LIFELONG LEARNING BY ADJUSTING PRIORS

Anonymous authors
Paper under double-blind review

ABSTRACT

In representational lifelong learning an agent aims to continually learn to solve novel tasks while updating its representation in light of previous tasks. Under the assumption that future tasks are 'related' to previous tasks, representations should be learned in such a way that they capture the common structure across learned tasks, while allowing the learner sufficient flexibility to adapt to novel aspects of a new task. We develop a framework for lifelong learning in deep neural networks that is based on generalization bounds, developed within the PAC-Bayes framework. Learning takes place through the construction of a distribution over networks based on the tasks seen so far, and its utilization for learning a new task. Thus, prior knowledge is incorporated through setting a history-dependent prior for novel tasks. We develop a gradient-based algorithm implementing these ideas, based on minimizing an objective function motivated by generalization bounds, and demonstrate its effectiveness through numerical examples. In addition to establishing the improved performance available through lifelong learning, we demonstrate the intuitive way by which prior information is manifested at different levels of the network.

1 INTRODUCTION

Learning from examples is the process of inferring a general rule from a finite set of examples. It is well known in statistics (e.g., Devroye et al. (1996)) that learning cannot take place without prior assumptions. This idea has led in Machine Learning to the notion of inductive bias (Mitchell, 1980). Recent work in deep neural networks has achieved significant success in using prior knowledge in the implementation of structural constraints, e.g. the use of convolutions and weight sharing as building blocks, capturing the translational invariance of image classification. However, in general the relevant prior information for a given task is not always clear, and there is a need for building prior knowledge through learning from previous interactions with the world. This idea is captured by the concept of *lifelong learning*, where an agent continually learns through interacting with the world, transferring the knowledge acquired along its path to any new task it encounters. This notion has been formulated by Baxter (2000) in a clear and simple context of 'task-environment'. In analogy to the standard single-task learning in which data is sampled from an unknown distribution, Baxter suggested to model a lifelong learning setting as if tasks are sampled from an unknown task distribution (environment), so that knowledge acquired from previous tasks can be used in order to improve performance on a novel task. Baxter's work not only provided an interesting and mathematically precise perspective for lifelong learning, but also provided generalization bounds demonstrating the potential improvement in performance due to prior knowledge. Baxter's seminal work, has led to a large number of extensions and developments.

In this contribution we work within the framework formulated by Baxter (2000), and, following the setup in Pentina & Lampert (2014), provide generalization error bounds within the PAC-Bayes framework. These bounds are then used to develop a practical learning algorithm that is applied to neural networks, demonstrating the utility of the approach. The main contributions of this work are the following. *(i)* An extension of the theoretical framework of Pentina & Lampert (2014), allowing us to apply any single-task PAC-Bayes bound in the context of lifelong learning. *(ii)* Developing a learning algorithm within this general framework and its implementation using probabilistic feed-forward neural networks. This yields transfer of knowledge between tasks through constraining the prior distribution on a learning network. *(iii)* Empirical demonstration of the performance enhancement compared to naive approaches.

As noted above, Baxter (2000) provided a basic mathematical formulation and initial results for lifelong learning. While there have been many developments in this field since then (e.g., Andrychowicz et al. (2016); Edwards & Storkey (2016); Finn et al. (2017); Ravi & Larochelle (2016)), most of them were not based on generalization error bounds which is the focus of the present work. An elegant extension of generalization error bounds to lifelong learning was provided by Pentina & Lampert (2014), mentioned above (recently extended in Pentina & Lampert (2015)). Their work, however, did not provide a practical algorithm applicable to deep neural networks. More recently, Dziugaite & Roy (2017) developed a single-task algorithm based on PAC-Bayes bounds that was demonstrated to yield good performance in simple classification tasks. Other recent theoretical approaches to lifelong or multitask learning (e.g. Alquier et al. (2017); Maurer et al. (2016)) provide increasingly general bounds but have not led directly to practical learning algorithms.

2 BACKGROUND: PAC-BAYES LEARNING

In the standard setting for supervised learning a set of (usually) independent pairs of input/output samples $S = \{(x_i, y_i)\}_{i=1}^m$ are given, each sample drawn from an *unknown* probability distribution D, namely $(x_i, y_i) \sim D$. We will use the notation $S \sim D^m$ to denote the distribution over the full sample. The usual learning goal is, based on S to find a function $h \in \mathcal{H}$, where $\mathcal{H}$ is the so-called hypothesis space, that minimizes the expected loss function $\mathbb{E}\ell(h, z)$, where $z = (x, y)$ [1]. As the distribution D is unknown, learning consists of selecting an appropriate h based on the sample S. In classification $\mathcal{H}$ is a space of classifiers mapping the input space to a finite set of classes. As noted in the Introduction, an inductive bias is required for effective learning. While in the standard approach to learning, described in the previous paragraph, one usually selects a single classifier (e.g., the one minimizing the empirical error), the PAC-Bayes framework, first formulated by McAllester (1999), considers the construction of a complete probability distribution over $\mathcal{H}$, and the selection of a single hypothesis $h \in \mathcal{H}$ based on this distribution. Since this distribution depends on the data it is referred to as a *posterior distribution* and will be denoted by Q. We note that while the term 'posterior' has a Bayesian connotation, the framework is not necessarily Bayesian, and the posterior does not need to be related to the prior through the likelihood function as in standard Bayesian analysis. The PAC-Bayes framework has been widely studied in recent years, and has given rise to significant flexibility in learning, and, more importantly, to some of the best generalization bounds available Audibert (2010); McAllester (2013); Lever et al. (2013). The framework has been recently extended to the lifelong learning setting by Pentina & Lampert (2014), and will be extended and applied to neural networks in the present contribution.

2.1 SINGLE-TASK PROBLEM FORMULATION

Following the notation introduced above we define the generalization error and the empirical error used in the standard learning setting,

$$er\left(h, D\right) \triangleq \mathop{\mathbb{E}}_{z\sim D} \ell(h, z) \quad ; \quad \widehat{er}\left(h, S\right) \triangleq \frac{1}{m}\sum_{j=1}^{m} \ell\left(h, z_i\right) \quad (h \in \mathcal{H}) \,. \tag{1}$$

Since the distribution D is unknown, $er\left(h, D\right)$ cannot be directly computed.

PAC-Bayesian learning In the PAC-Bayesian setting the learner outputs a distribution over the entire hypothesis space $\mathcal{H}$, i.e, the goal is to provide a *posterior* distribution $Q \in \mathcal{M}$, where $\mathcal{M}$ denotes the set of distributions over $\mathcal{H}$. The expected (over $\mathcal{H}$) *generalization error* and *empirical error* are then given in this setting by averaging (1) over the posterior distribution,

$$er\left(Q, D\right) \triangleq \mathop{\mathbb{E}}_{h\sim Q} \mathop{\mathbb{E}}_{z\sim D} \ell\left(h, z\right) \quad ; \quad \widehat{er}\left(Q, S\right) \triangleq \mathop{\mathbb{E}}_{h\sim Q} \frac{1}{m}\sum_{j=1}^{m} \ell\left(h, z_j\right) \quad (Q \in \mathcal{M}). \tag{2}$$

[1] Note that the framework is not limited to supervised learning and can also handle unsupervised learning.

This average describes a *Gibbs prediction* procedure - first drawing a hypothesis h from Q then applying it on the sample z. Similarly to (1), the generalization error $er(Q, D)$ cannot be directly computed since D is unknown.

2.2 PAC-BAYESIAN GENERALIZATION BOUND

In this section we introduce a PAC-Bayesian bound for the single-task setting. The bound will also serve us for the lifelong-learning setting in the next sections. PAC-Bayesian bounds are based on specifying some reference distribution $P \in \mathcal{M}$. P is called the *'prior'* since it must not depend on the observed data S. The distribution over hypotheses Q which is provided as an output from the learning process is called the posterior (since it is allowed to depended on S). [2]

Many PAC-Bayesian style generalization bounds have appeared in the literature. Since our lifelong-learning framework can work with different single-task bounds, we will first state a generic form.

Proposition 1 (Generic form of single-task PAC-Bayes bounds)**.** *Let $P \in \mathcal{M}$ be some prior distribution over $\mathcal{H}$, and let $\Psi()$ be a complexity measure (see below). Then for any $\delta \in (0, 1]$,*

$$\mathbb{P}_{S\sim D^m}\left\{er(Q, D) \leq \widehat{er}(Q, S) + \Psi(S, m, Q, P, \delta), \forall Q \in \mathcal{M}\right\} \geq 1 - \delta \,. \tag{3}$$

For example the original PAC-Bayes theorem (McAllester, 1999) is formulated using the complexity term $\Psi = \left[\left(D_{KL}(Q||P) + \log(2m/\delta)\right)/(2m-1)\right]^{1/2}$, where $D_{KL}(Q||P)$ is the Kullback-Leibler Divergence (KLD), $D_{KL}(Q||P) \triangleq \mathop{\mathbb{E}}_{h\sim Q} \log \frac{Q(h)}{P(h)}$. See section 7.1 in the appendix for a summary of possible complexity terms, obtained from well-known PAC-Bayesian single task bounds.

The bound (3) can be interpreted as stating that with high probability the expected error $er(Q, D)$ is upper bounded by the empirical error plus a complexity term. Since, with high probability, the bound holds for all $Q \in \mathcal{M}$ (uniformly), we can choose Q after observing the data S. By choosing Q that minimizes the bound we will get a learning algorithm with generalization guarantees. Note that PAC-Bayesian bounds express a trade-off between fitting the data (empirical error) and a complexity/regularization term (distance from prior) which encourages selecting a 'simple' hypothesis, namely one similar to the prior. The contribution of the prior-dependent regularization term to the objective is more significant for a smaller data set. In all bounds, for asymptotically large sample size m, the complexity $\Psi()$ converges to zero. Specific rates of convergence are further discussed in the appendix. The specific choice of P affects the bound's tightness and so should express prior knowledge about the problem. Generally, we want the prior to be close to posteriors which can achieve low training error. For example, we may want to use a prior that prefers simpler hypotheses (Occam's razor). In the lifelong-learning scenario will show how to learn the prior based on past experience from other learning tasks.

In general, the bound might not be tight, and can even be vacuous, i.e., greater than the maximal value of the loss. However, Dziugaite & Roy (2017) recently showed that a PAC-Bayesian bound can achieve non-vacuous values with deep-networks and real data sets. In this work our focus is on deriving an algorithm for lifelong-learning, rather than on actual calculation of the bound. We expect that even if the numerical value of the bound is vacuous, it still captures the behavior of the generalization error and so minimizing the bound is a good learning strategy.

3 PAC-BAYESIAN LIFELONG-LEARNING

In this section we introduce the lifelong-learning setting. In this setting a lifelong-learning agent observes several 'training' tasks from the same task environment. The lifelong-learner must extract some common knowledge from these tasks, which will be used for learning new tasks from the same environment. In the literature this setting is often called learning-to-learn, meta-learning or lifelong-learning. We will formulate the problem and provide a generalization bound which will later lead to a practical algorithm. Our work extends Pentina & Lampert (2014) and establishes a more general

[2] As noted above, the terms 'prior' and 'posterior' might be misleading, since, this is not a Bayesian inference setting (the prior an posterior are not connected trough the Bayes rule). However, PAC-Bayes and Bayesian analysis have interesting and practical connections, as we will see in the next sections (see also Germain et al. (2016)).

bound. Furthermore, we will demonstrate how to apply this result practically in non-linear deep models using stochastic learning.

3.1 META-LEARNING PROBLEM FORMULATION

We assume all tasks share the sample space $\mathcal{Z}$, hypothesis space $\mathcal{H}$ and loss function $\ell : \mathcal{H} \times \mathcal{Z} \to [0, 1]$. The learning tasks differ in the unknown sample distribution D_t associated with each task t. The lifelong-learning agent observes the training sets $S_1, ..., S_n$ corresponding to n different tasks. The number of samples in task i is denoted by m_i. Each observed dataset S_i is assumed to be generated from an unknown sample distribution $S_i \sim D_i^{m_i}$. As in Baxter (2000), we assume that the sample distributions D_i are generated $i.i.d.$ from an unknown tasks distribution τ.

The goal of the lifelong-learner is to extract some knowledge from the observed tasks that will be used as prior knowledge for learning new (yet unobserved) tasks from τ. The prior knowledge comes in the form of a distribution over hypotheses, $P \in \mathcal{M}$. When learning a new task, the learner uses the observed task's data S and the prior P to output a posterior distribution $Q(S, P)$ over $\mathcal{H}$. We assume that all tasks are learned via the same learning process. Namely, for a given S and P there is a specific output $Q(S, P)$. Hence $Q()$ is a function: $Q : \mathcal{Z}^{m_i} \times \mathcal{M} \to \mathcal{M}$. [3]

The quality of a prior P is measured by the expected loss when using it to learn new tasks, as defined by,

$$\mathit{er}\left(P, \tau\right) \triangleq \mathop{\mathbb{E}}_{D\sim\tau} \mathop{\mathbb{E}}_{S\sim D^m} \mathop{\mathbb{E}}_{h\sim Q(S,P)} \mathop{\mathbb{E}}_{z\sim D} \ell(h, z). \tag{4}$$

Since we want to prove a PAC-Bayes style bound for lifelong-learning, we assume that the lifelong-learner does not select a single prior P, but instead infers a distribution $\mathcal{Q}$ over all prior distributions in $\mathcal{M}$. [4]. Since $\mathcal{Q}$ is inferred *after* observing the tasks, it is called the hyper-posterior distribution, and serves as a prior for a new task, i.e., when learning a new task, the learner draws a prior from $\mathcal{Q}$ and then uses it for learning.

Ideally, the performance of the hyper-posterior $\mathcal{Q}$ is measured by the expected generalization loss of learning new tasks using priors generated from $\mathcal{Q}$ This quantity is denoted as the *transfer error*

$$\mathit{er}\left(\mathcal{Q}, \tau\right) \triangleq \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \mathit{er}\left(P, \tau\right). \tag{5}$$

While $\mathit{er}\left(\mathcal{Q}, \tau\right)$ is not computable, we can however evaluate the *empirical multi-task error*

$$\widehat{\mathit{er}}\left(\mathcal{Q}, S_1, ..., S_n\right) \triangleq \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} \frac{1}{n} \sum_{i=1}^{n} \widehat{\mathit{er}}\left(Q(S_i, P), S_i\right). \tag{6}$$

Although $\mathit{er}\left(\mathcal{Q}\right)$ cannot be evaluated, we will prove a PAC-Bayes style upper bound on it, that can be minimized over $\mathcal{Q}$.

In the single-task PAC-Bayes setting one selects a prior $P \in \mathcal{M}$ before seeing the data, and updates it to a posterior $Q \in \mathcal{M}$ after observing the training data. In the present lifelong setup, following the framework in Pentina & Lampert (2014), one selects an initial hyper-prior distribution $\mathcal{P}$, essentially a distribution over prior distributions P, and, following the observation of the data from all tasks, updates it to a hyper-posterior distribution $\mathcal{Q}$. As a simple example, assume the initial prior P is a Gaussian distribution over neural network weights, characterized by a mean and covariance. A hyper distribution would correspond in this case to a distribution over the mean and covariance of P.

3.2 LIFELONG-LEARNING PAC-BAYESIAN BOUND

The lifelong-learning bound in Theorem 1 holds with any single-task complexity term $\Psi(S, m, Q, P, \delta)$ which satisfies Proposition 1. The theorem is proved in the appendix (7.3).

[3] In the next section we will use stochastic optimization methods as learning algorithms, but we can assume convergence to a same solution for any execution with a given S and P.

[4] After proving the bound, we will simplify the lifelong-learning objective into choosing a single optimal prior.

Theorem 1 (General form of the lifelong-learning PAC-Bayes bound)**.** *Let $Q()$ be a mapping $Q : \mathcal{Z}^{m_i} \times \mathcal{M} \to \mathcal{M}$, and let $\mathcal{P}$ be some predefined hyper-prior distribution. Then for any $\delta \in (0, 1]$*

$$\mathbb{P}_{D_i \sim \tau, S_i \sim D_i^{m_i}, i=1,...,n} \Bigg\{ er\left(\mathcal{Q}, \tau\right) \leq \widehat{er}\left(\mathcal{Q}, S_1, ..., S_n\right) + \tag{7}$$

$$\frac{1}{n} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{P \sim \mathcal{Q}} \Psi(S_i, m_i, Q(S_i, P), P, \delta_i) + \frac{1}{\sqrt{n}} \left(D_{KL}(\mathcal{Q}||\mathcal{P}) + \log \frac{1}{\delta_0} + \frac{1}{8} \right), \forall \mathcal{Q} \Bigg\} \geq 1 - \delta,$$

for any $\{\delta_i\}_{i=1}^n$ that satisfy $\delta_i \in (0, 1]$ and : $\delta_0 + 1 - \prod_{i=1}^{n}(1 - \delta_i) = \delta$. [5]

Notice that the transfer error (7) is bounded by the empirical multi-task error (6) plus two complexity terms. The first is the average intra-task complexity of the observed tasks. This term converges to zero in the limit of large number of samples m_i in each task. The second is a task-environment level complexity term. It tends to zero as n (the number of observed tasks) grows. In comparison to Pentina & Lampert (2014), our lifelong-learning bound is more general and can work with different single-task complexity terms. It also takes into account the specific number of samples in each observed task (instead of the harmonic mean).

4 LIFELONG-LEARNING ALGORITHM

As in the single-task case, the bound of Theorem 1 can be evaluated from the training data and so can serve as a minimization objective for a principled lifelong-learning algorithm. Since the bound holds uniformly for all $\mathcal{Q}$, it is ensured to hold also for the inferred optimal $\mathcal{Q}^*$. In this section we will derive a practical learning procedure that can applied to a large family of differentiable models, including deep neural networks.

4.1 FROM BOUND TO LEARNING OBJECTIVE

The lifelong-learning objective is obtained from the bound (7) by omitting constants independent of $\mathcal{Q}$, and can be compactly written as

$$J(\mathcal{Q}) = \frac{1}{n} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{P \sim \mathcal{Q}} J_i(P) + \frac{1}{\sqrt{n}} D_{KL}(\mathcal{Q}||\mathcal{P}), \tag{8}$$

where we defined $J_i(P) \triangleq \widehat{er}\left(Q(S_i, P), S_i\right) + \Psi(S_i, m_i, Q(S_i, P), P, \delta_i)$. Notice that given a fixed prior P, $J_i(P)$ is the single-task PAC-Bayes bound for the i-th task.

After minimizing $J(\mathcal{Q})$ we obtain an optimal hyper-posterior $\mathcal{Q}^*$ in the sense that learning a new task with a prior generated from $\mathcal{Q}^*$ minimizes an upper bound (up to a constant) on the expected error. While optimizing a distribution over distributions may seem intractable, one way to approach this problem would be through parameterizing distributions with hyper-parameters and using hyper-distributions for these hyper-parameters. Such an approach was successfully used within a Bayesian setting for deep neural networks in Hernández-Lobato & Adams (2015). In this work we consider a simpler computational strategy, by searching for a *single* optimal prior Q^* for learning new tasks. In other words, we restrict the class of hyper-posterior distributions to singular distributions (delta functions).

Single prior form Unfortunately, we cannot directly set the PDF of $\mathcal{Q}$ to be a delta function at a single prior, since then the KLD term, $D_{KL}(\mathcal{Q}||\mathcal{P})$, would diverge. Using simple arguments, and neglecting constant terms, we establish in Section 7.4 in the appendix the following objective for optimization,

$$J(\theta_P) = \frac{1}{n} \sum_{i=1}^{n} J_i(\theta_P) + \frac{1}{\sqrt{n}} \log \frac{1}{\mathcal{P}(\theta_P)}. \tag{9}$$

We have abused notation by denoting $J(\theta_P) = J(\mathcal{Q})$. The objective $J(\theta_P)$ has a simple interpretation, the average of the single-task PAC-Bayes bounds of the observed tasks (given the prior parameters θ_P) plus a regularization term in the form of a hyper-prior.

[5] For example, we can choose: $\delta_0 = \frac{\delta}{2}$ and $\delta_i = 1 - \sqrt[n]{1 - \frac{\delta}{2}}, i = 1, ..., n$.

4.2 JOINT OPTIMIZATION

Theorem 1 allows us to choose *any* single-task learning procedure $Q(S_i, P) : \mathcal{Z}^{m_i} \times \mathcal{M} \to \mathcal{M}$ to infer a posterior. We will use a procedure which minimizes a single task PAC-Bayes bound due to the following advantages: *(i)* It minimizes a bound on the generalization error of the observed task. *(ii)* It uses the prior knowledge gained from the prior P to get a tighter bound and a better learning objective. *(iii)* As will be shown next, formulating the single task learning as an optimization problem enables joint learning of the shared prior and the task posteriors.

To formulate the single-task learning as an optimization problem, we parametrize the posterior of the i-th task with the vector $\phi_i \in \mathbb{R}^{N_Q}$, denoted by $Q_{\phi_i}(S_i, P)$. The single-task learning algorithm can be formulated as $\phi_i^* = \operatorname{argmin}_{\phi_i} J_i(\theta_P, \phi_i)$, where $J_i(\theta_P, \phi_i)$ is the single-task PAC-Bayes bound of the i-th observed task, evaluated with the prior parameters θ_P and the posterior parameters ϕ_i. The lifelong-learning problem of minimizing $J(\theta_P)$ over θ_P can now be written more explicitly,

$$\min_{\theta_P, \phi_1, ..., \phi_n} \left\{ \frac{1}{n} \sum_{i=1}^{n} J_i(\theta_P, \phi_i) + \frac{1}{\sqrt{n}} \log \frac{1}{\mathcal{P}(\theta_P)} \right\}. \tag{10}$$

4.3 DISTRIBUTIONS MODEL

In this section we make the lifelong-learning optimization problem (10) more explicit by defining a model for the posterior and prior distributions. First, we define the hypothesis class $\mathcal{H}$ as a family of functions parameterized by a *weight vector* $\left\{h_w : w \in \mathbb{R}^d\right\}$. Given this parameterization, the posterior and prior are distributions over $\mathbb{R}^d$.

We will present an algorithm for any differentiable model [6], but our aim is to use neural network (NN) architectures. In fact, we will use Stochastic NNs (Graves, 2011; Blundell et al., 2015) since in our setting the weights are random and we are optimizing their posterior distribution. The techniques presented next will be mostly based on Blundell et al. (2015).

Next we define the posteriors Q_{ϕ_i} and the prior P_{θ_P} as factorized Gaussian distributions[7],

$$P_{\theta_P}(w) = \prod_{k=1}^{d} \mathcal{N}\left(w_k; \mu_{P,k}, \sigma_{P,k}^2\right) \quad ; \quad Q_{\phi_i}(w) = \prod_{k=1}^{d} \mathcal{N}\left(w_k; \mu_{i,k}, \sigma_{i,k}^2\right),\ i = 1, ..., n. \tag{11}$$

The prior parameters vector $\theta_P \in \mathbb{R}^{2d}$ is composed of the elements $\mu_{P,k}$ and $\rho_{P,k} = \log \sigma_{P,k}^2, k = 1, ..., d$.[8] The posterior parameters $\phi_i \in \mathbb{R}^{2d}$ have the same structure. Since we aim to use deep models where d could be in the order of millions, distributions with more parameters might be impractical.

Since Q_{ϕ_i} and P_{θ_P} are factorized Gaussian distributions the KLD takes a simple analytic form,

$$D_{KL}(Q_{\phi_i} || P_{\theta_P}) = \frac{1}{2} \sum_{k=1}^{d} \left\{ \log \frac{\sigma_{P,k}^2}{\sigma_{i,k}^2} + \frac{\sigma_{i,k}^2 + \left(\mu_{i,k} - \mu_{P,k}\right)^2}{\sigma_{P,k}^2} - 1 \right\}. \tag{12}$$

As for the hyper-prior $\mathcal{P}(\theta_P)$, we choose a simple factorized Laplace distribution $\log 1/\mathcal{P}\left(\theta_P\right) = \kappa \left\|\theta_P\right\|_1 + c$, where κ is some predefined constant. The Laplace distribution encourages sparsity which may help generalization.

4.4 OPTIMIZATION TECHNIQUE

As an underlying optimization method, we will use stochastic gradient descent (SGD)[9]. In each iteration, the algorithm takes a parameter step in a direction of an estimated negative gradient. As is

[6] The only assumption on $\left\{h_w : w \in \mathbb{R}^d\right\}$ is that the loss function $\ell(h_w, z)$ is differentiable w.r.t w.

[7] This choice makes optimization easier, but in principle we can use other distributions as long as the PDF is differentiable w.r.t the parameters.

[8] Note that we use $\rho = \log \sigma^2$ as a parameter in order to keep the parameters unconstrained (while $\sigma^2 = \exp(\rho)$ is guaranteed to be strictly positive).

[9] Or some other variant of SGD.

well known, lower variance facilitates convergence and its speed. Recall that each single-task bound is composed of an empirical error term and a complexity term as in Proposition 1. The complexity term $\Psi()$ is a simple function of $D_{KL}(Q_{\phi_i}||P_{\theta_P})$ 12, which can easily be differentiated analytically. However, evaluating the gradient of the empirical error term is more challenging.

Recall the definition of the empirical error, $\widehat{er}\left(Q_{\phi_i}, S_i\right) = \mathbb{E}_{Q_{\phi_i}}(1/m_i)\sum_{j=1}^{m_i}\ell\left(h_w, z_j\right)$. This term poses two major challenges. *(i)* The data set S_i could be very large making it expensive to cycle over all the m_i samples. *(ii)* The term $\ell\left(h_w, z_j\right)$ might be highly non-linear in w, rendering the expectation intractable. Still, we can get an unbiased and low variance estimate of the gradient.

First, instead of using all of the data for each gradient estimation we will use a randomly sampled mini-batch $S_i' \subset S_i$. Next, we require an estimate of a gradient of the form $\nabla_\phi \mathop{\mathbb{E}}_{w\sim Q_\phi} f(w)$ which is a common problem in machine learning. We will use the 're-parametrization trick' (Rezende et al., 2014; Kingma & Welling, 2013) which is an efficient and low variance method. The re-parametrization trick is easily applicable in our setup since we are using Gaussian distributions. The trick is based on describing the Gaussian distribution $w \sim Q_{\phi_i}$ (11) as first drawing $\varepsilon \sim \mathcal{N}\left(\bar{0}, I_{d\times d}\right)$ [10] and then applying the deterministic function $w(\phi_i, \varepsilon) = \mu_i + \sigma_i \odot \varepsilon$ (where $\odot$ is an element-wise multiplication).

Therefore, we can switch the order of gradient and expectation to get

$$\nabla_\phi \mathop{\mathbb{E}}_{w\sim Q_\phi} f(w) = \nabla_\phi \mathop{\mathbb{E}}_{\varepsilon\sim\mathcal{N}(\bar{0}, I_{d\times d})} f(w(\phi_i, \varepsilon)) = \mathop{\mathbb{E}}_{\varepsilon\sim\mathcal{N}(\bar{0}, I_{d\times d})} \nabla_\phi f(w(\phi_i, \varepsilon)).$$

The expectation can be approximated by averaging a small number of Monte-Carlo samples with reasonable accuracy. For a fixed sampled ε, the gradient $\nabla_\phi f(w(\phi_i, \varepsilon))$ is easily computable with backpropagation.

In summary, the Lifelong learning by Adjusting Priors (LAP) algorithm is composed of two phases In the first phase (Algorithm 1 termed "meta-training") several observed "training tasks" are used to learn a prior. In the second phase (Algorithm 2, termed "meta-testing") the previously learned prior is used for the learning of a new task (which was unobserved in the first phase). Note that the first phase can be used independently as a multi-task learning method. Both algorithms are described in pseudo-code in the appendix (section 7.5).

5 Experimental Results

In this section we implement our method in several experiments [11]. We focus on a conceptual demonstration, rather than an extensive comparison with other algorithms, which is deferred to future work.

5.1 Toy Example

To illustrate the setup visually, we will consider a simple to toy example of a 2D estimation problem. In each task, the goal is to estimate the mean of the data generating distribution. In this setup, the samples z and the hypotheses h are vectors in $\mathbb{R}^2$. As a loss function we will use the Euclidean distance, $\ell(h, z) = \|h - z\|_2^2$. We artificially create the data of each task by generating 50 samples from the appropriate distribution: $\mathcal{N}\left((2,1)^\top, 0.1^2 I_{2\times 2}\right)$ in task 1, and $\mathcal{N}\left((4,1)^\top, 0.1^2 I_{2\times 2}\right)$ in task 2. The prior and posteriors are 2D factorized Gaussian distributions, $P = \mathcal{N}\left(\mu_P, \mathrm{diag}(\sigma_P^2)\right)$ and $Q_i = \mathcal{N}\left(\mu_i, \mathrm{diag}(\sigma_i^2)\right), i = 1, 2$. The hyper-prior term is $\kappa\left(\|\mu_P\|_1 + \|\sigma_P\|_1\right)$ with $\kappa = 10^{-6}$.

We run Algorithm 1 (meta-training) with complexity terms according to Theorem 2. As seen in Figure 1, the learned prior (namely, the prior learned from the two tasks) and single-task posteriors can be explained intuitively. First, the posteriors are located close to the ground truth means of each

[10] In fact, we will use the "**local** re-parameterization trick" (Kingma et al., 2015) in which we sample a different ε for each data point in the batch, which reduces the variance of the estimate. To make the computation more efficient with neural-networks, the random number generation is performed w.r.t the activations instead of the weights (see Kingma et al. (2015) for more details.).

[11] Our implementation uses the PyTorch framework. The code for reproducing all the experiments can be found in the GitHub repository: `https://github.com/ML-Paper/lifelong-learning`

task, with relatively small uncertainty covariance. Second, the learned prior is located in the middle between the two posteriors, and its covariance is larger in the first dimension. This is intuitively reasonable since the prior learned that tasks are likely to have values of around 1 in dimension 2 and values around 3 in the dimension 1, but with larger variance. Thus, new similar tasks can be learned using this prior with fewer samples.

5.1.1 CLASSIFICATION EXAMPLE

In this section we demonstrate the performance of our transfer method with image classification tasks solved by deep neural networks. In image classification, the data samples, $z = (x, y)$, are pairs of a an image, x, and a label, y. The hypothesis class $\left\{h_w : w \in \mathbb{R}^d\right\}$ is a the set of neural networks with a given architecture (which will be specified later). As a loss function $\ell(h_w, z)$ we will use the cross-entropy loss.

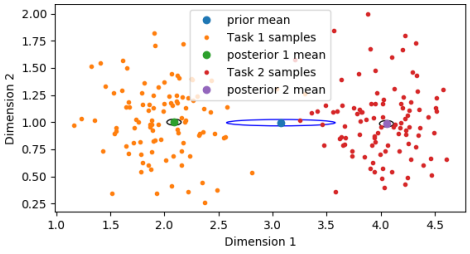


Figure 1: **Toy example:** the orange are red dots are the samples of task 1 and 2, respectively, and the green and purple dots are the means of the posteriors of task 1 and 2, respectively. The mean of the prior is a blue dot. The ellipse around each distribution's mean represents the covariance matrix.

We conduct two experiments with two different task environments. In the first environment, each task is created by a random permutation of the labels of the MNIST dataset (LeCun, 1998). In the second, each task is generated with a different permutation of the image pixels (as has been done in Goodfellow et al. (2013); Kirkpatrick et al. (2017). In both experiments, the meta-training set is composed of 5 tasks from the environment, each with $60,000$ training examples. Following the meta-training phase, the learned prior is used to learn a new meta-test task with fewer training samples $(2,000)$. The network architecture used for the permuted-labels experiment is a small CNN with 2 conv-layers and one additional fully connected hidden layer. In the the permuted-pixels experiment we used a fully-connected network with 3 hidden layers. See section 7.6 for more implementation details.

We compare the generalization performance of the LAP algorithm to several alternative methods for learning the new (meta-test) task:

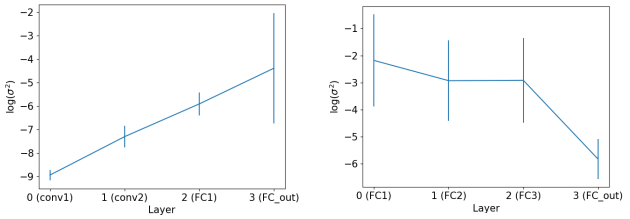
- **Scratch-standard**: standard learning from scratch.
- **Scratch-stochastic**: stochastic learning from scratch (with no prior/complexity term).
- **Warm-start-transfer**: Standard learning with initial weights taken from the standard learning of a single task from the meta-train set (with $60,000$ examples).
- **Oracle-transfer**: Same as the previous method, but some of the layers are frozen (unchanged from their initial value). In the permuted-labels experiment all layers except the output are frozen and vice versa in the permuted-pixels experiment. Note that in this method we are manually inserting prior knowledge based on our familiarity with the task environment. Therefore this method can be considered an "oracle".

As can be seen in Table 1, in both experiments the LAP transfer method improves considerably over learning from scratch and over the naive warm-start transfer method. The LAP algorithm also improves over the "oracle" transfer in the permuted-pixels experiment and is very close the the oracle in the permuted-labels experiment. Recall that in the oracle method we "hand-engineer" the transfer technique based on our knowledge about the problem. In contrast, our algorithm is applied identically in both experiments, since it can automatically learn the task environment by observing several tasks. Preliminary experiments with more challenging tasks (e.g., CIFAR10, Krizhevsky (2009)) and deeper networks show promising results.

As an interesting side note we observe that the stochastic models achieve slightly better results even without a prior term. This can be explained intuitively by the fact that noise injection during training leads to a solution that is more robust to perturbations, i.e. a wider minimum, which has been shown to achieve better generalization (Keskar et al., 2016; Chaudhari et al., 2016).

Table 1: Comparing the test error of different learning methods on the test tasks (average $\pm$ STD)

Method	Permuted-labels	Permuted-pixels
Scratch-standard	$2.78\% \pm 0.17\%$	$7.53\% \pm 0.46\%$
Scratch-stochastic	$2.59\% \pm 0.17\%$	$7.17\% \pm 0.46\%$
Warm-start-transfer	$1.88\% \pm 0.18\%$	$5.8\% \pm 0.99\%$
Oracle-transfer	$0.81\% \pm 0.06\%$	$2.12\% \pm 0.1\%$
LAP algorithm	$0.9\% \pm 0.08\%$	$1.26\% \pm 0.08\%$



(a) Permuted labels experiment (b) Permuted pixels experiment

Figure 2: Log weight uncertainty ($\log\left(\sigma^2\right)$) in each layer of the learned prior (average $\pm$ STD). Higher value means higher variance/uncertainty.

Analysis of learned prior Qualitative examination of the learned prior (Figure 2) affirms that it has indeed adjusted to each task environment. In the permuted-labels experiment the learned prior assigns low variance to the lower layers (fixed representation) and high variance to the output layer (which enable easy adjustment to different label permutations). As expected, in the permuted-pixels experiment the opposite phenomenon occurs. The mapping from the final hidden layer to the output becomes fixed, and the mapping from the input to the final hidden layer (representation) has more flexibility to change in light of the task data.

6 Discussion and Future Work

We have presented a framework for representational lifelong learning, motivated by PAC-Bayes generalization bounds, and implemented through the adjustment of a learned prior, based on tasks encountered so far. The framework bears conceptual similarity to the empirical Bayes method while not being Bayesian, and is implemented at the level of tasks rather than data. Combing the general approach with the rich representational structure of deep neural networks, and learning through gradient based methods leads to an efficient procedure for lifelong learning, as motivated theoretically and demonstrated empirically. There are several open issues to consider. First, the current version learns to solve all available tasks in parallel, while a truly sequential procedure should be sequential in nature. This can be easily incorporated into our framework by updating the prior following each novel task. Second, we have presented a form of the algorithm consisting of a degenerate hyper-posterior. We plan to apply the full power of the approach on a non-degenerate hyper-posterior, using, for example, the methods developed in (Hernández-Lobato & Adams, 2015). Third, we have presented preliminary suggestive experimental comparisons, and there is clearly room for a systematic comparison to standard bench-marks and algorithms. Fourth, there is much current effort in reinforcement learning to augment model free learning with model based components, where the latter are often formulated as supervised learning tasks. Incorporating our approach in such a context would be a worthwhile challenge. In fact, a similar framework to ours was recently proposed within an RL setting (Teh et al., 2017), although it was not motivated from performance guarantees as was our approach, but rather from intuitive heuristic arguments.

References

Pierre Alquier, Massimiliano Pontil, et al. Regret bounds for lifelong learning. In *Artificial Intelligence and Statistics*, pp. 261–269, 2017.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Jean-Yves Audibert. *PAC-Bayesian aggregation and multi-armed bandits*. PhD thesis, Université Paris-Est, 2010.

Jonathan Baxter. A model of inductive bias learning. *J. Artif. Intell. Res.(JAIR)*, 12(149-198):3, 2000.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pp. 1613–1622, 2015.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, and Yann LeCun. Entropy-sgd: Biasing gradient descent into wide valleys. *In ICLR 2017, arXiv preprint arXiv:1611.01838*, 2016.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *ICLR 2016, arXiv preprint arXiv:1511.07289*, 2015.

L Devroye, L Gyoörfi, and G Lugosi. *A Probabilistic Thalamiceory of Pattern Recognition*. Springer, 1996.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *In UAI 2016, arXiv preprint arXiv:1703.11008*, 2017.

Harrison Edwards and Amos Storkey. Towards a neural statistician. *In ICLR 2017, arXiv preprint arXiv:1606.02185*, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *In ICML 2017, arXiv preprint arXiv:1703.03400*, 2017.

Pascal Germain, Francis Bach, Alexandre Lacoste, and Simon Lacoste-Julien. Pac-bayesian theory meets bayesian inference. In *Advances In Neural Information Processing Systems*, pp. 1876–1884, 2016.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pp. 2348–2356, 2011.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *In ICLR 2015, arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *In ICLR 2014, 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.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, pp. 201611835, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Guy Lever, François Laviolette, and John Shawe-Taylor. Tighter pac-bayes bounds through distribution-dependent priors. *Theoretical Computer Science*, 473:4–28, 2013.

Andreas Maurer. A note on the pac bayesian theorem. *arXiv preprint cs/0411099*, 2004.

Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17(81):1–32, 2016.

David McAllester. Simplified pac-bayesian margin bounds. *Lecture notes in computer science*, pp. 203–215, 2003.

David McAllester. A pac-bayesian tutorial with a dropout bound. *arXiv preprint arXiv:1307.2118*, 2013.

David A McAllester. Pac-bayesian model averaging. In *Proceedings of the twelfth annual conference on Computational learning theory*, pp. 164–170. ACM, 1999.

Tom M Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, Rutgers Univ. New Jersey, 1980.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017.

Anastasia Pentina and Christoph H Lampert. A pac-bayesian bound for lifelong learning. In *ICML*, pp. 991–999, 2014.

Anastasia Pentina and Christoph H Lampert. Lifelong learning with non-iid tasks. In *Advances in Neural Information Processing Systems*, pp. 1540–1548, 2015.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pp. 1278–1286, 2014.

Matthias Seeger. Pac-bayesian generalisation error bounds for gaussian process classification. *Journal of machine learning research*, 3(Oct):233–269, 2002.

Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *arXiv preprint arXiv:1707.04175*, 2017.

Ilya O Tolstikhin and Yevgeny Seldin. Pac-bayes-empirical-bernstein inequality. In *Advances in Neural Information Processing Systems*, pp. 109–117, 2013.

7 APPENDIX

7.1 SINGLE TASK PAC-BAYESIAN BOUNDS

The original PAC-Bayes theorem was formulated by McAllester (1999). Later Maurer (2004) proved the following slightly tighter result:

Theorem 2 (McAllester's single-task bound)**.** *For $m \geq 8$, Proposition 1 holds with the complexity term*

$$\Psi(S, m, Q, P, \delta) = \sqrt{\frac{1}{2m}\left(D_{KL}(Q||P) + \log\frac{2\sqrt{m}}{\delta}\right)}. \tag{13}$$

Later works (Seeger, 2002) (McAllester, 2003), (Maurer, 2004) presented a more general result:

Theorem 3 (Seeger's single-task bound, KLD form)**.** *Let $P \in \mathcal{M}$ be some prior distribution over $\mathcal{H}$. For $m \geq 8$ and any $\delta \in (0,1]$ the following inequality holds:*

$$\mathbb{P}_{S\sim D^m}\left(kl(\widehat{er}\left(Q,S\right), er\left(Q,D\right)) \leq \frac{1}{m}\left(D_{KL}(Q||P) + \log\frac{2\sqrt{m}}{\delta}\right), \forall Q \in \mathcal{M}\right) \geq 1-\delta, \quad (14)$$

where $kl(p,q)$ is the KLD between two Bernoulli distributions: $kl(p,q) = p\log\frac{p}{q} + (1-p)\log\frac{(1-p)}{(1-q)}$

To a get a more convenient bound (in the form of Proposition 1) we can use Pinsker's inequality which implies $kl(p,q) \geq \frac{1}{2}(p-q)^2$. This will recover Theorem 2. But a tighter result can be obtained by closer examination of the $kl(p,q)$ function. Seeger (2002) stated that $kl(p,q) < \varepsilon$ implies $q \leq p + \sqrt{2p\varepsilon} + 2\varepsilon$. Therefore Theorem 3 leads to the following bound:

Theorem 4 (Seeger's single-task bound)**.** *For $m \geq 8$, Proposition 1 holds with the following complexity term:*

$$\Psi(S,m,Q,P,\delta) = \sqrt{2\varepsilon\widehat{er}\left(Q,S\right)} + 2\varepsilon, \quad (15)$$

where:

$$\varepsilon(m,Q,P,\delta) = \frac{1}{m}\left(D_{KL}(Q||P) + \log\frac{2\sqrt{m}}{\delta}\right).$$

In all bounds, for asymptotically large sample size m, the complexity $\Psi()$ converges to zero. The bound of Theorem 2 converges at a rate of about $1/\sqrt{m}$, as in basic VC-like bounds. The bound of Theorem 4 converges even faster (at a rate if $\frac{1}{m}$) if the empirical error $\widehat{er}\left(Q\right)$ is negligibly small (compared to $\frac{1}{m}D_{KL}(Q||P)$). Since this is commonly the case in modern deep learning, we expect this bound to be tighter than others in this regime.

More recent works presented possibly tighter bounds by taking into account the empirical variance (Tolstikhin & Seldin, 2013) or by specializing the bound deep for neural networks (Neyshabur et al., 2017). However, we leave the incorporation of these bounds for future work.

7.2 Pentina and Lampert's lemma

Lemma 1 (From (Pentina & Lampert, 2014))**.** *Let F be a set and let $\mathcal{F}$ be the set of distributions over F. Similarly, define the sets $A_1, ..., A_K$ and corresponding sets of distributions $\mathcal{A}_1, ..., \mathcal{A}_K$.*

Let f be a random variable taking values in F and let $X_1, ..., X_K$ be K independent random variables, with each X_k distributed according to the distributions $\mu_k \in \mathcal{A}_k$. Let $g_k : F \times A_k \to [a_k, b_k], k = 1, ..., K$ be a sequence of measurable functions. Define $\xi_k(f) = \mathop{\mathbb{E}}_{X\sim\mu_k} g_k(f,X)$, $\forall f \in F$.

Then for any fixed distribution $\pi \in \mathcal{F}$, any $\lambda > 0$ and any $\delta \in (0,1]$ the following inequality holds:

$$\mathbb{P}_{X_1,...,X_K}\left\{\mathop{\mathbb{E}}_{f\sim\rho}\sum_{k=1}^{K}\xi_k(f) \leq \mathop{\mathbb{E}}_{f\sim\rho}\sum_{k=1}^{K} g_k(f,X_k) + \right. \quad (16)$$
$$\left.\frac{1}{\lambda}\left(D_{KL}(\rho||\pi) + \frac{\lambda^2}{8}\sum_{k=1}^{K}(b_k-a_k)^2 + \log\frac{1}{\delta}\right), \forall \rho \in \mathcal{F}\right\} \geq 1-\delta$$

For proof see (Pentina & Lampert, 2014).

7.3 Proof of the lifelong-learning PAC-Bayes bound

The proof is based on two steps. First we bound the transfer error $er\left(\mathcal{Q}\right)$ using the average generalization error in each of the observed tasks. In the second step we will use the single-task PAC-Bayes bound for each observed task to get the final bound. [12]

[12] In the proof of the main theorem of (Pentina & Lampert, 2014) the same first step is used, but the second step is different.

Claim 1. *Let $\mathcal{P}$ be some distribution over $\mathcal{M}$ and let $\delta_0 \in (0,1)$, then the following inequality holds:*

$$\mathbb{P}_{D_i\sim\tau, S_i\sim D_i^{m_i}, i=1,...,n}\Bigg\{ er\left(\mathcal{Q},\tau\right) \leq \frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{P\sim\mathcal{Q}} er_i\left(Q(S_i,P),D_i\right) + \frac{1}{\sqrt{n}}\left(D_{KL}(\mathcal{Q}||\mathcal{P}) + \log\frac{1}{\delta_0} + \frac{1}{8}\right), \forall \mathcal{Q}, Q_1, ..., Q_n \Bigg\} \geq 1-\delta_0, \tag{17}$$

where $er_i\left(Q(S_i,P),D_i\right)$ is the generalization error in task i.

Proof. Apply Lemma 1 with the following substitutions: $K=n$, $\lambda=\sqrt{n}$, $\rho=\mathcal{Q}$, $\pi=\mathcal{P}$, $X_k=(D_k,S_k)$, $\mu_k=(\tau,D_k^{m_k})$ and $f=P$, where D_k denotes the sample distribution associated with the k-th observed task.

Also define

$$g_k(f,X_k) \triangleq \mathop{\mathbb{E}}_{h\sim Q(S_k,P)} \mathop{\mathbb{E}}_{z\sim D_k} \ell(z,h) = er_k\left(Q(S_k,P),D_k\right).$$

Consequently, we have

$$\xi_k(f) = \mathop{\mathbb{E}}_{X_k\sim\mu_k} g_k(f,X_k) = \frac{1}{n}\mathop{\mathbb{E}}_{t\sim\tau} \mathop{\mathbb{E}}_{S\sim D_t^{m_t}} \mathop{\mathbb{E}}_{h\sim Q(S,P)} \mathop{\mathbb{E}}_{z\sim D_t} \ell(z,h) = \frac{1}{n} er\left(P,\tau\right)$$

Since the loss $\ell(h,z)$ is bounded in $[0,1]$ then $g_k(f,X_k)$ is bounded in $[0,\frac{1}{n}]$. Plugging these results in Lemma 1 leads directly to the claim. □

Now we wish to bound the generalization error $er_i\left(Q(S_i,P),D_i\right)$ in each observed task individually.

Let $\Psi()$ be some form of single-task complexity term that satisfies the generic Proposition 1 and let $\delta_i \in (0,1], i=1,...n$. For a fixed $P\in\mathcal{M}$ we have a set of n inequalities for $i=1,...,n$:

$$\mathbb{P}_{S_i\sim D_i^m}\left(er_i\left(Q',D_i\right) \leq \widehat{er}_i\left(Q',S_i\right) + \Psi(S_i,m_i,Q',P,\delta_i), \forall Q'\in\mathcal{M}\right) \geq 1-\delta_i.$$

Since the above inequality holds uniformly for all $Q'\in\mathcal{M}$, then it also holds if we plug in $Q(S_i,P)$:

$$\mathbb{P}_{S_i\sim D_i^m}\left(er_i\left(Q(S_i,P),D_i\right) \leq \widehat{er}_i\left(Q,S_i\right) + \Psi(S_i,m_i,Q(S_i,P),P,\delta_i)\right) \geq 1-\delta_i \tag{18}$$

We want the intersection of the events of 17 and 18 to all hold with probability greater than $1-\delta$. We can use the union bound on all events, but for a tighter bound we can use the independence of the tasks. Using the $i.i.d$ tasks assumption, the intersection of the n events of 18 occurs with probability of at least $\prod_{i=1}^{n}(1-\delta_i)$.

Now, by the union bound, all events of 17 and 18 occur with probability of at least $1-\delta_0-(1-\prod_{i=1}^{n}(1-\delta_i))$. Therefore we require $\delta = 1+\delta_0-\prod_{i=1}^{n}(1-\delta_i)$.

Finally, we conclude that the inequality of the lifelong-learning bound indeed holds with probability of at least $1-\delta$.

7.4 Derivation of the single prior form

In this section we provide a detailed explanation of the derivation of the single prior form of the objective (mentioned in section 4.1).

We cannot directly set the PDF of $\mathcal{Q}$ to be a delta function at a single prior (since then the KLD term, $D_{KL}(\mathcal{Q}||\mathcal{P})$, would diverge). Instead, we will use a ball in parameter space and inspect the limit of infinitesimal volume. Given a parametric family of priors $\left\{P_\theta : \theta\in\mathbb{R}^{N_P}\right\}$, the space of hyper-posteriors consists of all distributions over $\mathbb{R}^{N_P}$. We will limit our search to a certain family of hyper-posteriors. Define the function $f_\varepsilon(\theta):\mathbb{R}^{N_P}\to\mathbb{R}$ with a value $1/\varepsilon$ inside a ball of volume $\varepsilon>0$ around the origin and zero otherwise. For any $\theta_P\in\mathbb{R}^{N_P}$ the PDF of the hyper-posterior associated with θ_P is defined by $\mathcal{Q}_\varepsilon(\theta;\theta_P) \triangleq f(\theta-\theta_P)$.

By inserting the hyper-posterior $\mathcal{Q}_\varepsilon(\theta;\theta_P)$ into the lifelong objective (eq. 8) we get:

$$J(\mathcal{Q}_\varepsilon(\theta;\theta_P)) = \frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\theta\sim\mathcal{Q}_\varepsilon(\theta;\theta_P)} J_i(\theta) +$$

$$\frac{1}{\sqrt{n}} D_{KL}(\mathcal{Q}_\varepsilon(\theta;\theta_P)||\mathcal{P}) \mathop{\mathbb{E}}_{\theta\sim\mathcal{Q}_\varepsilon(\theta;\theta_P)} \left\{ \frac{1}{n}\sum_{i=1}^{n} J_i(\theta) + \frac{1}{\sqrt{n}} \log \frac{\mathcal{Q}_\varepsilon(\theta;\theta_P)}{\mathcal{P}(\theta)} \right\}.$$

For small enough values of ε this equation can be approximated as

$$J(\mathcal{Q}_\varepsilon(\theta;\theta_P)) \cong \frac{1}{n}\sum_{i=1}^{n} J_i(\theta_P) + \frac{1}{\sqrt{n}} \log \frac{1/\varepsilon}{\mathcal{P}(\theta_P)} =$$

$$= \frac{1}{n}\sum_{i=1}^{n} J_i(\theta_P) - \frac{1}{\sqrt{n}} \log \mathcal{P}(\theta_P) + const.,$$

where $const.$ denote terms not dependent on θ_P[13].

7.5 PSEUDO CODE

Algorithm 1: LAP algorithm, meta-training phase (learning-to-learn)

Input : Data sets of observed tasks: $S_1, ..., S_n$
Output: Learned prior parameters θ_P

Initialize:

- $\theta_P = (\mu_P, \rho_P) \in \mathbb{R}^d \times \mathbb{R}^d$
- $\phi_i = (\mu_i, \rho_i) \in \mathbb{R}^d \times \mathbb{R}^d, \quad \text{for} \quad i = 1, ..., n$

while *not done* **do**
  for *each task* $i \in \{1, ..n\}$ [14] **do**
    • Sample a random mini-batch from the data $S_i' \subset S_i$
    • Approximate the empirical loss $\widehat{er}_i$ using S_i' and averaging Monte-Carlo draws
    • Calculate the complexity term Ψ_i
  end
  • $J \leftarrow \frac{1}{n}\sum_{i\in\{1,..n\}}(\widehat{er}_i + \Psi_i) + \frac{1}{\sqrt{n}} \log 1/\mathcal{P}(\theta_P)$
  • Evaluate the gradient of J w.r.t $\{\theta_P, \phi_i\}$ using backprop
  • Take an optimization step
end

Algorithm 2: LAP algorithm, meta-testing phase (learning a new task)

Input : Data set of a new task, S, and prior parameters, θ_P
Output: Posterior parameters ϕ' which solve the new task

Initialize:

- $\phi' \leftarrow \theta_P$

while *not done* **do**
  • Sample a random mini-batch from the data $S' \subset S$
  • Approximate the empirical loss $\widehat{er}$ using S' and averaging Monte-Carlo draws
  • Calculate the complexity term Ψ
  • $J \leftarrow \widehat{er} + \Psi$
  • Evaluate the gradient of J w.r.t ϕ' using backprop
  • Take an optimization step
end

[13] While the constant term increases as ε decreases, it does not affect the terms that are optimized over.

[14] For implementation considerations, when training with a large number of tasks we can sample a subset of tasks in each iteration ("meta min-batch") to estimate J.

7.6 CLASSIFICATION EXAMPLE IMPLEMENTATION DETAILS

The network architecture used for the permuted-labels experiment is a small CNN with 2 conv-layers of 10 and 20 filters, each with 5×5 kernels, a hidden linear layer with 50 units and a linear output layer. In the the permuted-pixels experiment we used a fully-connected network with 3 hidden layers of 400 units and a linear output layer. In both networks we use ELU (Clevert et al., 2015) as an activation function. We used Seeger's complexity term (Theorem 4). Both phases of the LAP algorithm (algorithms 1 and 2) ran for 300 epochs, with batches of 128 samples in each task. We average 3 Monte-Carlo samples of the stochastic network output in each step. As optimizer we used ADAM (Kingma & Ba, 2014) with learning rate of 10^{-3}. The means of the weights (μ parameters) are initialized randomly by $\mathcal{N}\left(0, 0.1^2\right)$, while the log-var of the weights (ρ parameters) are initialized by $\mathcal{N}\left(-10, 0.1^2\right)$. For Algorithm 1 (meta-training), the hyper-prior, $\log 1/\mathcal{P}(\theta_P)$, is set as the L_1 norm of the weights times $\kappa = 10^{-7}$.

To evaluate the trained network we used the maximum of the posterior for inference (i.e. we use only the means the weights). [15]

[15] Classifying using the the majority vote of several runs gave similar results in this experiment.