}\right)\tfrac{1}{\sigma(s)}. \tag{55}$$

There are exactly $(r-1)!$ permutations with one fixed element, so we have $\sum_{\sigma\in S_r}\frac{1}{\sigma(s)}=(r-1)!\sum_{p=1}^{r}\frac{1}{p}=(r-1)!H_{r,1}=:\mathfrak{A}_r$. To compute the remaining part of (55), we group the permutation values $\sigma(k)=t$ by each $t=1,\dots,r$ and move the sum over permutations inside the bracket:

$$-\tfrac{1}{\boldsymbol{y}}\sum_{\sigma\in S_r}\sum_{p=1}^{r}\sum_{q=1}^{p}\tfrac{y_{\sigma^{-1}(p)}y_{\sigma^{-1}(q)}}{p\sigma(s)}=-\tfrac{1}{\boldsymbol{y}}\sum_{t=1}^{r}\sum_{p=1}^{r}\sum_{q=1}^{p}\sum_{\sigma\in S_r,\sigma(s)=t}\tfrac{y_{\sigma^{-1}(p)}y_{\sigma^{-1}(q)}}{pt}. \tag{56}$$

Next, we compute the inner sum $\sum_{\sigma\in S_r,\sigma(s)=t}y_{\sigma^{-1}(p)}y_{\sigma^{-1}(q)}$. We rewrite the sum as the sum over inverse permutations:

$$\sum_{\sigma\in S_r,\sigma(s)=t}y_{\sigma^{-1}(p)}y_{\sigma^{-1}(q)}=\sum_{\pi\in S_r,\pi(t)=s}y_{\pi(p)}y_{\pi(q)}. \tag{57}$$

The number of positive terms is different for the two cases of $y_s=0$ and $y_s=1$. For $y_s=0$, using the Iverson brackets the sum can be rewritten as follows:

$$\sum_{\pi\in S_r,\pi(t)=s}y_{\pi(p)}y_{\pi(q)}=[p\neq t]\left([q<p\ \&\ q\neq t]|\boldsymbol{y}|(|\boldsymbol{y}|-1)(r-3)!+[q=p]|\boldsymbol{y}|(r-2)!\right). \tag{58}$$

We then sum the expression over $q=1,\dots,p$:

$$\sum_{q=1}^{p}[p\neq t]\left([q<p,q\neq t]|\boldsymbol{y}|(|\boldsymbol{y}|-1)(r-3)!+[q=p]|\boldsymbol{y}|(r-2)!\right)= \tag{59}$$

$$[p\neq t]\left((p-1-[t<p])|\boldsymbol{y}|(|\boldsymbol{y}|-1)(r-3)!+|\boldsymbol{y}|(r-2)!\right). \tag{60}$$

Finally, we multiply the expression by $\frac{-1}{|\boldsymbol{y}|pt}$ and compute the sum over p and t:

$$-\tfrac{1}{|\boldsymbol{y}|}\sum_{t=1}^{r}\sum_{p=1}^{r}\tfrac{1}{tp}[p\neq t]\left((p-1-[t<p])|\boldsymbol{y}|(|\boldsymbol{y}|-1)(r-3)!+|\boldsymbol{y}|(r-2)!\right)= \tag{61}$$

$$-\mathfrak{A}_r\tfrac{|\boldsymbol{y}|-1}{r-2}+\mathfrak{B}_r\left(1-\tfrac{3}{2}\tfrac{|\boldsymbol{y}|-1}{r-2}\right). \tag{62}$$

Combining with $\sum_{\sigma\in S_r}\frac{1}{\sigma(s)}=\mathfrak{A}_r$ we obtain the desired expression for $\beta(|\boldsymbol{y}|)$.

Now, consider the case of $y_s=1$. Again, we rewrite the sum as

$$\sum_{\pi\in S_r,\pi(t)=s}y_{\pi(p)}y_{\pi(q)}=[p=t]\left([q<p](|\boldsymbol{y}|-1)(r-2)!+[q=p](r-1)!\right) \tag{63}$$

$$+[p\neq t]\left([q<p\ \&\ q=t](|\boldsymbol{y}|-1)(r-2)!\right) \tag{64}$$

$$+[p\neq t]\left([q<p\ \&\ q\neq t](|\boldsymbol{y}|-1)(|\boldsymbol{y}|-2)(r-3)!\right) \tag{65}$$

$$+[p\neq t]\left([q=p](|\boldsymbol{y}|-1)(r-2)!\right), \tag{66}$$

sum it over q,

$$\sum_{q=1}^{p}\sum_{\pi\in S_r,\pi(t)=s}y_{\pi(p)}y_{\pi(q)}=[p=t]\left((p-1)(|\boldsymbol{y}|-1)(r-2)!+(r-1)!\right) \tag{67}$$

$$+[p\neq t]\left([t<p](|\boldsymbol{y}|-1)(r-2)!\right) \tag{68}$$

$$+[p\neq t]\left((p-1-[t<q])(|\boldsymbol{y}|-1)(|\boldsymbol{y}|-2)(r-3)!\right) \tag{69}$$

$$+[p\neq t](|\boldsymbol{y}|-1)(r-2)!, \tag{70}$$

and obtain the desired expression by multiplying the latter by $\frac{-1}{|\boldsymbol{y}|pt}$ and summing over p and t. The last step of the computation is completely analogous to the case $y_k=0$ and we omit it for brevity.

To compute $\left(L^TP_{\mathcal{F}}\Delta_{\pi\omega}\right)_{\boldsymbol{y}}$, we note that for all permutations π and ω it holds that $\mathbf{1}^\mathsf{T}F^\mathsf{T}\Delta_{\pi\omega}=0$. According to Lemma 11 and the Sherman-Woodbury formula, $(F^\mathsf{T}F)^{-1}$ is the sum of the scalar matrix $\frac{\mathbf{I}_r}{(r-2)!(rH_{r,2}-H_{r,1}^2)}$ and the multiple of the rank one matrix $\mathbf{1}\mathbf{1}^\mathsf{T}$. After the multiplication on $F^\mathsf{T}\Delta_{\pi\omega}$, the second term vanishes, so we get $(F^\mathsf{T}F)^{-1}F^\mathsf{T}\Delta_{\pi\omega}=\frac{\sum_{p=1}^{r}\frac{1}{\pi(p)}-\frac{1}{\omega(p)}}{(r-2)!(rH_{r,2}-H_{r,1}^2)}$. Finally, we rewrite $\left(L^TF\right)_{\boldsymbol{y},:}$ as $(\alpha(|\boldsymbol{y}|)-\beta(|\boldsymbol{y}|))y+\beta(|\boldsymbol{y}|)\mathbf{1}$. By the same argument, after the vector multiplication, the second component vanishes and we get $\left(L^\mathsf{T}F\left(F^\mathsf{T}F\right)^{-1}F\Delta_{\pi\omega}\right)_{\boldsymbol{y}}=\frac{(F\boldsymbol{y})\pi-(F\boldsymbol{y})\omega}{(r-2)!(rH_{r,2}-H_{r,1}^2)}$, which finishes the proof. □

Lemma 13. *For the score set $\mathcal{F}_{\text{sort}}$, we have $2(r-1)!\|P_{\mathcal{F}_{\text{sort}}}\Delta_{\pi\omega}\|_2^2=O(r)$. We also have that $\gamma(|\boldsymbol{y}|)$ defined in Lemma 12 with $|\boldsymbol{y}|=\lambda r,\lambda\in(0,1)$ vanishes as r approaches infinity: $\gamma(|\boldsymbol{y}|)=O(\frac{\log^2 r}{r})$. The condition number $\kappa(\mathcal{F}_{\text{sort}})$ grows as $\Theta(\log r)$.*

Proof. To derive an asymptotic bound for $\|P_{\mathcal{F}_{\text{sort}}}\Delta_{\pi\omega}\|_2^2$, we elaborate on the sum of squares $\sum_{p=1}^{r}\left(\frac{1}{\pi(p)}-\frac{1}{\omega(p)}\right)^2=2H_{r,2}-2\sum_{p=1}^{r}\frac{1}{\pi(p)\omega(p)}\leq 2H_{r,2}$ and apply the asymptotic bounds for the harmonic numbers $H_{r,1}^2=\Theta(\log^2 r), H_{r,2}=\Theta(1)$:

$$2(r-1)!\|P_{\mathcal{F}_{\text{sort}}}\Delta_{\pi\omega}\|_2^2=O(\tfrac{(r)!}{(r-2)!r})=O(r) \tag{71}$$

For the second part of the lemma, we rewrite $\alpha(|\boldsymbol{y}|)$ and $\beta(|\boldsymbol{y}|)$:

$$\alpha(|\boldsymbol{y}|)=\mathfrak{A}_r\left(1-\lambda(1-\tfrac{1}{r})\right)-\mathfrak{B}_r\tfrac{3}{2}(1-\lambda)-\mathfrak{C}_r\tfrac{1-\lambda}{r}+o(1) \tag{72}$$

$$\beta(|\boldsymbol{y}|)=\mathfrak{A}_r\left(1-\lambda\right)-\mathfrak{B}_r\left(1-\tfrac{3}{2}\lambda\right)+o(1) \tag{73}$$

$$\alpha(|\boldsymbol{y}|)-\beta(|\boldsymbol{y}|)=\mathfrak{A}_r\tfrac{1}{r}-\mathfrak{B}_r\tfrac{1}{2}-\mathfrak{C}_r\tfrac{1-\lambda}{r}+o(1) \tag{74}$$

By definition, we have $\mathfrak{A}_r=\Theta((r-1)!\log r)$, $\mathfrak{B}_r=\Theta((r-2)!\log^2 r)$, $\mathfrak{C}_r=\Theta((r-1)!)$, which gives us

$$\gamma(|\boldsymbol{y}|)=\tfrac{\alpha(|\boldsymbol{y}|)-\beta(|\boldsymbol{y}|)}{(r-2)!(rH_{r,2}-H_{r,1}^2)}=O\left(\tfrac{(r-2)!\log^2 r}{(r-1)!}\right)=O(\tfrac{\log^2 r}{r}), \tag{75}$$

what was to be shown.

Finally, the asymptotic bound for the condition number of $\mathcal{F}_{\text{sort}}$ trivially follows from its exact expression in Lemma 11 and the asymptotic bounds for the harmonic numbers. □

Appendix C Article. The Deep Weight Prior

Authors. Andrei Atanov, Arsenii Ashukha, Kirill Struminsky, Dmitriy Vetrov, Max Welling
Published in: Proceedings of the 7th International Conference on Learning Representations (ICLR 2019). ICLR, 2019. P. 1-17
Abstract. Bayesian inference is known to provide a general framework for incorporating prior knowledge or specific properties into machine learning models via carefully choosing a prior distribution. In this work, we propose a new type of prior distributions for convolutional neural networks, deep weight prior (DWP), that exploit generative models to encourage a specific structure of trained convolutional filters e.g., spatial correlations of weights. We define DWP in the form of an implicit distribution and propose a method for variational inference with such type of implicit priors. In experiments, we show that DWP improves the performance of Bayesian neural networks when training data are limited, and initialization of weights with samples from DWP accelerates training of conventional convolutional neural networks.

THE DEEP WEIGHT PRIOR

Andrei Atanov*
Samsung-HSE Laboratory, National Research University Higher School of Economics
ai.atanow@gmail.com

Arsenii Ashukha*
Samsung AI Center Moscow
ars.ashuha@gmail.com

Kirill Struminsky
Skolkovo Institute of Science and Technology
National Research University
Higher School of Economics
k.struminsky@gmail.com

Dmitry Vetrov
Samsung AI Center Moscow
Samsung-HSE Laboratory, National Research University Higher School of Economics
vetrovd@yandex.ru

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research
m.welling@uva.nl

ABSTRACT

Bayesian inference is known to provide a general framework for incorporating prior knowledge or specific properties into machine learning models via carefully choosing a prior distribution. In this work, we propose a new type of prior distributions for convolutional neural networks, *deep weight prior (dwp)*, that exploit generative models to encourage a specific structure of trained convolutional filters e.g., spatial correlations. We define *dwp* in a form of an implicit distribution and propose a method for variational inference with such type of implicit priors. In experiments, we show that *dwp* improves the performance of Bayesian neural networks when training data are limited, and initialization of weights with samples from *dwp* accelerates training of conventional convolutional neural networks.

1 INTRODUCTION

Bayesian inference is a tool that, after observing training data, allows to transforms a prior distribution over parameters of a machine learning model to a posterior distribution. Recently, stochastic variational inference (Hoffman et al., 2013) – a method for approximate Bayesian inference – has been successfully adopted to obtain a *variational approximation* of a posterior distribution over weights of a deep neural network (Kingma et al., 2015). Currently, there are two major directions for the development of Bayesian deep learning. The first direction can be summarized as the improvement of approximate inference with richer variational approximations and tighter variational bounds (Dikmen et al., 2015). The second direction is the design of probabilistic models, in particular, prior distributions, that widen the scope of applicability of the Bayesian approach.

Prior distributions play an important role for sparsification (Molchanov et al., 2017; Neklyudov et al., 2017), quantization (Ullrich et al., 2017) and compression (Louizos et al., 2017; Federici et al., 2017) of deep learning models. Although these prior distributions proved to be helpful, they are limited to fully-factorized structure. Thus, the often observed spatial structure of convolutional filters cannot be enforced with such priors. Convolutional neural networks are an example of the model family, where a correlation of the weights plays an important role, thus it may benefit from more flexible prior distributions.

Convolutional neural networks are known to learn similar convolutional kernels on different datasets from similar domains (Sharif Razavian et al., 2014; Yosinski et al., 2014). Based on this fact, within a specific data domain, we consider a distribution of convolution kernels of trained convolutional

*Equal contribution

networks. In the rest of the paper, we refer to this distribution as the *source kernel distribution*. Our main assumption is that within a specific domain the source kernel distribution can be efficiently approximated with convolutional kernels of models that were trained on a small subset of problems from this domain. For example, given a specific architecture, we expect that kernels of a model trained on notMNIST dataset – a dataset of grayscale images – come from the same distribution as kernels of the model trained on MNIST dataset. In this work, we propose a method that estimates the source kernel distribution in an implicit form and allows us to perform variational inference with the specific type of implicit priors.

Our contributions can be summarized as follows:

1. We propose *deep weight prior*, a framework that approximates the *source kernel distribution* and incorporates prior knowledge about the structure of convolutional filters into the prior distribution. We also propose to use an implicit form of this prior (Section 3.1).
2. We develop a method for variational inference with the proposed type of implicit priors (Section 3.2).
3. In experiments (Section 4), we show that variational inference with *deep weight prior* significantly improves classification performance upon a number of popular prior distributions in the case of limited training data. We also find that initialization of conventional convolution networks with samples from a deep weight prior leads to faster convergence and better feature extraction without training i.e., using random weights.

2 DEEP BAYES

In Bayesian setting, after observing a dataset $\mathcal{D} = \{x_1, \dots, x_N\}$ of N points, the goal is to transform our prior knowledge $p(\omega)$ of the unobserved distribution parameters ω to the posterior distribution $p(\omega \mid \mathcal{D})$. However, computing the posterior distribution through Bayes rule $p(\omega \mid \mathcal{D}) = p(\mathcal{D} \mid \omega)p(\omega)/p(\mathcal{D})$ may involve computationally intractable integrals. This problem, nonetheless, can be solved approximately.

Variational Inference (Jordan et al., 1999) is one of such approximation methods. It reduces the inference to an optimization problem, where we optimize parameters θ of a variational approximation $q_\theta(\omega)$, so that KL-divergence between $q_\theta(\omega)$ and $p(\omega \mid \mathcal{D})$ is minimized. This divergence in practice is minimized by maximizing the *variational lower bound* $\mathcal{L}(\theta)$ of the marginal log-likelihood of the data w.r.t parameters θ of the variational approximation $q_\theta(W)$.

$$\mathcal{L}(\theta) = L_{\mathcal{D}} - D_{\mathrm{KL}}(q_\theta(\omega) \| p(\omega)) \to \max_\theta \tag{1}$$

$$\text{where} \quad L_{\mathcal{D}} = \mathbb{E}_{q_\theta(\omega)} \log p(D \mid \omega) \tag{2}$$

The variational lower bound $\mathcal{L}(\theta)$ consists of two terms: 1) the (conditional) *expected log likelihood* $L_{\mathcal{D}}$, and 2) the regularizer $D_{\mathrm{KL}}(q_\theta(\omega) \| p(\omega))$. Since $\log p(\mathcal{D}) = \mathcal{L}(\theta) + D_{\mathrm{KL}}(q_\theta(\omega) \| p(\omega \mid \mathcal{D}))$ and $p(\mathcal{D})$ does not depend on $q_\theta(w)$ maximizing of $\mathcal{L}(\theta)$ minimizes $D_{\mathrm{KL}}(q_\theta(\omega) \| p(\omega \mid \mathcal{D}))$. However, in case of intractable expectations in equation 1 neither the variational lower bound $\mathcal{L}(\theta)$ nor its gradients can be computed in a closed form.

Recently, Kingma & Welling (2013) and Rezende et al. (2014) proposed an efficient mini-batch based approach to stochastic variational inference, so-called *stochastic gradient variational Bayes* or *doubly stochastic variational inference*. The idea behind this framework is reparamtetrization, that represents samples from a parametric distribution $q_\theta(\omega)$ as a deterministic differentiable function $\omega = f(\theta, \epsilon)$ of parameters θ and an (auxiliary) noise variable $\epsilon \sim p(\epsilon)$. Using this trick we can efficiently compute an unbiased stochastic gradient $\nabla_\theta \mathcal{L}$ of the variational lower bound w.r.t the parameters of the variational approximation.

Bayesian Neural Networks. The stochastic gradient variational Bayes framework has been applied to approximate posterior distributions over parameters of deep neural networks (Kingma et al., 2015). We consider a discriminative problem, where dataset $\mathcal{D}$ consists of N object-label pairs $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$. For this problem we maximize the variational lower bound $\mathcal{L}(\theta)$ with respect to

parameters θ of a variational approximation $q_\theta(W)$:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \mathbb{E}_{q_\theta(W)} \log p(y_i \mid x_i, W) - D_{\mathrm{KL}}(q_\theta(W) \| p(W)) \to \max_\theta \tag{3}$$

where W denotes weights of a neural network, $q_\theta(W)$ is a variational distribution, that allows reparametrization (Kingma & Welling, 2013; Figurnov et al., 2018) and $p(W)$ is a prior distribution. In the simplest case $q_\theta(W)$ can be a fully-factorized normal distribution. However, more expressive variational approximations may lead to better quality of variational inference (Louizos & Welling, 2017; Yin & Zhou, 2018). Typically, Bayesian neural networks use fully-factorized normal or log-uniform priors (Kingma et al., 2015; Molchanov et al., 2017; Louizos & Welling, 2017).

Variational Auto-encoder. Stochastic gradient variational Bayes has also been applied for building generative models. The variational auto-encoder proposed by Kingma & Welling (2013) maximizes a variational lower bound $\mathcal{L}(\theta, \phi)$ on the marginal log-likelihood by amortized variational inference:

$$\mathcal{L}(\theta, \phi) = \sum_{i=1}^{N} \mathbb{E}_{q_\theta(z_i \mid x_i)} \log p_\phi(x_i \mid z_i) - D_{\mathrm{KL}}(q_\theta(z_i \mid x_i) \| p(z_i)) \to \max_{\theta, \phi}, \tag{4}$$

where an *inference model* $q_\theta(z_i \mid x_i)$ approximates the posterior distribution over local latent variables z_i, *reconstruction model* $p_\phi(x_i \mid z_i)$ transforms the distribution over latent variables to a conditional distribution in object space and a prior distribution over latent variables $p(z_i)$. The vanilla VAE defines $q_\theta(z \mid x)$, $p_\phi(x \mid z)$, $p(z)$ as fully-factorized distributions, however, a number of richer variational approximations and prior distributions have been proposed (Rezende & Mohamed, 2015; Kingma et al., 2016; Tomczak & Welling, 2017). The approximation of the data distribution can then be defined as an intractable integral $p(x) \approx \int p_\phi(x \mid z) p(z)\, dz$ which we will refer to as an implicit distribution.

3 Deep Weight Prior

In this section, we introduce the *deep weight prior* – an expressive prior distribution that is based on generative models. This prior distribution allows us to encode and favor the structure of learned convolutional filters. We consider a neural network with L convolutional layers and denote parameters of l-th convolutional layer as $w^l \in \mathbb{R}^{I_l \times O_l \times H_l \times W_l}$, where I_l is the number of input channels, O_l is the number of output channels, H_l and W_l are spatial dimensions of kernels. Parameters of the neural network are denoted as $W = (w^1, \dots w^L)$. A variational approximation $q_\theta(W)$ and a prior distribution $p(W)$ have the following factorization over layers, filters and channels:

$$q_\theta(W) = \prod_{l=1}^{L} \prod_{i=1}^{I_l} \prod_{j=1}^{O_l} q(w_{ij}^l \mid \theta_{ij}^l) \qquad p(W) = \prod_{l=1}^{L} \prod_{i=1}^{I_l} \prod_{j=1}^{O_l} p_l(w_{ij}^l), \tag{5}$$

where $w_{ij}^l \in \mathbb{R}^{H_l \times W_l}$ is a kernel of j-th channel in i-th filter of l-th convolutional layer. We also assume that $q_\theta(W)$ allows reparametrization. The prior distribution $p(W)$, in contrast to popular prior distributions, is not factorized over spatial dimensions of the filters H_l, W_l.

For a specific data domain and architecture, we define the *source kernel distribution* – the distribution of trained convolutional kernels of the l-th convolutional layer. The source kernel distribution favors learned kernels, and thus it is a very natural candidate to be the prior distribution $p_l(w_{ij}^l)$ for convolutional kernels of the l-th layer. Unfortunately, we do not have access to its probability density function (p.d.f.), that is needed for most approximate inference methods e.g., variational inference. Therefore, we assume that the p.d.f. of the source kernel distribution can be approximated using kernels of models trained on external datasets from the same domain. For example, given a specific architecture, we expect that kernels of a model trained on CIFAR-100 dataset come from the same distribution as kernels of the model trained on CIFAR-10 dataset. In other words, the p.d.f. of the source kernel distribution can be approximated using a small subset of problems from a specific data domain. In the next subsection, we propose to approximate this intractable probability density function of the source kernel distribution using the framework of generative models.

Algorithm 1 Stochastic Variational Inference With Implicit Prior Distribution

Require: dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$
Require: variational approximations $q(w \mid \theta_{ij}^l)$ and reverse models $r(z \mid w; \psi_l)$
Require: reconstruction models $p(w \mid z; \phi_l)$, priors for auxiliary variables $p_l(z)$
while not converged **do**
  $\hat{M} \leftarrow$ mini-batch of objects form dataset $\mathcal{D}$
  $\hat{w}_{ij}^l \leftarrow$ sample weights from $q(w|\theta_{ij}^l)$ with reparametrization
  $\hat{z}_{ij}^l \leftarrow$ sample auxiliary variables from $r(z \mid \hat{w}_{ij}^l; \psi_l)$ with reparametrization
  $\hat{\mathcal{L}}^{aux} \leftarrow L_{\hat{M}} + \sum_{l,i,j} - \log q(\hat{w}_{ij}^l \mid \theta_{ij}^l) - \log r(\hat{z}_{ij}^l \mid \hat{w}_{ij}^l; \psi_l) + \log p_l(\hat{z}_{ij}^l) + \log p(\hat{w}_{ij}^l \mid \hat{z}_{ij}^l; \phi_l)$
  Obtain unbiased estimate $\hat{g}$ with $\mathbb{E}[\hat{g}] = \nabla \mathcal{L}^{aux}$ by differentiating $\hat{\mathcal{L}}^{aux}$
  Update parameters θ and ψ using gradient $\hat{g}$ and a stochastic optimization algorithm
end while
return Parameters θ, ψ

3.1 Model of Prior Distribution

In this section, we discuss explicit and implicit approximations $\hat{p}_l(w)$ of the probability density function $p_l(w)$ of the source kernel distribution of l-th layer. We assume to have a trained convolutional neural network, and treat kernels from the l-th layer of this network $w_{ij}^l \in R^{H_l \times W_l}$ as samples from the source kernel distribution of l-th layer $p_l(w)$.

Explicit models. A number of approximations allow us to evaluate probability density functions explicitly. Such families include but are not limited to Kernel Density Estimation (Silverman, 1986), Normalizing Flows (Rezende & Mohamed, 2015; Dinh et al., 2017) and PixelCNN (van den Oord et al., 2016). For these families, we can estimate the KL-divergence $D_{KL}(q(w \mid \theta_{ij}^l) \| \hat{p}_l(w))$ and its gradients without a systematic bias, and then use them for variational inference. Despite the fact that these methods provide flexible approximations, they usually demand high memory or computational cost (Louizos & Welling, 2017).

Implicit models. Implicit models, in contrast, can be more computationally efficient, however, they do not provide access to an explicit form of probability density function $\hat{p}_l(w)$. We consider an approximation of the prior distribution $p_l(w)$ in the following implicit form:

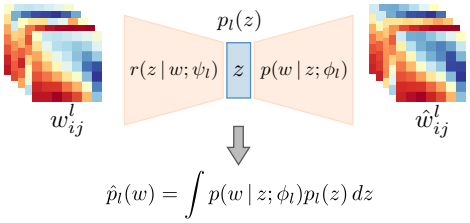
$$\hat{p}_l(w) = \int p(w \mid z; \phi_l) p_l(z)\, dz, \tag{6}$$

where a conditional distribution $p(w \mid z; \phi_l)$ is an explicit parametric distribution and $p_l(z)$ is an explicit prior distribution that does not depend on trainable parameters. Parameters of the conditional distribution $p(w \mid z; \phi_l)$ can be modeled by a differentiable function $g(z; \phi_l)$ e.g. neural network. Note, that while the conditional distribution $p(w \mid z; \phi_l)$ usually is a simple explicit distribution, e.g. fully-factorized Gaussian, the marginal distribution $\hat{p}_l(w)$ is generally a more complex intractable distribution.

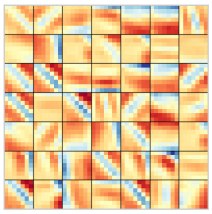
Parameters ϕ_l of the conditional distribution $p(w \mid z; \phi_l)$ can be fitted using the variational auto-encoder framework. In contrast to the methods with explicit access to the probability density, variational auto-encoders combine low memory cost and fast sampling. However, we cannot obtain an unbiased estimate the logarithm of probability density function $\log \hat{p}_l(w)$ and therefore cannot build an unbiased estimator of the variational lower bound (equation 3). In order to overcome this limitation we propose a modification of variational inference for implicit prior distributions.

3.2 Variational Inference With Implicit Prior Distribution

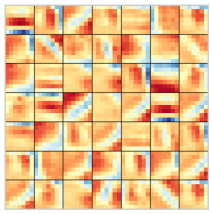
Stochastic variational inference approximates a true posterior distribution by maximizing the variational lower bound $\mathcal{L}(\theta)$ (equation 1), which includes the KL-divergence $D_{\mathrm{KL}}(q(W) \| p(W))$ between a variational approximation $q_\theta(W)$ and a prior distribution $p(W)$. In the case of simple prior and variational distributions (e.g. Gaussian), the KL-divergence can be computed in a closed form or unbiasedly estimated. Unfortunately, it does not hold anymore in case of an implicit prior distribution $\hat{p}(W) = \Pi_{l,i,j} \hat{p}_l(w_{ij}^l)$. In that case, the KL-divergence cannot be estimated without bias.



(a) Learning of DWP with VAE



(b) Learned filters



(c) Samples from DWP

Figure 1: At subfig. 1(a) we show the process of learning the prior distribution over kernels of one convolutional layer. First, we train encoder $r(z \mid w; \phi_l)$ and decoder $p(w \mid z; \psi_l)$ with VAE framework. Then, we use the decoder to construct the prior $\hat{p}_l(w)$. At subfig. 1(b) we show a batch of learned kernels of shape 7×7 form the first convolutional layer of a CNN trained on NotMNIST dataset, at subfig. 1(c) we show samples form the deep weight prior that is learned on these kernels.

To make the computation of the variational lower bound tractable, we introduce an auxiliary lower bound on the KL-divergence. KL-divergence:

$$D_{\mathrm{KL}}(q(W)\|\hat{p}(W)) = \sum_{l,i,j} D_{\mathrm{KL}}(q(w_{ij}^l|\theta_{ij}^l)\|\hat{p}_l(w_{ij}^l)) \leq \sum_{l,i,j} \big(- H(q(w_{ij}^l \mid \theta_{ij}^l)) + \\ + \mathbb{E}_{q(w_{ij}^l \mid \theta_{ij}^l)} \big[D_{\mathrm{KL}}(r(z \mid w_{ij}^l; \psi_l)\|p_l(z)) - \mathbb{E}_{r(z \mid w_{ij}^l; \psi_l)} \log p(w_{ij}^l \mid z; \phi_l) \big] \big) = D_{\mathrm{KL}}^{bound}, \quad (7)$$

where $r(z \mid w; \psi_l)$ is an auxiliary inference model for the prior of l-th layer $\hat{p}_l(w)$, The final auxiliary variational lower bound has the following form:

$$\mathcal{L}^{aux}(\theta, \psi) = L_D - D_{\mathrm{KL}}^{bound} \leq L_D - D_{\mathrm{KL}}(q_\theta(W)\|\hat{p}(W)) = \mathcal{L}(\theta) \quad (8)$$

The lower bound $\mathcal{L}^{aux}$ is tight if and only if the KL-divergence between the auxiliary reverse model and the intractable posterior distribution over latent variables z given w is zero (Appendix A).

In the case when $q_\theta(w)$, $p(w \mid z; \phi_l)$ and $r(z \mid w; \psi_l)$ are explicit parametric distributions which can be reparametrized, we can perform an unbiased estimation of a gradient of the auxiliary variational lower bound $\mathcal{L}^{aux}(\theta, \psi)$ (equation 8) w.r.t. parameters θ of the variational approximation $q_\theta(W)$ and parameters ψ of the reverse models $r(z \mid w; \psi_l)$. Then we can maximize the auxiliary lower bound w.r.t. parameters of the variational approximation and the reversed models $\mathcal{L}^{aux}(\theta, \psi) \to \max_{\theta, \psi}$. Note, that parameters ϕ of the prior distribution $\hat{p}(W)$ are fixed during variational inference, in contrast to the Empirical Bayesian framework (MacKay, 1992).

Algorithm 1 describes stochastic variational inference with an implicit prior distribution. In the case when we can calculate an entropy $H(q)$ or the divergence $D_{\mathrm{KL}}(r(z \mid w; \psi_l)\|p_l(z))$ explicitly, the variance of the estimation of the gradient $\nabla \hat{\mathcal{L}}^{aux}(\theta, \psi)$ can be reduced. This algorithm can also be applied to an implicit prior that is defined in the form of Markov chain:

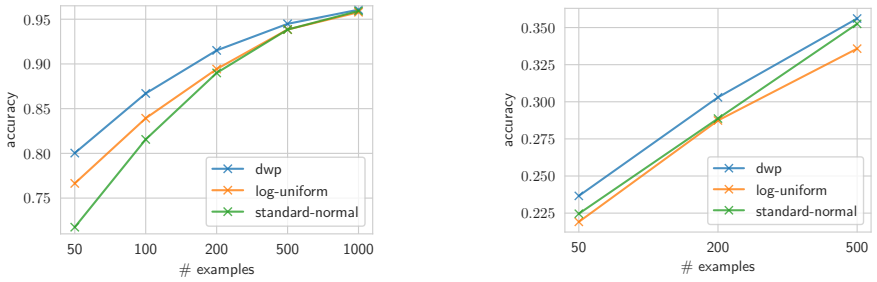
$$\hat{p}(w) = \int dz_0 \ldots dz_T p(w \mid z_T) p(z_0) \prod_{t=0}^{T-1} p(z_{t+1} \mid z_t), \quad (9)$$

where $p(z_{t+1} \mid z_t)$ is a transition operator (Salimans et al., 2015), see Appendix A. We provide more details related to the form of $p(w \mid z; \phi_l)$, $r(z \mid w; \psi_l)$ and $p_l(z)$ distributions in Section 4.

3.3 Learning Deep Weight Prior

In this subsection we explain how to train deep weight prior models for a particular problem. We present samples from learned prior distribution at Figure 1(c).

Source datasets of kernels. For kernels of a particular convolutional layer l, we train an individual prior distribution $\hat{p}_l(w) = \int p(w \mid z; \phi_l) p_l(z)\, dz$. First, we collect a source dataset of the kernels of the l-th layer of convolutional networks (source networks) trained on a dataset from a similar domain. Then, we train reconstruction models $p(w \mid z; \phi_l)$ on these collected source datasets for



(a) Results for MNIST

(b) Results for CIFAR-10

Figure 2: For different sizes of training set of MNIST and CIFAR-10 datasets, we demonstrate the performance of variational inference with a fully-factorized variational approximation with three different prior distributions: deep weight prior (dwp), log-uniform, and standard normal. We found that variational inference with a deep weight prior distribution achieves better mean test accuracy comparing to learning with standard normal and log-uniform prior distributions.

each layer, using the framework of variational auto-encoder (Section 2). Finally, we use the reconstruction models to construct priors $\hat{p}_l(w)$ as shown at Figure 1(a). In our experiments, we found that regularization is crucial for learning of source kernels. It helps to learn more structured and less noisy kernels. Thus, source models were learned with L_2 regularization. We removed kernels of small norm as they have no influence upon predictions (Molchanov et al., 2017), but they make learning of the generative model more challenging.

Reconstruction and inference models for prior distribution. In our experiments, inference models $r(z \mid w; \psi_l)$ are fully-factorized normal-distributions $\mathcal{N}(z \mid \mu_{\psi_l}(w), \text{diag}(\sigma^2_{\psi_l}(w)))$, where parameters $\mu_{\psi_l}(w)$ and $\sigma_{\psi_l}(w)$ are modeled by a convolutional neural network. The convolutional part of the network is constructed from several convolutional layers that are alternated with ELU (Clevert et al., 2015) and max-pooling layers. Convolution layers are followed by a fully-connected layer with $2 \cdot z^l_{dim}$ output neurons, where z^l_{dim} is a dimension of the latent representation z, and is specific for a particular layer.

Reconstruction models $p(w \mid z; \phi_l)$ are also modeled by a fully-factorized normal-distribution $\mathcal{N}(w \mid \mu_{\phi_l}(z), \text{diag}(\sigma^2_{\phi_l}(z)))$ and network for μ_{ϕ_l} and $\sigma^2_{\phi_l}$ has the similar architecture as the inference model, but uses transposed convolutions. We use the same architectures for all prior models, but with slightly different hyperparameters, due to different sizes of kernels. We also use fully-factorized standard Gaussian prior $p_l(z_i) = \mathcal{N}(z_i \mid 0, 1)$ for latent variables z_i. We provide a more detailed description at Appendix F.

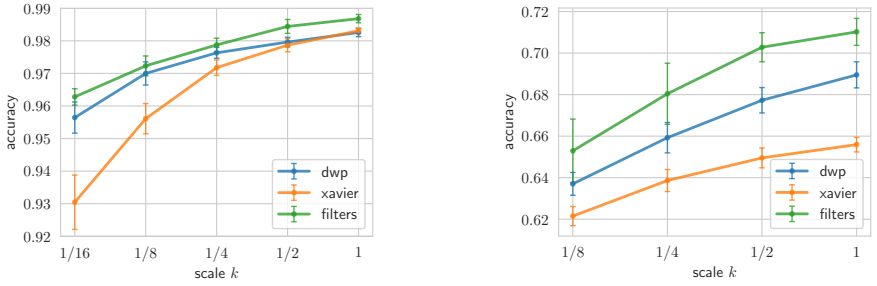
4 Experiments

We apply deep weight prior to variational inference, random feature extraction and initialization of convolutional neural networks. In our experiments we used MNIST (LeCun et al., 1998), NotMNIST (Bulatov, 2011), CIFAR-10 and CIFAR-100 (Krizhevsky & Hinton, 2009) datasets. Experiments were implemented[1] using PyTorch (Paszke et al., 2017). For optimization we used Adam (Kingma & Ba, 2014) with default hyperparameters. We trained prior distributions on a number of source networks which were learned from different initial points on NotMNIST and CIFAR-100 datasets for MNIST and CIFAR-10 experiments respectively.

4.1 Classification

In this experiment, we performed variational inference over weights of a discriminative convolutional neural network (Section 3) with three different prior distributions for the weights of the convolutional layers: deep weight prior (dwp), standard normal and log-uniform (Kingma et al., 2015). We did not perform variational inference over the parameters of the fully connected layers. We used

[1] The code is available at `https://github.com/bayesgroup/deep-weight-prior`



(a) Results for MNIST

(b) Results for CIFAR-10

Figure 3: We study the influence of initialization of convolutional filters on the performance of random feature extraction. In the experiment, the weights of convolutional filters were initialized randomly and fixed. The initializations were sampled from deep weight prior (dwp), learned filters (filters) and samples from Xavier distribution (xavier). We performed the experiment for different size of the model, namely, to obtain models of different sizes we scaled a number of filters in all convolutional layers linearly by k. For every size of the model, we averaged results by 10 runs. We found that initialization with samples from deep weight prior and learned filters significantly outperform Xavier initialization. Although, initialization with filters performs marginally better, *dwp* does not require to store a potentially big set of all learned filters. We present result for MNIST and CIFAR-10 datasets at sub figs. 3(a) and 3(b) respectively.

a fully-factorized variational approximation with additive parameterization proposed by Molchanov et al. (2017) and local reparametrization trick proposed by Kingma et al. (2015). Note, that our method can be combined with more complex variational approximations, in order to improve variational inference.

On MNIST dataset we used a neural network with two convolutional layers with $32, 128$ filters of shape 7×7, 5×5 respectively, followed by one linear layer with 10 neurons. On the CIFAR dataset we used a neural network with four convolutional layers with 128, 256, 256 filters of shape 7×7, 5×5, 5×5 respectively, followed by two fully connected layers with 512 and 10 neurons. We used a max-pooling layer (Nagi et al., 2011) After the first convolutional layer. All layers were divided with leaky ReLU nonlinearities (Nair & Hinton, 2010).

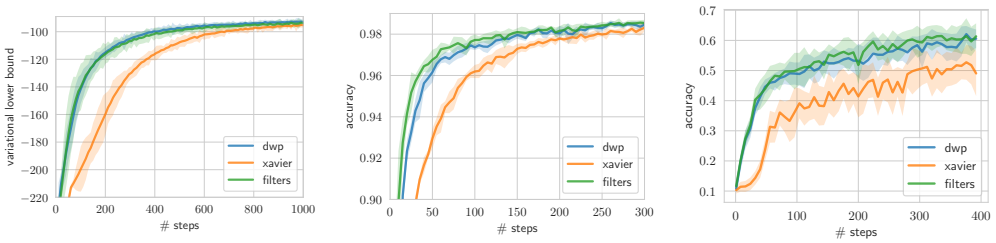
At figure 2 we report accuracy for variational inference with different sizes of training datasets and prior distributions. Variational inference with deep weight prior leads to better mean test accuracy, in comparison to log-uniform and standard normal prior distributions. Note that the difference gets more significant as the training set gets smaller.

4.2 Random Feature Extraction

Convolutional neural networks produce useful features even if they are initialized randomly (Saxe et al., 2011; He et al., 2016; Ulyanov et al., 2017). In this experiment, we study an influence of different random initializations of convolutional layers – that is fixed during training – on the performance of convolutional networks of different size, where we train only fully-connected layers. We use three initializations for weights of convolutional layers: learned kernels, samples from deep weight prior, samples from Xavier distribution (Glorot & Bengio, 2010). We use the same architectures as in Section 4.1. We found that initializations with samples from deep weight prior and learned kernels significantly outperform the standard Xavier initialization when the size of the network is small. Initializations with samples form deep weight prior and learned filters perform similarly, but with deep weight prior we can avoid storing all learned kernels. At Figure 3, we show results on MNIST and CIFAR-10 for different network sizes, which are obtained by scaling the number of filters by k.

4.3 Convergence

Deep learning models are sensitive to initialization of model weights. In particular, it may influence the speed of convergence or even a local minimum a model converges to. In this experiment, we study the influence of initialization on the convergence speed of two settings: a variational auto-



(a) VAE on MNIST

(b) ConvNet on MNIST

(c) ConvNet on CIFAR-10

Figure 4: We found that initialization of weights of the models with deep weight priors or learned filters significantly increases the training speed, comparing to Xavier initialization. At subplot 4(a) we report a variational lower bound for variational auto-encoder, at subplots 4(b) and 4(c) we report accuracy for convolution networks on MINTS and CIFAR-10.

encoder on MNIST, and convolutional networks on MNIST and CIFAR-10. We compare three different initializations of weights of conventional convolutional layers: learned filters, samples from deep weight prior and samples form Xavier distribution.

Figure 4 provides the results for a convolutional variational auto-encoder trained on MNIST and for a convolutional classification network trained on CIFAR-10 and MNIST. We found that deep weight prior and learned filters initializations perform similarly and lead to significantly faster convergence comparing to standard Xavier initialization. Deep weight prior initialization however does not require us to store a possibly large set of filters. Also, we plot samples from variational auto-encoders at a different training steps Appendix E.

5 Related Work

The recent success of transfer learning (Yosinski et al., 2014) shows that convolutional networks produce similar convolutional filters while being trained on different datasets from the same domain e.g. photo-realistic images. In contrast to Bayesian techniques (Kingma et al., 2015; Kochurov et al., 2018), these methods do not allow to obtain a posterior distribution over parameters of the model, and in most cases, they require to store convolutional weights of pre-trained models and careful tuning of hyperparameters.

The Bayesian approach provides a framework that incorporates prior knowledge about weights of a machine learning model by choosing or leaning a prior distribution $p(w)$. There is a huge amount of works on prior distributions for Bayesian inference (MacKay, 1992; Williams, 1995), where empirical Bayes – an approach that tunes parameters of the prior distribution on the training data – plays an important role (MacKay, 1992). These methods are widely used for regularization and sparsification of linear models (Bishop & Tipping, 2003), however, applied to deep neural networks (Kingma et al., 2015; Ullrich et al., 2017), they do not take into account the structure of the model weights, e.g. spatial correlations, which does matter in case of convolutional networks. Our approach allows to perform variational inference with an implicit prior distribution, that is based on previously observed convolutional kernels. In contrast to an empirical Bayes approach, parameters ϕ of a deep weight prior (equation 6) are adjusted before the variational inference and then remain fixed.

Prior to our work implicit models have been applied to variational inference. That type of models includes a number of flexible variational distributions e.g., semi-implicit (Yin & Zhou, 2018) and Markov chain (Salimans et al., 2015; Lamb et al., 2017) approximations. Implicit priors have been used for introducing invariance properties (Nalisnick & Smyth, 2018), improving uncertainty estimation (Ma et al., 2018) and learning meta-representations within an empirical Bayes approach (Karaletsos et al., 2018).

In this work, we propose to use an implicit prior distribution for stochastic variational inference (Kingma et al., 2015) and develop a method for variational inference with the specific type of implicit priors. The approach also can be generalized to prior distributions in the form of a Markov chain. We show how to use this framework to learn a flexible prior distribution over kernels of Bayesian convolutional neural networks.

6 DISCUSSION & CONCLUSION

In this work we propose *deep weight prior* – a framework for designing a prior distribution for convolutional neural networks, that exploits prior knowledge about the structure of learned convolutional filters. This framework opens a new direction for applications of Bayesian deep learning, in particular to transfer learning.

Factorization. The factorization of deep weight prior does not take into account inter-layer dependencies of the weights. Although a more complex factorization might be a better fit for CNNs. Accounting inter-layer dependencies may give us an opportunity to recover a distribution in the space of trained networks rather than in the space of trained kernels. However, estimating prior distributions of more complex factorization may require significantly more data and computational budget, thus the topic needs an additional investigation.

Inference. An alternative to variational inference with auxiliary variables (Salimans et al., 2015) is semi-implicit variational inference (Yin & Zhou, 2018). The method was developed only for semi-implicit variational approximations, and only the recent work on doubly semi-implicit variational inference generalized it for implicit prior distributions (Molchanov et al., 2018). These algorithms might provide a better way for variational inference with a deep weight prior, however, the topic needs further investigation.

ACKNOWLEDGMENTS

We would like to thank Ekaterina Lobacheva, Dmitry Molchanov, Kirill Neklyudov and Ivan Sosnovik for valuable discussions and feedback on the earliest version of this paper. Andrei Atanov was supported by Samsung Research, Samsung Electronics. Kirill Struminsky was supported by Ministry of Education and Science of the Russian Federation (grant 14.756.31.0001).

REFERENCES

Christopher M Bishop and Michael E Tipping. Bayesian regression and classification. *Nato Science Series sub Series III Computer And Systems Sciences*, 190:267–288, 2003.

Yaroslav Bulatov. Notmnist dataset. technical report. 2011. URL `http://yaroslavvb.blogspot.it/2011/09/notmnist-dataset.html`.

Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *CoRR*, abs/1509.00519, 2015. URL `http://arxiv.org/abs/1509.00519`.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Onur Dikmen, Zhirong Yang, and Erkki Oja. Learning the information divergence. *IEEE transactions on pattern analysis and machine intelligence*, 37(7):1442–1454, 2015.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *5th International Conference on Learning Representations*, 2017.

Marco Federici, Karen Ullrich, and Max Welling. Improved bayesian compression. *arXiv preprint arXiv:1711.06494*, 2017.

Michael Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit reparameterization gradients. *arXiv preprint arXiv:1805.08498*, 2018.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Kun He, Yan Wang, and John Hopcroft. A powerful generative model using random weights for the deep image representation. In *Advances in Neural Information Processing Systems*, pp. 631–639, 2016.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Theofanis Karaletsos, Peter Dayan, and Zoubin Ghahramani. Probabilistic meta-representations of neural networks. *arXiv preprint arXiv:1810.00555*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems 29*, pp. 4743–4751. 2016.

Max Kochurov, Timur Garipov, Dmitry Podoprikhin, Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Bayesian incremental learning for deep neural networks. *arXiv preprint arXiv:1802.07329*, 2018.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Alex M Lamb, Devon Hjelm, Yaroslav Ganin, Joseph Paul Cohen, Aaron C Courville, and Yoshua Bengio. Gibbsnet: Iterative adversarial inference for deep graphical models. In *Advances in Neural Information Processing Systems*, pp. 5089–5098, 2017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. *arXiv preprint arXiv:1703.01961*, 2017.

Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pp. 3288–3298, 2017.

Chao Ma, Yingzhen Li, and José Miguel Hernández-Lobato. Variational implicit processes. *arXiv preprint arXiv:1806.02390*, 2018.

David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2498–2507, 2017.

Dmitry Molchanov, Valery Kharitonov, Artem Sobolev, and Dmitry Vetrov. Doubly semi-implicit variational inference. *arXiv preprint arXiv:1810.02789*, 2018.

Jawad Nagi, Frederick Ducatelle, Gianni A Di Caro, Dan Cireşan, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jürgen Schmidhuber, and Luca Maria Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*, pp. 342–347. IEEE, 2011.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Eric Nalisnick and Padhraic Smyth. Learning priors for invariance. In *International Conference on Artificial Intelligence and Statistics*, pp. 366–375, 2018.

Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P Vetrov. Structured bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems*, pp. 6775–6784, 2017.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, pp. 1530–1538, 2015.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 1278–1286, 2014.

Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pp. 1218–1226, 2015.

Andrew M Saxe, Pang Wei Koh, Zhenghao Chen, Maneesh Bhand, Bipin Suresh, and Andrew Y Ng. On random weights and unsupervised feature learning. In *ICML*, pp. 1089–1096, 2011.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.

B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1986. ISBN 9780412246203. URL `https://books.google.ru/books?id=e-xsrjsL7WkC`.

Jakub M Tomczak and Max Welling. Vae with a vampprior. *arXiv preprint arXiv:1705.07120*, 2017.

Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. *arXiv:1711.10925*, 2017.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pp. 4790–4798, 2016.

Peter M Williams. Bayesian regularization and pruning using a laplace prior. *Neural computation*, 7(1):117–143, 1995.

Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5660–5669, 2018.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pp. 3320–3328, 2014.

A VARIATIONAL INFERENCE WITH IMPLICIT PRIOR DISTRIBUTION

We consider a variational lower bound $\mathcal{L}$ with variational approximation $q(w)$ and prior distribution defined in a form of Markov chain $p(w) = \int dz_0 \dots dz_T p(w \mid z_T) \prod_{t=0}^{T} p(z_{t+1} \mid z_t) p(z_0)$ and joint distribution $p(w, \boldsymbol{z}) = p(w \mid z_T) \prod_{t=0}^{T} p(z_{t+1} \mid z_t) p(z_0)$. Where $p(z_{t+1} \mid z_t)$ is a transition operator, and $\boldsymbol{z} = (z_0, \dots, z_T)$ (Salimans et al., 2015). Unfortunately, gradients of $\mathcal{L}$ cannot be efficiently estimated, but we construct a tractable lower bound $\mathcal{L}^{aux}$ for $\mathcal{L}$:

$$\mathcal{L} = \mathbb{E}_{q(w)} \left[\log p(x \mid w) p(w) - \log q(w)\right] = \mathbb{E}_{q(w)} \mathbb{E}_{r(\boldsymbol{z} \mid w)} \left[\log p(x \mid w) p(w) - \log q(w)\right] = \quad (10)$$

$$= \mathbb{E}_{q(w)} \mathbb{E}_{r(\boldsymbol{z} \mid w)} \left[\log p(x \mid w) \frac{p(w, \boldsymbol{z})}{p(\boldsymbol{z} \mid w)} \frac{r(\boldsymbol{z} \mid w)}{r(\boldsymbol{z} \mid w)} - \log q(w)\right] = \quad (11)$$

$$= \mathbb{E}_{q(w)} \mathbb{E}_{r(\boldsymbol{z} \mid w)} \left[\log p(x \mid w) \frac{p(w, \boldsymbol{z})}{r(\boldsymbol{z} \mid w)} - \log q(w)\right] + \mathbb{E}_{q(w)} D_{\mathrm{KL}}(r(\boldsymbol{z} \mid w) \| p(\boldsymbol{z} \mid w)) = \quad (12)$$

$$= \mathcal{L}^{aux} + \mathbb{E}_{q(w)} D_{\mathrm{KL}}(r(\boldsymbol{z} \mid w) \| p(\boldsymbol{z} \mid w)) \geq \mathcal{L}^{aux}. \quad (13)$$

Inequality 13 has a very natural interpretation. The lower bound $\mathcal{L}^{aux}$ is tight if and only if the KL-divergence between the auxiliary reverse model and the posterior intractable distribution $p(z \mid w)$ is zero.

The deep weight prior (Section 3) is a special of Markov chain prior for $T = 0$ and $p(w) = \int p(w \mid z) p(z) dz$. The auxiliary variational bound has the following form:

$$\mathcal{L}^{aux} = \mathbb{E}_{q(w)} \mathbb{E}_{r(z \mid w)} \left[\log p(x \mid w) \frac{p(w \mid z) p(z)}{r(z \mid w)} - \log q(w)\right] = \quad (14)$$

$$= \mathbb{E}_{q(w)} \left[\log p(x \mid w)\right] + H(q) - \mathbb{E}_{q(w)} \left[D_{\mathrm{KL}}(r(z \mid w) \| p(z) - \mathbb{E}_{r(z \mid w)} \log p(w \mid z))\right]. \quad (15)$$

where the gradients in equation 14 can be efficiently estimated in case $q(w)$, for explicit distributions $q(w), p_\phi(w \mid z), r(z \mid w)$ that can be reparametrized.

B THE ESTIMATE OF THE APPROXIMATION GAP WITH IWAE ESTIMATES

	$\mathcal{L}^{aux}(\theta, \psi)$	$\mathcal{L}^{IWAE}_{10000}(\theta, \psi)$	$G(\theta, \psi) \geq$	$\mathcal{L}^{aux}(\theta, \psi_{p_l})$
nats, $\times 10^3$	-23.375 ± 0.230	-9.957	13.418	-128.325 ± 2.436

Table 1: Comparison of the proposed auxiliary lower bound with IWAE lower bound estimation.

During variational inference with deep weight prior (Algorithm 1) we optimize a new auxiliary lower bound $\mathcal{L}^{aux}(\theta, \psi)$ on the evidence lower bound $\mathcal{L}(\theta)$. However, the quality of such inference depends on the gap $G(\theta, \psi)$ between the original variational lower bound $\mathcal{L}(\theta)$ and the variational lower bound in auxiliary space $\mathcal{L}^{aux}(\theta, \psi)$:

$$G(\theta, \psi) = \mathcal{L}(\theta) - \mathcal{L}^{aux}(\theta, \psi). \quad (16)$$

The gap $G(\theta, \psi)$ cannot be calculated exactly, but it can be estimated by using tighter but less computationally efficient lower bound. We follow Burda et al. (2015) and construct tighter lower bound $\mathcal{L}^{IWAE}_K(\theta, \psi)$:

$$\mathcal{L}^{IWAE}_K(\theta, \psi) = L_D + H(q(W)) + \quad (17)$$

$$+ \sum_{l,i,j} \mathbb{E}_{q(w^l_{ij} \mid \theta^l_{ij})} \mathbb{E}_{z_1, \dots, z_K \sim q(z \mid w^l_{ij} \psi_l)} \log \left(\frac{1}{K} \sum_{k=1}^{K} \frac{p(w^l_{ij} \mid z_k) p_l(z_k)}{q(z_k \mid w^l_{ij} \psi_l)} \right). \quad (18)$$

The estimate $\mathcal{L}^{IWAE}_K(\theta, \psi)$ converges to $\mathcal{L}(\theta)$ with K goes to infinity (Burda et al., 2015). We estimate the gap with $K = 10000$ as follows:

$$G(\theta, \psi) \geq \mathcal{L}^{IWAE}_{10000}(\theta, \psi) - \mathcal{L}^{aux}(\theta, \psi). \quad (19)$$

The results are presented at the Table 1. In order to show the range of the estimate and the gain from learning of $q(z \mid w^l_{ij} \psi_l)$ we compare results to the value of auxiliary lower bound $\mathcal{L}^{aux}(\theta, \psi_{p_l})$ computed at the point ψ_{p_l} where $q(z \mid w^l_{ij} \psi_{p_l}) \equiv p_l(z)$. The estimate of the gap, however, may be not very accurate and we consider it as a sanity check.

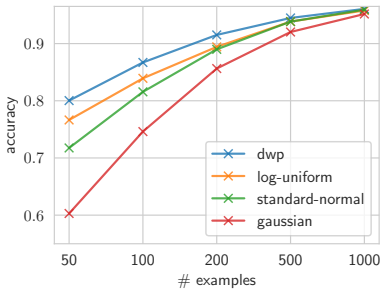


Figure 5: For different sizes of training set of MNIST dataset, we demonstrate the performance of variational inference with a fully-factorized variational approximation with three different prior distributions: deep weight prior (dwp), log-uniform, standard normal and learned multivariate gaussian. For more details see Section 4.1.

C Univariate Gaussian Prior

We examined a multivariate normal distribution $\hat{p}_l(w) = \mathcal{N}(w|\mu_l, \Sigma_l)$. We used a closed-form maximum-likelihood estimation for parameters μ_l, Σ_l over source dataset of learned kernels for each layer. We conducted the same experiment as in Section 4.1 for MNIST dataset for this gaussian prior, the results presented at Fig. 5. We found that the gaussian prior performs marginally worse than deep weight prior, log-uniform and standard normal. The gaussian prior could find a bad local optima and fail to approximate potentially multimodal source distribution of learned kernels.

D Visualization of Deep Weight Prior Latent Space

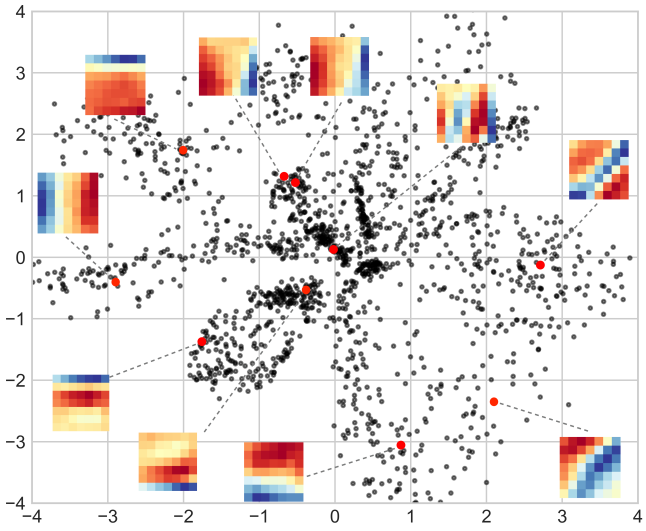
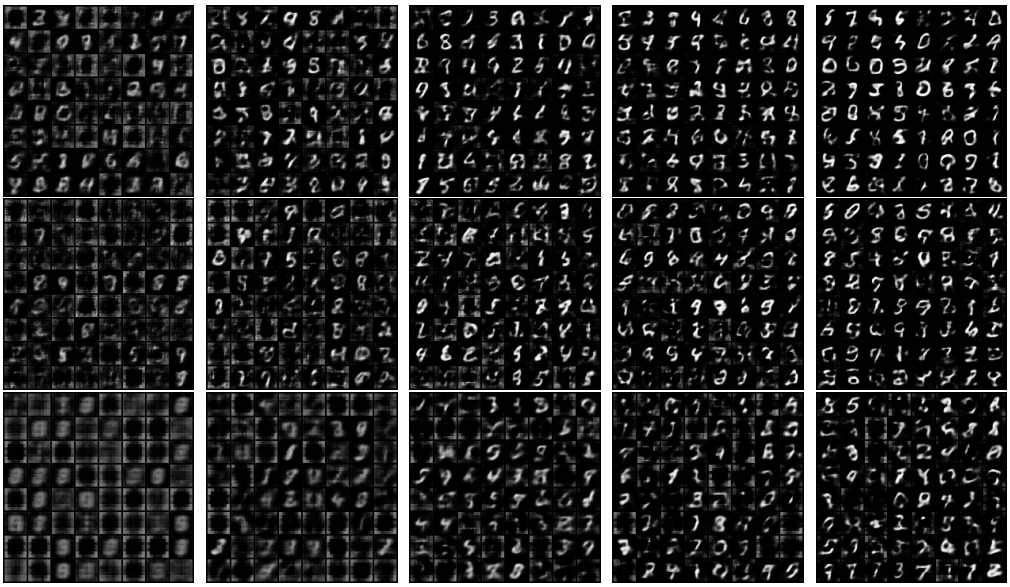


Figure 6: An illustration for Section 4.1. We visualize latent representations of convolutional filters for ConvNet on NotMNIST. Every point corresponds to mean of latent representation $q(z \mid w_i)$, where w_i is a kernel of shape 7×7 from the first convolutional layer, and $q(z \mid w_i)$ is an inference network with a two-dimensional latent represenation.

E SAMPLES FORM VARIATIONAL AUTO-ENCODERS



(a) 100 steps (b) 200 steps (c) 300 steps (d) 400 steps (e) 500 steps

Figure 7: An illustration for the Section 4.3 of samples from variational auto-encoder for three different types of initialization of convolutional layers after 100, 200, 300, 400 and 500 steps of optimization. The first row corresponds to deep weight prior initialization, the second to initialization with learned kernels, and the third to Xavier initialization.

F PRIOR ARCHITECTURES

Encoder5x5	**Decoder5x5**	**Encoder7x7**	**Decoder7x7**
Conv, 64, 3×3	Conv, 128, 1×1	Conv, 32, 3×3	ConvT, 64, 3×3
Conv, 64, 3×3	ConvT, 128, 3×3	Conv, 64, 3×3	ConvT, 64, 3×3
Conv, 128, 3×3	ConvT, 128, 3×3	Conv, 64, 3×3	ConvT, 32, 3×3
Conv, 128, 3×3	ConvT, 64, 1×1	$2 \times$ Linear, z_{dim}	$2 \times$ ConvT, $1, 1 \times 1$
$2 \times$ Linear, z_{dim}	$2 \times$ Conv, $1, 1 \times 1$		
= 260040 params	= 304194 params	= 56004 params	= 56674 params

Table 2: Architectures of variational auto-encoders for prior distributions. On the left for filters of shapes 5×5 and for filters of shape 7×7 on the right. See more details at Section 4 and Appendix H.1. All layers were divided with ELU non-literary.

G NETWORK ARCHITECTURES

Classification MNIST	**Classification CIFAR**	**Variational Auto-encoder MNIST**
Conv, 32, 7×7	Conv2d, 128, 7×7	Conv2d, 64, stride 2, 7×7
Conv, 128, 5×5	Conv2d, 256, 5×5	Conv2d, 128, 5×5
Linear, 10	Conv2d, 256, 5×5	$2\times$ Linear, z_{hid}
	Linear, 512	ConvT, 128, 5×5
	Linear, 10	ConvT, 64, stride 2, 5×5
		ConvT, 1, stride 2, 5×5
= 115658 params	= 5759498 params	= 1641665 params

Table 3: Network Architectures for MNIST and CIFAR-10/CIFAR-100 datasets (Section 4).

H PyTorch architectures

H.1 VAE Priors

- VAE model for 7x7 kernels ($z_{dim} = 2$, 300 epochs, Adam optimizer with linear learning rate decay from 1e-3 to 0.):

```
VAE(
  (encoder): Encoder7x7(
    (features): Sequential(
      (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1))
      (1): ELU(alpha=1.0)
      (2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
      (3): ELU(alpha=1.0)
      (4): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1))
      (5): ELU(alpha=1.0))
    (fc_mu): Conv2d(64, 2, kernel_size=(1, 1), stride=(1, 1))
    (fc_var): Conv2d(64, 2, kernel_size=(1, 1), stride=(1, 1)))
  (decoder): Decoder7x7(
    (decoder): Sequential(
      (0): ConvTranspose2d(2, 64,
      kernel_size=(3, 3), stride=(1, 1))
      (1): ELU(alpha=1.0)
      (2): ConvTranspose2d(64, 64,
      kernel_size=(3, 3), stride=(1, 1))
      (3): ELU(alpha=1.0)
      (4): ConvTranspose2d(64, 32,
      kernel_size=(3, 3), stride=(1, 1))
      (5): ELU(alpha=1.0))
    (fc_mu): Conv2d(32, 1, kernel_size=(1, 1), stride=(1, 1))
    (fc_var): Conv2d(32, 1, kernel_size=(1, 1), stride=(1, 1))))
```

- VAE model for 5x5 kernels ($z_{dim} = 4$, 300 epochs, Adam optimizer with linear learning rate decay from 1e-3 to 0.):

```
VAE(
  (encoder): Encoder5x5(
    (features): Sequential(
      (0): Conv2d(1, 64, kernel_size=(3, 3),
      stride=(1, 1), padding=(1, 1))
      (1): ELU(alpha=1.0)
      (2): Conv2d(64, 64, kernel_size=(3, 3),
      stride=(1, 1), padding=(1, 1))
      (3): ELU(alpha=1.0)
      (4): Conv2d(64, 128, kernel_size=(3, 3),
      stride=(1, 1))
      (5): ELU(alpha=1.0)
      (6): Conv2d(128, 128, kernel_size=(3, 3),
      stride=(1, 1))
      (7): ELU(alpha=1.0))
    (fc_mu): Conv2d(128, 4, kernel_size=(1, 1), stride=(1, 1))
    (fc_sigma): Conv2d(128, 4, kernel_size=(1, 1), stride=(1, 1)))
  (decoder): Decoder5x5(
    (activation): ELU(alpha=1.0)
    (decoder): Sequential(
      (0): Conv2d(4, 128, kernel_size=(1, 1), stride=(1, 1))
      (1): ELU(alpha=1.0)
      (2): ConvTranspose2d(128, 128,
      kernel_size=(3, 3), stride=(1, 1))
      (3): ELU(alpha=1.0)
      (4): ConvTranspose2d(128, 128,
      kernel_size=(3, 3), stride=(1, 1))
      (5): ELU(alpha=1.0)
      (6): Conv2d(128, 64, kernel_size=(1, 1), stride=(1, 1))
      (7): ELU(alpha=1.0))
    (fc_mu): Sequential(
      (0): Conv2d(64, 1, kernel_size=(1, 1), stride=(1, 1)))
    (fc_var): Sequential(
      (0): Conv2d(64, 1, kernel_size=(1, 1), stride=(1, 1))
    )))
```

H.2 NOTMNIST-MNIST

- Source models trained on notMNIST (l2=1e-3, 100 epochs, Adam optimizer with linear learning rate decay from 1e-3 to 0.) look as follows:

```
FConvMNIST(
  (features): Sequential(
    (conv1): Conv2d(1, 256, kernel_size=(7, 7), stride=(1, 1))
    (relu1): LeakyReLU(negative_slope=0.01)
    (mp1): MaxPool2d(kernel_size=2,
    stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv2d(256, 512, kernel_size=(5, 5), stride=(1, 1))
    (relu2): LeakyReLU(negative_slope=0.01)
    (mp2): MaxPool2d(kernel_size=2,
    stride=2, padding=0, dilation=1, ceil_mode=False)
    (flatten): Flatten())
  (classifier): Linear(in_features=4608,
  out_features=10, bias=True))
```

- The final model (deterministic) trained on MNIST looks as follows (Adam optimizer with linear learning rate decay from 1e-3 to 0.):

```
FConvMNIST(
  (features): Sequential(
    (conv1): Conv2d(1, 32, kernel_size=(7, 7), stride=(1, 1))
    (relu1): LeakyReLU(negative_slope=0.01)
    (mp1): MaxPool2d(kernel_size=2, stride=2,
    padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv2d(32, 128, kernel_size=(5, 5), stride=(1, 1))
    (relu2): LeakyReLU(negative_slope=0.01)
    (mp2): MaxPool2d(kernel_size=2,
    stride=2, padding=0, dilation=1, ceil_mode=False)
    (flatten): Flatten())
  (classifier): Linear(in_features=1152,
  out_features=10, bias=True))
```

- The final model (bayesian) trained on MNIST looks as follows (Adam optimizer with linear learning rate decay from 1e-3 to 0.):

```
FConvMNIST(
  (features): Sequential(
    (conv1): BayesConv2d(
      (mean): Conv2d(1, 32, kernel_size=(7, 7), stride=(1, 1))
      (var): LogScaleConv2d(1, 32,
      kernel_size=(7, 7), stride=(1, 1), bias=False))
    (relu1): LeakyReLU(negative_slope=0.01)
    (mp1): MaxPool2d(kernel_size=2, stride=2,
    padding=0, dilation=1, ceil_mode=False)
    (conv2): BayesConv2d(
      (mean): Conv2d(32, 128, kernel_size=(5, 5), stride=(1, 1))
      (var): LogScaleConv2d(32, 128,
      kernel_size=(5, 5), stride=(1, 1), bias=False))
    (relu2): LeakyReLU(negative_slope=0.01)
    (mp2): MaxPool2d(kernel_size=2,
    stride=2, padding=0, dilation=1, ceil_mode=False)
    (flatten): Flatten())
  (classifier): Linear(in_features=1152,
  out_features=10, bias=True))
```

H.3 CIFAR

- The source model for CIFAR looks as follows (l2=1e-4, 300 epochs, Adam, Linear learning rate decay from 1e-3 to 0.):

```
CIFARNet(
  (features): Sequential(
    (conv1): Conv2d(3, 128, kernel_size=(7, 7), stride=(1, 1))
    (bn1): BatchNorm2d(128,
       eps=1e-05, momentum=0.1,
       affine=True, track_running_stats=True)
    (relu1): LeakyReLU(negative_slope=0.01)
```

```
    (maxpool): MaxPool2d(
       kernel_size=2, stride=2,
       padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv2d(128, 256, kernel_size=(5, 5), stride=(1, 1))
    (bn2): BatchNorm2d(
       256, eps=1e-05, momentum=0.1,
       affine=True, track_running_stats=True)
    (relu2): LeakyReLU(negative_slope=0.01)
    (conv3): Conv2d(256, 256, kernel_size=(5, 5), stride=(1, 1))
    (bn3): BatchNorm2d(256,
       eps=1e-05, momentum=0.1,
       affine=True, track_running_stats=True)
    (relu3): LeakyReLU(negative_slope=0.01)
    (conv4): Conv2d(256, 512, kernel_size=(5, 5), stride=(1, 1))
    (bn4): BatchNorm2d(512,
       eps=1e-05, momentum=0.1,
       affine=True, track_running_stats=True)
    (relu4): LeakyReLU(negative_slope=0.01)
    (flatten): Flatten())
  (classifier): Sequential(
    (fc1): Linear(in_features=512, out_features=512, bias=True)
    (bn1): BatchNorm1d(512,
       eps=1e-05, momentum=0.1,
       affine=True, track_running_stats=True)
    (relu1): LeakyReLU(negative_slope=0.01)
    (linear): Linear(in_features=512, out_features=100, bias=True)
))
```

- The final deterministic model (for CIFAR10) looks as follows:

```
CIFARNetNew(
  (features): Sequential(
    (conv1): Conv2d(3, 128, kernel_size=(7, 7), stride=(1, 1))
    (relu1): LeakyReLU(negative_slope=0.01)
    (maxpool): MaxPool2d(kernel_size=2, stride=2,
              padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv2d(128, 256, kernel_size=(5, 5), stride=(1, 1))
    (relu2): LeakyReLU(negative_slope=0.01)
    (conv3): Conv2d(256, 256, kernel_size=(5, 5), stride=(1, 1))
    (relu3): LeakyReLU(negative_slope=0.01)
    (flatten): Flatten())
  (classifier): Sequential(
    (fc1): Linear(in_features=6400, out_features=512, bias=True)
    (relu1): LeakyReLU(negative_slope=0.01)
    (linear): Linear(in_features=512, out_features=10, bias=True)
))
```

- The final Bayesian model (for CIFAR10) looks as follows:

```
CIFARNetNew(
  (features): Sequential(
    (conv1): BayesConv2d(
      (mean): Conv2d(3, 128, kernel_size=(7, 7), stride=(1, 1))
      (var): LogScaleConv2d(3, 128,
      kernel_size=(7, 7), stride=(1, 1), bias=False))
    (relu1): LeakyReLU(negative_slope=0.01)
    (maxpool): MaxPool2d(kernel_size=2, stride=2, padding=0,
    dilation=1, ceil_mode=False)
    (conv2): BayesConv2d(
      (mean): Conv2d(128, 256, kernel_size=(5, 5), stride=(1, 1))
      (var): LogScaleConv2d(128, 256, kernel_size=(5, 5),
      stride=(1, 1), bias=False))
    (relu2): LeakyReLU(negative_slope=0.01)
    (conv3): BayesConv2d(
      (mean): Conv2d(256, 256, kernel_size=(5, 5), stride=(1, 1))
      (var): LogScaleConv2d(256, 256,
      kernel_size=(5, 5), stride=(1, 1), bias=False))
    (relu3): LeakyReLU(negative_slope=0.01)
    (flatten): Flatten())
  (classifier): Sequential(
    (fc1): Linear(in_features=6400, out_features=512, bias=True)
    (relu1): LeakyReLU(negative_slope=0.01)
    (linear): Linear(in_features=512, out_features=10, bias=True)))
```

Appendix D Article. A new approach for sparse Bayesian channel estimation in SCMA uplink systems

Authors. Kirill Struminsky, Stanislav Kruglik, Dmitry Vetrov, Ivan Oseledets
Published in: 8th International Conference on Wireless Communications & Signal Processing (WCSP). – IEEE, 2016. – C. 1-5.
Abstract. The rapid growth of traffic and number of simultaneously available devices leads to the new challenges in constructing fifth generation wireless networks (5G). To handle with them various schemes of non-orthogonal multiple access (NOMA) were proposed. One of these schemes is Sparse Code Multiple Access (SCMA), which is shown to achieve better link level performance. In order to support SCMA signal decoding channel estimation is needed and sparse Bayesian learning framework may be used to reduce the requirement of pilot overhead. In this paper we propose a modification of sparse Bayesian learning based channel estimation algorithm that is shown to achieve better accuracy of user detection and faster convergence in numerical simulations.

A New Approach for Sparse Bayesian Channel Estimation in SCMA Uplink Systems

Kirill Struminsky *, Stanislav Kruglik †‡§, Dmitry Vetrov *†, Ivan Oseledets †
HSE *, Skoltech †, IITP RAS ‡, MIPT §
Russian Federation
Email: kstruminskiy@hse.ru, kruglik@iitp.ru, vetrovd@yandex.ru, i.oseledets@skoltech.ru

***Abstract*—The rapid growth of traffic and number of simultaneously available devices leads to the new challenges in constructing fifth generation wireless networks (5G). To handle with them various schemes of non-orthogonal multiple access (NOMA) were proposed. One of these schemes is Sparse Code Multiple Access (SCMA), which is shown to achieve better link level performance. In order to support SCMA signal decoding channel estimation is needed and sparse Bayesian learning framework may be used to reduce the requirement of pilot overhead. In this paper we propose a modification of sparse Bayesian learning based channel estimation algorithm that is shown to achieve better accuracy of user detection and faster convergence in numerical simulations.**

***Index Terms*—5G; SCMA; channel estimation; sparse Bayesian learning; active user detection**

I. Introduction

A fifth generation (5G) wireless communication standard, which expected to be commercially used in 2020, includes support of very diverse applications and tremendous number of users as a basic part of IoT concept. It must also support massive connectivity, achieve spectral efficiency and lower latency [1]. Unfortunately, newly launched Long Term Evolution Advanced (LTE-A) networks are not efficient enough to meet all requirements that are imposed to 5G systems, especially in uplink (UL). To deal with them a new Non-Orthogonal Multiple Access (NOMA) scheme – Sparse Code Multiple Access (SCMA) was introduced in [2]. As other NOMA schemes SCMA brings some controllable interference to implement overloading at the cost of increased receiver complexity in order to achieve higher spectral efficiency and massive connectivity [3]. In SCMA systems incoming data from different streams are directly mapped to the codewords from multi-dimensional codebooks. Multiple users select their codebooks and pilots and then transmit their data in pre-configured resource blocks without preliminary request procedures. The main advantage of SCMA over another NOMA schemes is some potential gain of multi-dimensional constellation shaping [4].

The main problem with coherent signal detection is necessity of channel estimation, because demodulation of received signal is possible only after obtaining channel state information. In SCMA, this question is being studied unlike traditional digital telecommunication systems. In [5] blind active user detection with joint message passing algorithm was introduced. Unfortunately, it makes an assumption that SCMA layers share the same time-frequency resource block which isn't true in general. The other approaches – compressive sensing detectors with orthogonal matching pursuit [6] and compressive sampling matching pursuit [7] suffer from severe performance loss in case of further decreasing of number of received pilot resources. This problem arises due to convex relaxation from l_0 minimization to l_1 minimization [8].

To handle with these problems and to increase overall frequency efficiency a robust active UE detector based on Sparse Bayesian Learning was proposed in [9]. In this article, we modify this approach and achieve better convergence rate which is important in case of tremendous number of simultaneously active users as one of the factors that can increase total decoding speed and as a result decrease latency. Our numerical simulation results show a five time increase of convergence rate in case of 36 potential users with 6 active users and 20 pilots in one fading block. Our algorithm employs a modified iterative scheme for approximate Bayesian channel parameter estimation. This modification was first mentioned in [10] as the one that shows better convergence results in machine learning problems.

The rest of this paper is organized as follows. The system model is presented in Section II. Our modification of SBL detector is presented in Section III. Comparison of different user detectors based on SBL framework performed by numerical simulation is presented in Section IV. This paper is concluded in Section V.

II. System model

A. SCMA encoding

An SCMA encoding procedure is defined as mapping from $log_2(M)$ bits to a K-dimensional complex codebook of size M [2]. In the uplink transmission each user has its own codebook and its data bits are mapped into K-dimensional complex codeword with $N < K$ non-zero entries selected from corresponding codebook. After it these bits are transmitted through K resource elements (REs) (for example Orthogonal Frequency-Division Multiple Access resource elements). In this case one RE is the resource of one subcarrier in Orthogonal Frequency-Division Multiplexing and multiple overlapped SCMA blocks may fit within assigned time-frequency resources [11]. In the uplink transmission scheme with SCMA multiple access is achieved through the sharing of the same time-frequency resources among SCMA layers of multiple active users [12].

978-1-5090-2860-3/16/$31.00 ©2016 IEEE

Signal from U users that arrived through different channels after the synchronous level multiplexing can be expressed as [2]

$$\vec{y} = \sum_{u=1}^{U} \sqrt{P_u} diag(\vec{h_u}) \vec{x_u} + \vec{n}$$

where $x_u = [x_{u,1}, x_{u,2}, ..., x_{u,K}]^T$ is SCMA codeword transmitted by user u, P_u – power of received signal, $\vec{h_u} = [h_{u,1}, h_{u,2}, \ldots, h_{u,K}]^T$ – channel vector of user u, $\vec{n}$ – Gaussian ambient noise and cell-off interference.

B. Channel model

In this article we consider system with allocation of SCMA codewords from N users into one resource block. We assume time-invariant channel response within range t_1 to t_2 and block Rayleigh fading with size of each block equal to B. Total amount of fading blocks is Q. Within each fading block we assign N_p REs of pilots to identify different users and N_d REs to transmit SCMA codewords ($B = N_p + N_d$). An active user pick its pilot sequence and its corresponding codebook during SCMA UL access.

In one resource block the received pilot vector $\vec{y}$ can be expressed as follows:

$$\vec{y} = [P_1 \ P_2 \ \ldots \ P_N] \begin{bmatrix} \vec{c_1} a_1 \\ \vec{c_2} a_2 \\ \vdots \\ \vec{c_N} a_N \end{bmatrix} + \vec{n}$$

where $P_n = \begin{bmatrix} \vec{p}_{n,1} & 0 & \ldots & 0 \\ 0 & \vec{p}_{n,2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \vec{p}_{n,Q} \end{bmatrix}$ $(1 \le n \le N)$

$$\vec{c}_n = [c_{n1} \ c_{n2} \ \ldots \ c_{nQ}]^T$$

and $p_{n,q}$–n-th pilot sequence of q-th fading block, $a_n \in \{0, 1\}$–n-th pilot sequence activity indicator, $c_{n,q}$–q-th fading block channel response of active user who picked n-th pilot-code, $\vec{n}$ – additive white Gaussian noise.

During transmission we make following additional assumptions:

- Each user is synchronized in symbol level as well as in block level
- Only small part of all available users transmit information simultaneously
- During transmission active user picks a pilot sequence and its corresponding SCMA codebook according to its user index or randomly
- Pilot sequences are assigned without repetitions
- For each user channel response within one fading block is constant
- The size of fading block depends on the channel condition
- Each resourse block is competitively grant-free accessed by multiple users

It should be mentioned that the number of available pilot codes must exceed the number of active users in order to ensure low error rate for user separation.

III. Channel estimation algorithm

A. Real-valued formulation and notation

In matrix notatin received pilot vector may be expressed as

$$\vec{y} = P\vec{\theta} + \vec{n}. \tag{1}$$

Here we have to estimate vector $\vec{\theta} = [\vec{c_1} a_1, \vec{c_2} a_2, \ldots, \vec{c_N} a_n]^T$ given $\vec{y}$ and P.

All the entries of (1) are complex-valued and for the sake of clarity we will reformulate the problem of $\vec{\theta}$ estimation in terms of real-valued vectors and matrices. We use standard embedding to the real vector spaces:

$$\begin{aligned}
\vec{y}_{\mathbb{R}} \leftarrow & [\mathrm{Re}(y_1), \ \mathrm{Im}(y_1), \ \ldots, \ \mathrm{Re}(y_D), \ \mathrm{Im}(y_D)]^T \\
\vec{n}_{\mathbb{R}} \leftarrow & [\mathrm{Re}(n_1), \ \mathrm{Im}(n_1), \ \ldots, \ \mathrm{Re}(n_D), \ \mathrm{Im}(n_D)]^T \\
\vec{\theta}_{\mathbb{R}} \leftarrow & [\mathrm{Re}(c_{(1,1)} a_1), \ \mathrm{Im}(c_{(1,1)} a_1), \ \ldots, \ \mathrm{Re}(c_{(1,Q)} a_1), \ \mathrm{Im}(c_{(1,Q)} a_1), \\
& \ldots \\
& \mathrm{Re}(c_{(N,1)} a_N), \mathrm{Im}(c_{(N,1)} a_N), \ldots, \mathrm{Re}(c_{(N,Q)} a_N), \mathrm{Im}(c_{(N,Q)} a_N)]^T
\end{aligned}$$

Vector $\vec{\theta}$ contains information about channel coefficients for Q fading blocks for N users. We used double indices to denote its components, the dimensionality of embedded vector $\vec{\theta}_{\mathbb{R}}$ is twice the dimensionality of $\vec{\theta}$. The embedding replaces each complex fading block coefficient with its real and imaginary parts, so we will use double indices to denote $\theta_{\mathbb{R}}$ components where the second integer in index now varies from 1 to $2Q$.

Each component of P is mapped into 2×2 block in real-valued matrix by the following rule:

$$\begin{aligned}
P_{\mathbb{R},2i,2j}, P_{\mathbb{R},2i+1,2j+1} &\leftarrow \mathrm{Re}(P_{ij}) \\
P_{\mathbb{R},2i+1,2j} &\leftarrow \mathrm{Im}(P_{ij}) \\
P_{\mathbb{R},2i,2j+1} &\leftarrow -\mathrm{Im}(P_{ij})
\end{aligned}$$

From now we will operate only with real-valued vectors, thus with a slight abuse of notation we will use the original symbols to denote real-valued representations of vectors and matrices (e.g. $\vec{y}$ instead of $\vec{y}_{\mathbb{R}}$).

B. Probabilistic model

We consider the following probabilistic model of pilots transmission for SCMA UL detection:

$$p(\vec{y}, \vec{\theta} | \vec{\gamma}) = p(\vec{y} | \vec{\theta}) p(\vec{\theta} | \vec{\gamma}) \tag{2}$$

This model inherets distribution $p(\vec{y}|\vec{\theta})$ from the channel design and introduces a prior distribution $p(\vec{\theta}|\vec{\gamma})$ in order to perform Bayesian inference in the model.

Here the received vector $\vec{y}$ depends on channel parameter vector $\vec{\theta}$ as follows:

$$p(\vec{y}|\vec{\theta}) = \mathcal{N}(\vec{y} | P\vec{\theta}, diag(\rho, ..., \rho))$$

This distribution both captures linear dependence between $\vec{\theta}$ and $\vec{y}$, and additive Gaussian noise with variance ρ. We assume that ρ is a fixed parameter that is known to the receiver.

Prior distribution over $\vec{\theta}$ is parametrized by vector $\vec{\gamma}$ and defined as follows:

$$p(\vec{\theta}|\vec{\gamma}) = \prod_{n=1}^{N}\prod_{q=1}^{2Q}\frac{1}{\sqrt{2\pi\gamma_n}}\exp\left\{\left(-\frac{\theta^2_{(n,q)}}{2\gamma_n}\right)\right\} = \mathcal{N}(\vec{\theta}|0,\Gamma),$$

were $\Gamma = diag(\underbrace{\gamma_1,...,\gamma_1}_{2Q},...,\underbrace{\gamma_N,...,\gamma_N}_{2Q})$.

Note that for each user n channel components $(\theta_{(n,1)},...,\theta_{(n,2Q)})$ have normal distribution with variance parametrized by γ_n. If $\gamma_n \to 0$, then the components of distribution concentrates at zero. On the other hand, as γ_n becomes larger, the distribution becomes less restrictive. Thus $p(\vec{\theta}|\vec{\gamma})$ is capable to model sparse vectors $\vec{\theta}$ with certain parameters $\vec{\gamma}$.

C. Parameter estimation

A standard approach to infer a model parameter or an unobserved variable with respect to observed data is to compute it's posterior distribution via Bayes rule. Given $\vec{\gamma}$, one can easily compute posterior distribution $p(\vec{\theta}|\vec{y},\vec{\gamma})$ to estimate $\vec{\theta}$. However, parameters $\vec{\gamma}$ are not known, but the estimation performance dramatically depends on their values.

Bayesian approach for point estimates suggests to select parameters that maximize model evidence:

$$\vec{\gamma}_* = arg\max p(\vec{y}|\vec{\gamma})$$

where evidence is defined by the summation rule:

$$p(\vec{y}|\vec{\gamma}) = \int p(\vec{y},\vec{\theta}|\vec{\gamma})p(\vec{\theta}|\vec{\gamma})\mathrm{d}\vec{\theta}.$$

Since the density function $p(\vec{y}|\vec{\gamma})$ is a convolution of two normal distributions, it has the following form:

$$p(\vec{y}|\vec{\gamma}) = \frac{1}{(2\pi)^D}|\Sigma_t|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}\vec{y}^T\Sigma_t^{-1}\vec{y}\right),$$

where $\Sigma_t = \frac{1}{2\rho}I + P\Gamma P^T$.

Analytical model evidence maximization leads to the system of non-linear equations, and it is more practical to maximize a lower bound for evidence via EM-algorithm. Consider the following log-evidence representation:

$$\log p(\vec{y}|\gamma) = \mathbb{E}_{q(\vec{\theta})}\log p(\vec{y},\vec{\theta}|\vec{\gamma}) + H(q(\vec{\theta})) + KL(q(\vec{\theta})||p(\vec{\theta}|\vec{y},\vec{\gamma})). \quad (3)$$

Kullback-Leibler divergence $KL(q(\vec{\theta})||p(\vec{\theta}|\vec{y},\vec{\gamma}))$ is always non-negative, thus terms $\mathcal{L}(q(\theta),\gamma) = \mathbb{E}_{q(\vec{\theta})}\log p(\vec{y},\vec{\theta}|\vec{\gamma}) + H(q(\vec{\theta}))$ form a lower bound for log-evidence. EM-algorithm iteratevly maximizes the evidence lower bound by minimizing the third term in 3 and maximizing the first term in 3 is known as EM-algorithm.

On k-th E-step the algorithm minimizes the third term in (3) with respect to distribution $q(\vec{\theta})$ by setting it to be equal to

$$p(\vec{\theta}|\vec{y},\vec{\gamma}^{(k)}) = \mathcal{N}(\mu_\theta,\Sigma_\theta)$$
$$\vec{\mu}_\theta = \Gamma^{(k)}P^T(P\Gamma^{(k)}P^T + \frac{1}{2\rho}I)^{-1}\vec{y}$$
$$\Sigma_\theta = \Gamma^{(k)} - \Gamma^{(k)}P^T(P\Gamma^{(k)}P^T + \frac{1}{2\rho}I)^{-1}P\Gamma^{(k)}$$

for $\Gamma^{(k)} = diag(\underbrace{\gamma_1^{(k)},...,\gamma_1^{(k)}}_{2Q},...,\underbrace{\gamma_N^{(k)},...,\gamma_N^{(k)}}_{2Q})$.

Indices for $\vec{\mu}_\theta$ and Σ_θ are inherited from indices for $\vec{\theta}$. On the M-step the first term in (3) is maximized with respect to $\vec{\gamma}$. Finding exact maximum of the first term of (3) give us the following set of equations:

$$0 = 2Q - \frac{1}{\gamma_n}\sum_{q=1}^{2Q}(\Sigma_{\theta(n,q)(n,q)} + \mu^2_{\theta(n,q)}), \quad n = 1,...,N. \quad (4)$$

Solving each equation with respect to γ_n leads to the following M-step:

$$\gamma_n^{(k+1)} = \frac{\sum_{q=1}^{2Q}\left(\Sigma_{\theta(n,q)(n,q)} + \mu^2_{\theta(n,q)}\right)}{2Q}. \quad (5)$$

This scheme was proposed [9], yet in the original paper a different prior $p(\vec{\theta}|\vec{\gamma})$ was used and the summation in (5) was introduced as a heuristic. It turns out that the original scheme can be represented as EM-algorithm in our probabilistic model.

In this paper we propose to follow the approach of McKay [10] [13] for the M-step, since it is known to converge faster in practice. Firstly, we rearrange the terms in (4) as follows:

$$\frac{1}{\gamma_n}\sum_{q=1}^{2Q}\mu^2_{\theta(n,q)} = 2Q - \frac{1}{\gamma_n}\sum_{q=1}^{2Q}\Sigma_{\theta(n,q)(n,q)}. \quad (6)$$

Let us consider an iterative scheme where new values of $\gamma_n^{(k+1)}$ are evaluated by substituting all entries of γ_n with $\gamma_n^{(k)}$ on the right hand side of 6, substituting γ_n with $\gamma_n^{(k+1)}$ on the left hand side of the equation and solving it with respect to $\gamma_n^{(k+1)}$. We obtain the following iterative scheme:

$$\gamma_n^{(k+1)} = \frac{\gamma_n^{(k)}\sum_{q=1}^{2Q}\mu^2_{\theta(n,q)}}{\gamma_n^{(k)}2Q - \sum_{q=1}^{2Q}\Sigma_{\theta(n,q)(n,q)}}. \quad (7)$$

It is clear that fixed points of this scheme are extreme points of the evidence lower bound. This M-step defines EM-algorithm summarized in **Algorithm 1**.

Complexity of the first algorithm was analyzed in [9], our detector has the same complexity $O(K(D^3+(QN)^3)$. Here K is the maximal number of iterations, D is the dimensionality of y, Q is number of fading blocks and N is number of potential users.

Algorithm 1 SBL-Based Active User Detector and Channel Estimator for SCMA UL

Require: $\vec{y}$ is a real-valued received vector, P is a real-valued matrix, ρ defines AWGN variance, $\vec{\gamma}^{initial}$ is an initial prior parameter, δ defines decision threshold, K defines maximal number of iterations

function ESTIMATOR($\vec{y}, P, \rho, \vec{\gamma}^{initial}, \delta, K$)
  $\vec{\gamma}^{(1)} \leftarrow \vec{\gamma}^{initial}$
  for $k \leftarrow 1$ to K **do**
    $\Gamma^{(k)} \leftarrow diag(\gamma_1^{(k)}, ...\gamma_1^{(k)}, ..., \gamma_N^{(k)}, ..., \gamma_N^{(k)})$
    $\vec{\mu}_\theta \leftarrow \Gamma^{(k)} P^T (P\Gamma^{(k)} P^T + \frac{1}{2\rho} I)^{-1} \vec{y}$
    $\Sigma_\theta \leftarrow \Gamma^{(k)} - \Gamma^{(k)} P^T (P\Gamma^{(k)} P^T + \frac{1}{2\rho} I)^{-1} P\Gamma^{(k)}$
    for $n \leftarrow 1$ to N **do**
      $\gamma_n^{(k+1)} \leftarrow \frac{\sum_{q=1}^{2Q} \mu^2_{\theta(n,q)}}{\gamma_n^{(k)} 2Q + \sum_{q=1}^{2Q} \Sigma_{\theta(n,q)(n,q)}}$
      if $\gamma_n^{(k+1)} < \delta$ **then**
        $\gamma_n^{(k+1)} \leftarrow 0$
      end if
    end for
  end for
  $\Gamma \leftarrow diag(\gamma_1^{(K)}, ...\gamma_1^{(K)}, ..., \gamma_N^{(K)}, ..., \gamma_N^{(K)})$
  $\hat{\theta} \leftarrow \Gamma P^T (P\Gamma P^T + \frac{1}{2\rho} I)^{-1} \vec{y}$
  return $\hat{\theta}$
end function

IV. Simulation results

Finally, simulation results are obtained to compare channel estimation and user detection performance of the proposed algorithm and the sparse Bayesian estimator from [9].

In our experiments we perform link-level simulation for uplink transmission in Rayleigh fading channel. We simulate a SCMA UL system with $N = 36$ potential and 6 active users. The total resource block was divided into $Q = 5$ fading blocks, within each fading block pilot sequences of 20 elemets were assigned to each user.

We used the pilot sequences that are used in demodulated reference signal in LTE systems [14]. We set 6 base pilot sequences to be Zadoff-Chu sequences and then obtain new sequences with cyclic shift of base pilot sequences.

We used Mean Square Error ($MSE = \mathbb{E}\left[\frac{(\vec{\theta}-\theta^*)^T(\vec{\theta}-\theta^*)}{N}\right]$) to measure the quality of channel estimation and User Detection Error Rate ($UDER = \mathbb{E}\left[\frac{\sum_{n=1}^{N}[\hat{a}_n \neq a_{n^*}]}{N}\right]$) to measure the quality of user detection. Also we analyzed the convergence of the algorithm with respect to the maximal number of iterations K.

Figure 1 shows the convergence of mean squared error for diffefent channel conditions. The simulation results show that the proposed scheme almost reaches the minimum of mean squared error several times faster than the original algorithm. On the other hand, the proposed scheme is less precise at low signal-to-noise ratios. Figure 2 presents the convergence of user detection error rate for different channel conditions. It

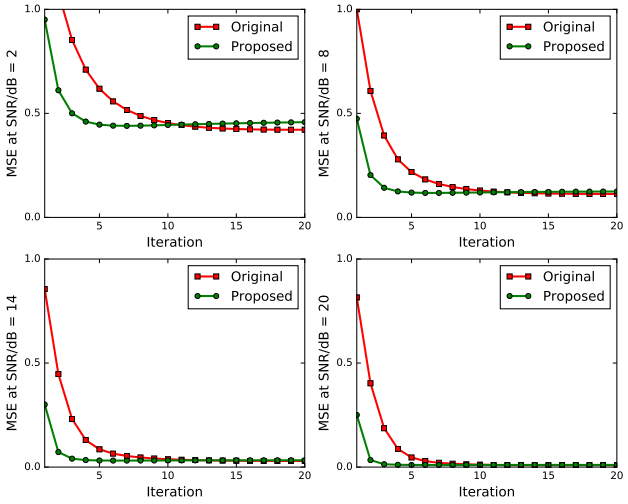


Fig. 1. Convergence of the Mean Square Error of channel estimation for different noise ratios

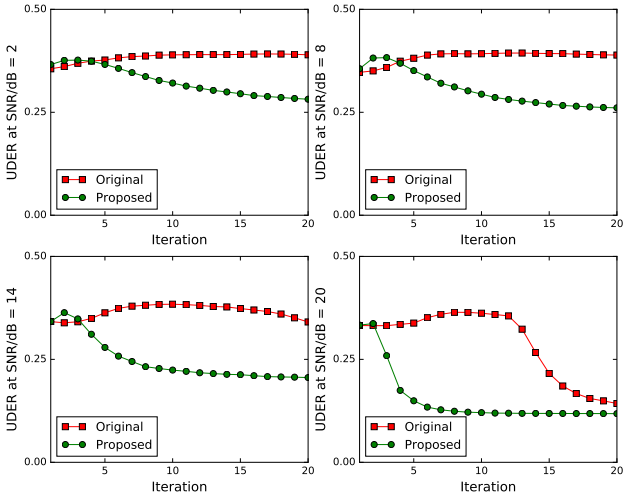


Fig. 2. Convergence of the user detection error rate for different noise ratios

can be seen that the proposed scheme has both lower detection error rate and converges faster.

Figure 3 demonstrates error rate after $K = 20$ iteration steps for the original scheme and $K = 10, 20$ iteration steps for the proposed scheme with different channel conditions. The simulation results show that the proposed scheme maintains user detection improvement even for fewer number of iterations.

V. Conclusion

In this paper the improvement of active user detector based on the theory of sparse Bayesian learning is proposed. This improvement is inspired by an empirical fact about EM-algorithm performance in the probabilistic model that we use. Simulation results are provided to substantiate the performance improvement of the detector and practical value of the proposed scheme. Although no theoretical explanations for this behavior are presented in literature, such explanations

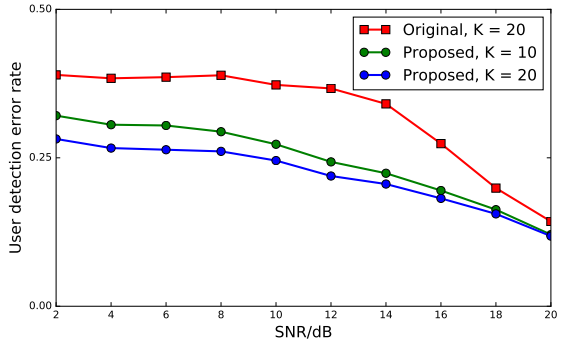


Fig. 3. User detection error rate after K iterations

and adaptation of faster or approximate procedures to replace matrix inversion in the algorithm are the directions for future research.

ACKNOWLEDGMENT

Stanislav Kruglik was supported by RFBR project No. 16-01-00716. Dmitry Vetrov was supported by RFBR project No. 15-31-20596 (mol-a-ved).

REFERENCES

[1] Chih-Lin I et all., *New Paradigm of 5G Wireless Internet*. IEEE Journal on Selected Areas in Communications, volume 34, issue 3, pp. 474 -482, 2016.

[2] Hosein Nikopour and Hadi Baligh, *Sparse code multiple access*. IEEE 24th PIMRC, pp. 332-336, 2013.

[3] B. Wang, K. Wang, Z. Lu, T. Xie and J. Quan, *Comparation study of non-orthogonal multiple access schemes for 5G*. BMSB, pp. 1-5, 2015.

[4] L. Dai, B. Wang, Y. Yuan, S. Han, C. I and Z. Wang, *Non-orthogonal multiple access for 5G: solutions, challenges, opportunities, and future research thrends*. IEEE Commun. Mag., volume 53, issue 9, pp. 74-81, 2015.

[5] Nikopour H. ; Baligh H. Bayesteh, A; Yi. *Blind detection of scma for uplink grant-free multiple-access*. In Wireless Communications Systems (ISWCS), 2014 11th International Symposium on, pages 853-857. IEEE, 2014.

[6] D.L. Donoho. *Compressed sensing*. Information Theory, IEEE Transactions on, 52(4):1289-1306, 2006.

[7] Tropp, J.A. ; Gilbert. *Signal recovery from random measurements via orthogonal matching pursuit*. In Information Theory, IEEE Transactions on, pages 4665-4666. IEEE, 2007.

[8] D. Needell; J.A. Tropp. *Cosamp: Iterative signal recovery from incomplete and inaccurate samples*. Applied and Computational Harmonic Analysis, 26(3); 301-321, 2008.

[9] Yufeng Wang; Shidong Zhou ; Limin Xiao ; Xiujun Zhang ; Jin Lian. *Sparse Bayesian learning based user detection and channel estimation for SCMA uplink systems*.Wireless Communications & Signal Processing (WCSP), 2015 International Conference on, pages 1-5, 2015.

[10] David J. C. MacKay. *Bayesian Interpolation*. Neural Computation 4, 415-447, 1991.

[11] H. ; Yi E. Au, K. ;Liqing Zhang ; Nikopour. *Uplink contention based scma for 5g radio access*. In Globecom Workshops (GC Wkshps), 2014, pages 900-905. IEEE, 2014.

[12] H. ; Bayesteh A. ; Baligh H. Taherzadeh, M. ; Nikopour. *Scma codebook design*. In Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th, pages 1-5. IEEE, 2014.

[13] Tipping, M. E. *Sparse Bayesian learning and the relevance vector machine*. Journal of Machine Learning Research 1, 211-244, 2001.

[14] 3GPP official website, LTE official specification, Web: http://www.3gpp.org/specifications.

Appendix E Article. Well Log Data Standardization, Imputation and Anomaly Detection Using Hidden Markov Models

Authors. Kirill Struminskiy, Artyom Klenitskiy, Alexander Reshytko, Dmitry Egorov, Artyom Shchepetnov, Artur Sabirov, Dmitry Vetrov, Artyom Semenikhin, Oksana Osmonalieva, Boris Belozerov
Published in: Petroleum Geostatistics 2019. – European Association of Geoscientists & Engineers, 2019. – T. 2019. – No. 1. – C. 1-5
Abstract. The main goal of this work is to develop a systematic approach to work with raw well log data. Toaccomplish this goal, we propose to fit a simple unsupervised generative model to the input data and au-tomate the preprocessing step using the generative model. This approach allows to detect the anomaliesin the data as the regions that the model struggles to explain (i.e., samples with extremely low like-lihood), infer approximations to the missing features using the Bayes rule and incorporate additional expert knowledge in the design of the model.

Introduction

Data quality today remains the main problem which complicates the usage of machine learning methods. Well logs can be recorded at different times and drilling conditions by different companies and tools within beds with different geological settings. This situation makes data preprocessing step, including handling missing data, input anomalies detection and standardization of well logs a crucial and very time-consuming part of any statistical model application. In order to accelerate machine learning application for real geological tasks appropriate tool for well log data preprocessing is needed.

The main goal of this work is to develop a systematic approach to work with raw well log data. To accomplish this goal, we propose to fit a simple unsupervised generative model to the input data and automate the preprocessing step using the generative model. This approach allows to detect the anomalies in the data as the regions that the model struggles to explain (i.e., samples with extremely low likelihood), infer approximations to the missing features using the Bayes rule and incorporate additional expert knowledge in the design of the model. The contributions of this work are:

- the design of a generative model based on a hidden Markov model which is capable for well log data normalization based on particular well and geological conditions,
- the development of sub-routines for anomaly detection, data imputation and the development of heuristics to facilitate the training of the generative model,
- application of the developed model on a real problem of net pay thickness autointerpretation, where we compare the effect of different preprocessing schemes on the performance of a supervised classifier.

Geological settings and input data

As a base for the presented research, we chose one of the Western Siberia mature oilfields located at Khanty-Mansiisk autonomous district and operated by Gazpromneft. The oilfield is characterized by the complex geological setting and long life during which well log data were produced by different companies with different tools and their calibration schemes. Target formation was deposited in a shallow marine environment during a transgressive system tract and composed of lithologies from medium and fine-grained sand to siltstone and mudstone. Each of sub-beds and depositional zones has different geological, geophysical and petrophysical properties without any stable marker bed which can be used as a reference for normalization. As a result, each well needs individual attention during the data preprocessing step.

Input data included the same set of logs for each of more than 350 wells (GR, SP, neutron, ILD, LLD) recorded by a different tool in different scales but with the same physical meaning.

Method and Theory

In this work, we chose the hidden Markov model as a generative model for the raw input data. The temporal structure of HMM allows to capture the vertical continuity of the observed data within the well, and discrete latent variables can act as a proxy for lithological facies. Eidsvik et al. (2004) applied hidden Markov models to infer geological attributes from a well log and Schumann (2002) proposed a well-log classification algorithm that uses the hidden states of HMM. Lindberg and Grana (2015) showed that the vertical continuity of hidden models can lead to improved facies classification results.

Unlike the previous work, we use the hidden Markov model only as a tool for data preprocessing. Besides capturing the continuity of well logs, HMMs have other appealing characteristics. First, for a Gaussian observation model used in Eidsvik et al. (2004) one can infer hidden variables based only on a subset of logs using marginals of multivariate Gaussian distribution. Then, one can approximate the missing logs based on the observation model and the inferred latent variables. This provides us with a

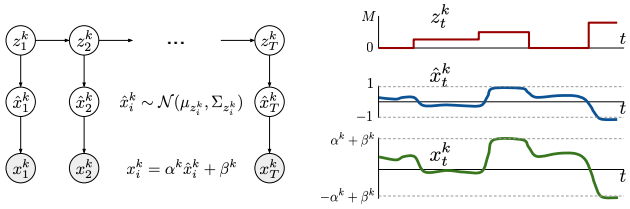


Figure 1 *Left: graphical model for the modified HMM. Grey circles represent the observed uncalibrated variables. Right: schematic representation of the generative process for data, calibrated data (blue) is produced using the hidden variables (red) and then we observe the uncalibrated data (green) obtained after a linear transformation.*

tool for data imputation that allows applying learning algorithms even on partially missing data. Second, the observation model can be further extended to incorporate additional knowledge. In our case, the logs throughout the data did not have a fixed measurement scale, and we modified the default HMM model to include log calibration as a step of the generative process (see Figure. 1).

Below we describe the proposed model and the heuristics for training in detail.

For each well $k \in \{1,\dots,K\}$ we introduce a sequence of M-valued discrete latent variables $Z^k = (z_1^k, z_2^k, \dots, z_{T_k}^k)$ to encode the essential soil properties along the well. We assume that for each well the latent variables form a Markov chain with initial probability distribution π, $p(z_1^k = i) = \pi_i$ and transition probabilities T, $p(z_t^k = i \mid z_{t-1}^k = j) = \mathrm{T}_{ij}$. We then assume that there exists a reference measurement scale, in which the logs follow Gaussian distribution with the parameters specified by the latent variable z_t^k: if $z_t^k = m$, $\hat{x}_t^k \sim \mathcal{N}(\mu_m, \Sigma_m)$. Here the hat sign in $\hat{x}_t^k$ indicates the reference measurement scale and $\mu_m \in \mathbb{R}^d, \Sigma_m \in \mathbb{R}^{d\times d}$ are the model parameters. Even though the reference measurement scale is unknown, we assume that it is constant within a well and can be recovered as a component-wise linear transformation. In other words, there exists a set of calibration parameters $\alpha^k, \beta^k \in \mathbb{R}^d$ such, that for the observed logs $X^k = (x_1^k, x_2^k, \dots, x_{T_k}^k)$ holds $x_t^k = \alpha^k \odot \hat{x}_t^k + \beta^k$. As a consequence, if $z_t^k = m$, the observed value x_t^k has Gaussian distribution with mean $\alpha^k \odot \mu_m + \beta^k$ and covariance matrix $\mathrm{diag}(\alpha^k)\Sigma_m \mathrm{diag}(\alpha^k)^T$. For the model parameters $\Theta = \left(\pi, \mathrm{T}, \{\alpha^k, \beta^k\}_{k=1}^K, \{\mu_m, \Sigma_m\}_{m=1}^M\right)$ the resulting joint of the model likelihood is

$$p(X, Z|\Theta) = \prod_{k=1}^{K} \left[\left(\pi_{z_1^k} \prod_{t=2}^{T_k} \mathrm{T}_{z_t^k, z_{t-1}^k} \right) \times \prod_{t=1}^{T_k} \mathcal{N}\left(x_t^k \mid \alpha^k \odot \mu_{z_t^k} + \beta^k, \mathrm{diag}(\alpha^k)\Sigma_{z_t^k}\mathrm{diag}(\alpha^k)^T \right). \right] \quad (1)$$

To tune the model parameters we use the Baum-Welch algorithm to compute a lower bound on the marginal likelihood $\log p(X|\Theta)$ and then apply a gradient ascent optimization scheme to maximize the lower bound with respect to Θ.

By design, the optimal model parameters are not unique. Indeed, the calibration parameters (α^k, β^k) can be adjusted to different choices of the reference measure scale. For example, the change of scale for the Gaussian parameters $(\mu_m, \Sigma_m) \to (C\mu_m, C^2\Sigma_m)$ together with the calibration parameters update $(\alpha^k, \beta^k) \to (\frac{\alpha^k}{C}, \beta^k)$ has no effect of the joint likelihood and, as a result, the model outputs. We do not put any restrictions on the reference measure scale during training and adjust it after the training.

The training procedure is prone to producing sub-optimal models. Therefore, to avoid poor local optima during the training, we initialize the calibration parameters α^k and β^k with the standard deviation and mean of logs computed across the k-th well. Additionally, to initialize μ_m we apply mean-std scaler to logs and then run K-means on the standardized log values. We use cluster means as the initial values of μ_m and matrix $\sigma^2 I$ as the initial values of Σ_m for a small σ.

Results

As we mentioned in the introduction, well logs data is not clean, and the set of performed measurements can be different in different wells. As a first step, we trained the model on a set of wells for which all five well logs under consideration present. Figure 2 presents an example with results for one of them.

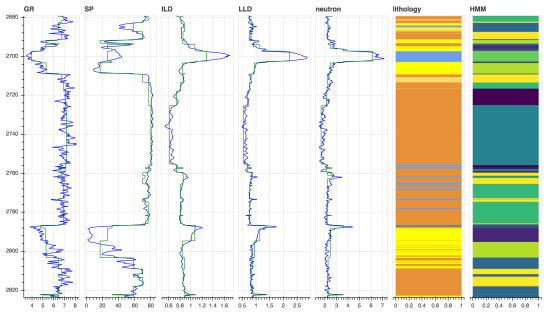


Figure 2 *Example of fitted well. Blue - input well logs, green - data generated by model, lithology - human interpretation, HMM - hidden states from HMM model.*

The number of hidden states in Hidden Markov model is a tunable hyperparameter. We tested different values in the range from 5 to 100. Choice of the number of states exhibits a trade-off: a larger number of hidden states results in a better fit of the data but leads to less interpretable states. Experiments showed that good numbers are 10 for more interpretable model and 30 for a more accurate model. Numbers of states beyond 30 did not lead to notable improvements.

We used two methods to measure the quality of the trained model. Firstly, it is natural to expect that hidden states learned by the model should correspond to some lithological properties. Therefore we compared the states with lithology labels from human interpretation. Contingency matrix for this comparison is presented in figure 3. There is no one-to-one correspondence because the number of lithology types and hidden states is different, but in general, the correspondence looks quite good.

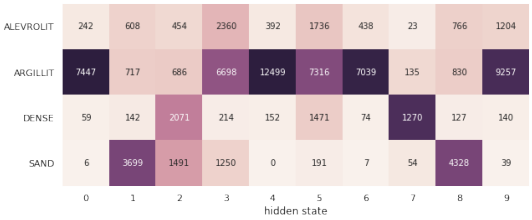


Figure 3 *Contingency matrix between human lithology labels and hidden states from model. Number in each cell represents number of data points which fall in it.*

Additionally, using the generative model, we recovered the values of the well logs from the learned hidden states. For a given hidden state m, we chose the value of well log to be equal to $\alpha^k \odot \mu_m + \beta^k$. Comparison of values generated by the model with true values is a good test of model fit. We used the coefficient of determination R^2 as a metric for this comparison. Table 1 presents calculated R^2 for the considered logs. The average coefficient of determination across all logs was $R^2 = 0.79$.

well log	GR	SP	ILD	LLD	neutron
R^2	0.68	0.87	0.91	0.78	0.72

Table 1 *Coefficient of determination for different well logs.*

One of the benefits of this approach is that the trained model can be applied to automatic detection of anomalies in the input well logs. Poor fit of the data for a given well compared to other wells strongly

indicates that the well contains an anomaly. If we look at per well R^2 (figure 4), we can see that there are several wells with particularly low metric value. Manual inspection of these wells showed that these cases are real anomalies. Severe anomalies can hurt further modeling and should be discarded or handled manually.

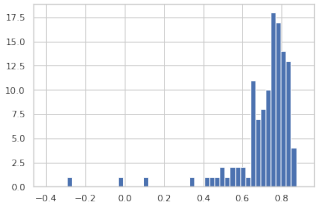


Figure 4 *Distribution of per well coefficient of determination* R^2

We further tested the model as a preprocessing step for the task of net pay intervals classification described in Belozerov et al. (2018). The task is to predict for each point in a well whether it belongs to net pay interval or not. We compare two approaches to the preprocessing. The first is per well scaling using the mean and the standard deviation of each curve. The second is obtaining calibration parameters α^k and β^k from the HMM for rescaling. Recurrent neural networks (GRU - Gated Recurrent Unit) were used for classification because they are suitable for such structured as a sequence data and show better performance than more classical non-deep learning models. Due to the imbalance of the target classes, we use F-score (harmonic mean of precision and recall) to measure the performance. In our experiments, rescaling with HMM lead to an improvement in performance. F-score of a simple scaler was 0.72, and for HMM F-score was 0.74.

Finally, we applied the model for data imputation. Some of the wells in the data did not contain ILD and LLD logs. Nevertheless, the HMM model can infer hidden states in these wells only from the observed logs and fill missing curves from using the inferred states. We tested this approach on the same task of net pay intervals classification. F-score on wells with missing logs was 0.37 for training without imputation. It improved to 0.56 after imputation step, showing the potential of such an approach.

Conclusions

In this work, we applied a generative model based on Hidden Markov model to tasks of well log data processing. The proposed algorithm showed capability for the solution of a wide range of problems related to logging data which previously were time-consuming and in many cases could not be automated. These tasks include well logs reconstruction, data normalization, anomaly detection.

Automating of pointed out tasks by the presented approach can dramatically increase the speed of research and application of machine learning models for well log data and improve its economic effectiveness.

References

Belozerov, B., Egorov, D., Bukhanov, N., Zakirov, A., Osmonalieva, O., Golitsyna, M., Reshytko, A., Semenikhin, A., Shindin, E. and Lipets, V. [2018] Automatic Well Log Analysis Across Priobskoe Field Using Machine Learning Methods. *Society of Petroleum Engineers 18RPTC*.

Eidsvik, J., Mukerji, T. and Switzer, P. [2004] Estimation of geological attributes from a well log: an application of hidden Markov chains. *Mathematical Geology*, **36**(3), 379–397.

Lindberg, D.V. and Grana, D. [2015] Petro-elastic log-facies classification using the expectation–maximization algorithm and hidden Markov models. *Mathematical Geosciences*, **47**(6), 719–752.

Schumann, A. [2002] Hidden Markov models for lithological well log classification. *Terra Nostra*, **4**, 373–378.