Multi-Task Learning: Unlocking Potential with New Python Library

Kirill Semlin Iryna Zabarianska Ilgam Latypov
Alexander Terentyev

Multi-task learning (MTL) has emerged as a powerful paradigm in machine learning, aiming to leverage the relationships among related tasks to enhance the performance of individual tasks. This approach is particularly beneficial in scenarios where data is scarce for each task, making it challenging to develop robust models independently. By recognizing that many real-world problems can be framed as a series of similar yet distinct tasks, MTL provides a framework for sharing knowledge across these tasks, ultimately leading to improved outcomes.

The central objective of MTL is to explore and exploit task relatedness to bolster the performance of each individual task. However, uncovering these relationships can be complex and highly nonlinear, presenting significant challenges.

This article presents an overview of various models for MTL and announces a new Python library, **torch.<name of lib>**, featuring their implementation.

1 Preliminaries

General designations

2 Task Clustering and Gating for MTL

Introduction and preliminaries. More details in the article [1].

Clustering of Tasks

Suppose we have several clusters of similar tasks. Then we could take as a prior a mixture of $n_{cluster}$ Gaussians

$$A_i \sim \sum_{\alpha=1}^{n_{cluster}} q_\alpha \mathcal{N}(m_\alpha, \Sigma_\alpha),$$

where q_α represents the *a priori* probability for any task to be 'assigned to' cluster α. Then the posterior data likelihood expressed as

$$P(D|\Lambda) = \int dA_i P(D_i|A_i, \Lambda) \sum_{\alpha=1}^{n_{cluster}} q_\alpha P(A_i|m_\alpha, \Sigma_\alpha).$$

In this way, the posterior distribution effectively 'assigns' tasks to that cluster that is most compatible with the data within the task, in the sense that all other clusters (Gaussians) do contribute much less.

Gating of Tasks

In task clustering approach the prior is task-independent: all tasks are assigned to each of the clusters with the same probabilities q_α. A natural extension is to incorporate the task-dependent features f_i in a gating model, for example

$$q_{i\alpha} = e^{U_\alpha^T f_i} / \sum_{\alpha'} e^{U_{\alpha'}^T f_i}.$$

The above task clustering approach is a special case with $n_{feature} = 1$ and $f_i = 1$ for all tasks i.

3 MTL with Latent Hierarchiess

Introduction and preliminaries. More details in the article [2].

Domain Adaptation

A tree structure will be generated according to a K-coalescent process, and weight vectors will be propagated along this tree. The root of the tree represents the "global" weight vector, while the leaves correspond to the weight vectors specific to each task. It is assumed that the weight vectors evolve according to Brownian diffusion. algo

Multi-Task Learning

The algorithm for MTL will be a modification of the algorithm used for DA. In the case of MTL, weight vectors will not be shared; instead, their covariance structure will be shared. This model is somewhat more complex to specify because Brownian motion is not applicable to a covariance structure (for instance, it does not preserve positive semi-definiteness). To address this issue, the covariance structure will be decomposed into correlations and standard deviations, assuming a constant global correlation matrix while allowing the standard deviations to vary across the tree.

4 Sparse Bayesian MTL

Introduction and preliminaries. More details in the article [3].

A general family of group sparsity inducing priors

We assume that the effective prior on each block $W_i \in \mathbb{R}^{P\times D_i}$ has the form of a matrix-variate Gaussian scale mixture, extending the multivariate Gaussian scale mixture:

$$p(W_i) = \int_0^\infty \mathcal{N}(0, \gamma_i^{-1}\Omega_i, \Sigma)p(\gamma_i)d\gamma_i, \quad \sum_{i=1}^{Q} D_i = D,$$

where $\Omega_i \in \mathcal{R}^{D_i\times D_i}$, $\Sigma \in \mathcal{R}^{P\times P}$ and $\gamma_i > 0$ is the latent precision for block W_i.

To induce sparsity in W_i, we select an appropriate hyperprior for γ_i. We impose a generalized inverse Gaussian prior for the latent precision variables:

$$\gamma_i \sim \mathcal{N}^{-1}(\omega, \chi, \phi) = \frac{\chi^{-\omega}(\sqrt{\chi\phi})^\omega}{2K_\omega\sqrt{\chi\phi}}\gamma_i^{\omega-1}e^{-\frac{1}{2}(\chi\gamma_i^{-1}+\phi\gamma_i)}$$

where $K_\omega(.)$ is the modified Bessel function of the second kind, ω is the index, $\sqrt{\chi\phi}$ defines the concentration of the distribution and $\sqrt{\chi/\phi}$ defines its scale. The effective prior is then a symmetric matrix-variate generalised hyperbolic distribution

$$p(W_i) \propto \frac{K_{\omega+\frac{PD_i}{2}}\left(\sqrt{\chi(\phi + tr\{\Omega_i^{-1}W_i^T\Sigma^{-1}W_i\}}\right)}{\left(\sqrt{\frac{\chi(\phi+tr\{\Omega_i^{-1}W_i^T\Sigma^{-1}W_i\}}{\chi}}\right)^{\omega+\frac{PD_i}{2}}}.$$

This is crucial for further construction of sparse models and variational inference.

5 Variational MTL with Gumbel-Softmax Priors

Introduction and preliminaries. More details in the article [4].

Learning Task Relatedness via Gumbel-Softmax Priors

To leverage the shared knowledge, we propose to specify the prior of the classifier for the current task using the variational posteriors over classifiers of other tasks:

$$p(w_t^{(\eta)}|D_{1:T\setminus t}) = \sum_{i\neq t}\alpha_{ti}q_\theta(w_i^{(\eta-1)}|D_i),$$

where η denotes the iteration number.

During training, our goal is for each task to learn relevant shared knowledge from closely related tasks while minimizing interference from irrelevant ones. To achieve this, we employ the Gumbel-Softmax technique to learn mixing weights, defined as:

$$\alpha_{ti} = \frac{exp((log\pi_{ti} + g_{ti})/\tau}{\sum_{i\neq t} exp((log\pi_{ti} + g_{ti})/\tau}.$$

Here α_{ti} is the mixing weight that indicates the relatedness between tasks t and i. The parameter π_{ti} is a learnable component indicating the likelihood of knowledge transfer between tasks, while g_{ti} is sampled from a Gumbel distribution, using inverse transform sampling. The temperature parameter τ controls the softmax behavior. This approach helps manage potential negative transfer by encouraging lower mixing weights for less relevant tasks, effectively reducing interference.

For the variational posteriors, we define them as fully factorized Gaussians for each class:

$$q_\theta(w_t|D_t) = \prod_{c=1}^{C} q_\theta(w_{t,c}|D_{t,c}) = \prod_{c=1}^{C} \mathcal{N}(\mu_{t,c}, diag(\sigma_{t,c}^2)).$$

6 New Python Library: torch.<name of lib>

We implement the following models for MTL in **torch.<name of lib>**:

1. ...;
2. ...;
3. ...;
4. ...;

We aim to deliver dependable, efficient, and user-friendly tools that will be advantageous for both researchers and practitioners. We anticipate that the library's implementation will foster further advancements in this scientific field. Keep an eye out for updates!

Git: https://github.com/intsystems/bmm-multitask-learning.git