From Multi-label Learning to Cross-Domain Transfer: A Model-Agnostic Approach

Jesse Read

`jesse.read@polytechnique.edu`

LIX Laboratory, Ecole Polytechnique, Institut Polytechnique de Paris, France

In multi-label learning, a particular case of multi-task learning where a single data point is associated with multiple target labels, it was widely assumed in the literature that, to obtain best accuracy, the dependence among the labels should be explicitly modeled. This premise led to a proliferation of methods offering techniques to learn and predict labels together, for example where the prediction for one label influences predictions for other labels. Even though it is now acknowledged that in many contexts a model of dependence is not required for optimal performance, such models continue to outperform independent models in some of those very contexts, suggesting alternative explanations for their performance beyond label dependence, which the literature is only recently beginning to unravel. Leveraging and extending recent discoveries, we turn the original premise of multi-label learning on its head, and approach the problem of joint-modeling specifically under the absence of any measurable dependence among task labels; for example, when task labels come from separate problem domains. We shift insights from this study towards building an approach for transfer learning that challenges the long-held assumption that transferability of tasks comes from measurements of similarity between the source and target domains or models. This allows us to design and test a method for transfer learning, which is model driven rather than purely data driven, and furthermore it is black box and model-agnostic (any base model class can be considered). We show that essentially we can create task-dependence based on source-model capacity. The results we obtain have important implications and provide clear directions for future work, both in the areas of multi-label and transfer learning.

1 Introduction

Tackling multiple tasks in a single machine learning framework is becoming commonplace, and is found in different forms, and applied to a wide variety of problems. Specific cases of multi-task learning include multi-label classification, where tasks are binary classification problems, sharing common input; and transfer learning, where only one or a subset of tasks will be evaluated and the other tasks are leveraged to obtain better performance (predictive performance, computational performance, or both) under that evaluation.

The *raison d'être* of these areas of the literature stems from the same promise: by formulating tasks together, we can obtain greater predictive performance and/or reduced computational requirements, than if we approach each task separately. To a significant degree, the promise has been met. The multi-label literature, for example, has flourished over the previous decades with a plethora of successful learning methods that out-compete independent models.

However, the success of considering multiple tasks jointly rather than separately has been less widespread in, for example, transfer learning (even if the success has been just as significant; it is nevertheless less widely employed; more restricted to particular domains than, say, multi-label learning) due to a number of major challenges invoked by one assumption: that the tasks in consideration should exhibit similarity. In multi-label learning, since tasks share a common input and a common dataset, there is usually some assumed inter-dependence among the output labels. In transfer learning, the source task does not form an integral part of the target task, rather one needs to identify a source task of adequate similarity. This increases manual and computational search time (the savings of which was a primary motivator for considering transfer learning in the first place).

Purely data-driven approaches can be costly, and for than reason model-based transfer-learning is an interesting alternative. Recent developments have almost invariably taken the form of novel deep

neural architectures, and have met great success in domains such as natural language and images, where data is large, and deep learning models are notoriously successful yet also too expensive to train from scratch on all problems due to the enormous quantities of data needed to be processed. Therefore the current common practice is essentially importing and (optionally) fine-tuning hidden layers from other neural networks and this practice is for a restricted small set of domains.

However, very many machine learning engineers and data scientists around the world use algorithms that do not fall into the category of 'deep learning' (Grisel [2021] estimates 600,000 monthly scikit-learn users – a framework which contains only rudimentary neural network learning algorithms). Yet, many machine learning models cannot be easily dissected into hidden layers expressed as weight matrices as in popular deep-learning frameworks, and even when such an expression is trivial, sharing such layers is not yet widely practiced (at least; not beyond the image and natural-language domains).

In this article we approach the problem of model-based model-agnostic transfer learning (i.e., model-driven/no source data available, and no possibility to inspect the models). The key component of our approach is to remove the assumption of task similarity, i.e., the source and target tasks may come from *unrelated* domains. More precisely: we do not require any more similarity between target and source domains, than between any randomly selected domains in the scope of any task having somewhere some useful application.

We do this by building from aspects and observations that have arisen from the multi-label literature. Notably, in this domain, model-agnostic approaches are commonplace (where they are often known as *problem transformation* or data transformation methods); implicating a wide diversity of model classes (decision trees, support vector machines, logistic regression, etc.). These models are often adapted to produce predictions for multiple tasks simultaneously, or separate instantiations of such models are linked together across the tasks via their predictions in a modular fashion within a 'meta' method. Such methods are widely reported to outperform independent models. Yet, interestingly, one can observe that these empirical studies indicate such outperformance (of interconnected dependence-based models vis-a-vis independent models) even when label dependence is not explicitly required to minimize a loss metric. This allows us to formulate a new hypothesis: transfer learning can still be advantageous even when target tasks come from unrelated application domains.

This is an extremely challenging context. Because the model representation is out of reach we can only access inputs and outputs and thus face an acute information bottleneck. We overcome this challenge by generalizing to multi-label transfer learning, where source tasks are multi-label tasks, producing an output vector of predictions, of sufficient size and richness to convey some additional capacity to the target model. Since the source data is unavailable, to obtain these predictions we must feed the source model with *target* inputs. This embodies the main challenge: source and target domains are unrelated to each other, so we must use the source task as an analogy to the target task; creating a kind of artificial dependence between the tasks and making use of similarity that occurs by chance.

In other words this paper looks to translate some important results from multi-label learning into the particular case of model-agnostic cross-domain transfer learning; and then study and develop them further in this context. The main objective is not to forward a number of empirical outcomes on benchmark data sets supporting a claim to a new state of the art method, but rather to study and demonstrate underlying mechanisms, and discuss their implications. We develop a deeper understanding on how task similarity/correlation is involved in multi-task learning, from the perspective of multi-label classification. Alongside empirical testing, we discuss the implications on different types of transfer learning, including, for example, adaptation to concept shift in data-streams.

The article is outlined as follows: In Section 2 we provide notational preliminaries and formally define the problem setting. In Section 3 we review transfer learning and discuss particular methodologies of interest, looking also at related areas relevant to our study such as multi-task and multi-label learning, and concept drift adaptation. In Section 4 we draw lessons from the existing (and relatively well-developed) area of multi-label learning, and use them to formulate a number of hypotheses towards our goal of model-agnostic cross-domain transfer learning. In Section 5 we take these lessons and apply them to develop a novel approach to transfer learning under the numerous constraints and motivations outlined above; named Transfer Chains. Both sections Section 4 and Section 5 are supplemented by a number of empirical evaluations within. We pool together outcomes and discuss at length in Section 6, leading to number of observations and recommendations that have important implications in the general area of transfer learning. We conclude in Section 7, outlining some promising future directions.

2 Notation and Problem Setting

A task is defined as seeking to build model

$$h : \mathcal{X} \to \mathcal{Y}$$

mapping instances $\boldsymbol{x}$ from an input domain $\mathcal{X} = \mathbb{R}^d$ to labels $\boldsymbol{y}$ belonging to the output domain $\mathcal{Y} = \mathbb{R}^m$, under the goal of minimizing some loss metric $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. In the multi-label case, $m > 1$ (typically, in the multi-label classification setting, $\mathcal{Y} = \{0, 1\}^m$). In the multi-task setting, we can explicitly denote a number of tasks $1, 2, \ldots$, i.e., $\mathcal{X}_1, \mathcal{X}_2, \ldots$ and $\mathcal{Y}_1, \mathcal{Y}_2, \ldots$. Note that there is also ambiguity here: the multi-label setting could be defined as a kind of multi-task setting having a single $\mathcal{X}$ and multiple task labels $\mathcal{Y}_1, \mathcal{Y}_2, \ldots$; one for each task/label. And, even, $\mathcal{Y}_1 = \mathcal{Y}_{1,1} \times \ldots \times \mathcal{Y}_{1,m_1}$ (multi-dimensional labels). We embrace this ambiguity throughout, in order to bridge between multi-label and multi-task settings (which are otherwise often treated separately in the literature); the distinction is only in the input: a multi-label task has a single domain $\mathcal{X}$. Transfer learning is the specific multi-task task where a number of tasks are source models and the others are target tasks. For notational simplicity we denote a single source task (S) and single target task (T), but without loss of generality; indeed in experiments we consider several source tasks.

We approach the task of model-based model-agnostic transfer, implying the construction of target model h_T, leveraging a source model h_S, under the unique combination of constraints that we:

1. do not have access to the source data set(s),

2. cannot inspect the source model(s); and

3. the source domain is not related to the target domain.

Note that this is different from, and much more difficult than, the huge majority of transfer learning approaches which seek to find and inspect source tasks, measure their similarity to the target task, then transfer relevant parts of either into the source model, prior to (possible) fine tuning.

We are supplied with

1. A source model $h_S : \mathcal{X}_S \to \mathcal{Y}_S$ (or, generally, models) already built to tackle some source task

2. A target data set $\mathcal{D}_T$

and the goal is to build a model for the target task $h_T : \mathcal{X}_T \to \mathcal{Y}_T$.

Despite the unique combination of constraints our general objective is to nevertheless the same; to make use of source models in the form of transfer learning to improve accuracy on the target task.

There is no strict requirement to use a source model h_S at all when constructing the target model h_T, yet we are motivated to do so by the potential to improve learning (as opposed to a purely data-driven approach to the target task), including Torrey and Shavlik [2010] that predictive performance:

1. is initially higher (after initialization)

2. increases more quickly (during training); and

3. plateaus at higher level (when training is finished).

We investigate if such an advantage can hold after removing the common assumption that source task and target task are in some way related. This is the far less-common *cross-domain* transfer learning. We assume this setting to the extent that we do not expect any more similarity between the target task and source task than would be present between any two tasks drawn uniformly at random from all possible real-world datasets. These strict assumptions make it infeasible to compete with modern state of the art transfer-learning methods (which explicitly assume and aim to leverage similarity). But they raise interesting considerations of if/when, and how, it can be beneficial to carry out transfer learning in this setting. We investigate these considerations.

3 Related Work

The task we approach is essentially a heavily constrained type of transfer learning. In this section we look at links to existing work on transfer learning in the literature as relevant to this setting. Table 1 places model-based transfer learning in the context of related tasks. Recall, we are specifically interested in the case where knowledge is stored in the form of existing models (rather than data sets). Data-driven approaches are enormously important and have advanced the applicability of machine learning in many areas, but do not coincide with our particular constraints. Meanwhile, Figure 1 exemplifies the differences (and similarities) among different models.

Table 1: Transfer learning and related methodologies; denoted with respect to two potential tasks: 1 and 2, when the tasks may be learned concurrently and are both of interest, and tasks S (source) and T (target) when the target task is the main focus. In self-taught learning, only data from input-domain of the source task is available. In transfer learning, input data and/or source models can be available; in our case we consider only source models and no data.

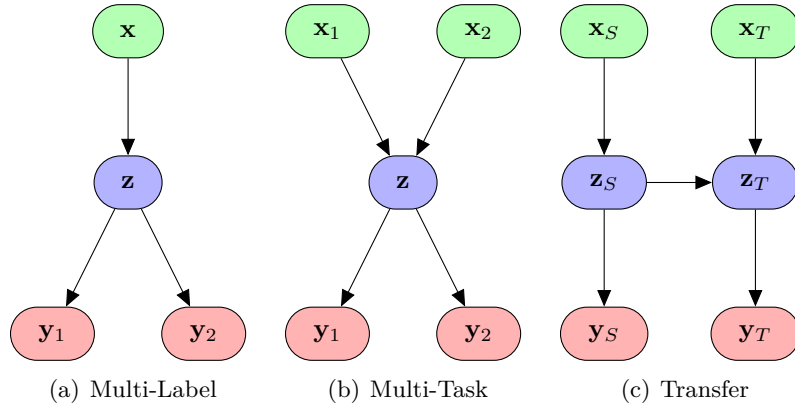| Name	Target Task	Source Task	Comment
[Semi-]Supervised learning	$\mathcal{X} \mapsto \mathcal{Y}$		Single task
Multi-label learning	$\mathcal{X} \mapsto \{\mathcal{Y}_1, \mathcal{Y}_2\}$		Several tasks, $\mathcal{X}$ in common
Multi-task learning	$\{\mathcal{X}_1, \mathcal{X}_2\} \mapsto \{\mathcal{Y}_1, \mathcal{Y}_2\}$		Several tasks
Transfer learning	$\mathcal{X}_T \mapsto \mathcal{Y}_T$	$\mathcal{X}_S \mapsto \mathcal{Y}_S$	Source and target tasks
Concept-drift (input-only)	$\mathcal{X}_S \to \mathcal{Y}_T$	$\mathcal{X}_S \mapsto \mathcal{Y}_S$	$\mathcal{X}$ in common, models available
Concept-drift (output-only)	$\mathcal{X}_T \to \mathcal{Y}_S$	$\mathcal{X}_S \mapsto \mathcal{Y}_S$	$\mathcal{Y}$ in common, models available
Self-taught learning	$\mathcal{X}_T \mapsto \mathcal{Y}_T$	$\mathcal{X}_S \mapsto \mathcal{Y}_S$	Source task unknown

(a) Multi-Label   (b) Multi-Task   (c) Transfer

Figure 1: Simplified architectures (with hidden representation, $\boldsymbol{z}$) for (a) multi-label learning, (b) multi-task learning, and (c) transfer learning. In (c) we see two intermediate $\boldsymbol{z}$-layers, but only one is used per task. In fact, the multi-label task is a generalization, having multi-dimensional labels; $\boldsymbol{y}_1 \in \mathbb{R}^{m_2}$; $\boldsymbol{y}_2 \in \mathbb{R}^{m_2}$).

3.1 Types of transfer learning and methods

The canonical task of transfer learning in machine learning, from which many diverse problem settings arise, can be phrased as transferring knowledge from a *source task* to a *target task*. The transfer can be in the form of data (data-based transfer), a model representation (model-based transfer), or even hyper-parameters (often called in this form meta learning). A general survey and introduction is given by Pan and Yang [2010], Torrey and Shavlik [2010].

Deep transfer learning (DTL) (a survey by Tan et al. [2018]) is by far the most common form of transfer today, owing to the enormous current popularity and renown effectiveness of deep learning. It is a straightforward, well-documented, and largely successful recipe for transfer learning, usually cast as follows (and depicted in Figure 1(c)):

1. Identify a suitable deep neural network source model $h_S : \mathcal{X}_S \to \mathcal{Y}_S$; where source domain $\mathcal{X}_S \times \mathcal{Y}_S$ is related in some way to the target domain $\mathcal{X}_T \times \mathcal{Y}_T$;

2. transplant one or several hidden layers (depicted as $\boldsymbol{z}_S \mapsto \boldsymbol{z}_T$ in Figure 1(c)) from the source architecture into the target architecture; and

3. train ($\boldsymbol{z}_T \mapsto \boldsymbol{y}_T$) and – optionally – fine tune the full target network ($\boldsymbol{x}_T \mapsto \boldsymbol{z}_T \mapsto \boldsymbol{y}_T$).

We can cite the image vectors of Simonyan and Zisserman [2015] and word-embeddings of Mikolov et al. [2013] as approaches having an enormous impact on the community, especially in the areas of computer vision and natural language processing where the representations of these models (and others of similar nature) are often incorporated into new models as a standard, or even necessary component for producing state of the art models. Recent work incorporating deep-learning frameworks such as attention mechanisms [Wang et al., 2019] has further improved this area.

In the probabilistic sense, transfer learning can be carried out by using the posterior distribution of

the source model as a prior for another. By Bayes' theorem[1],

$$P(\boldsymbol{Y}_T|\boldsymbol{x}_T) = \frac{P(\boldsymbol{x}_T|\boldsymbol{Y}_T)P(\boldsymbol{Y}_T)}{P(\boldsymbol{x}_T)} \propto P(\boldsymbol{x}_T|\boldsymbol{Y}_T)P(\boldsymbol{Y}_T)$$

where we then use $P(\boldsymbol{Y}_T) \approx P(\boldsymbol{Y}_S|\boldsymbol{x}_T)$ from the posterior of the source task. And more specifically, in Bayesian methods we may place a prior over model parameters [Karbalayghareh et al., 2018, Chandra and Kapoor, 2020].

Concept drift and – particularly in this context – the *adaptation to* concept drift [Gama et al., 2014, Zhao et al., 2020] is a central concept to a large volume of literature on learning from data streams. Even if not typically discussed explicitly by authors, adaptation to drift is indeed a form of transfer learning. Namely, the target task (i.e., target *concept*) is a degradation of the source task wrt that task. This form of transfer learning is interesting to us, because sudden and complete concept shift (considered frequently in the literature [Gama et al., 2014]), meaning that source and target domains are unrelated, is equivalent to the scenario we are aiming to tackle. The majority of the data-stream literature suggests the best course of action is to scrap completely the earlier (source) models. Our results specifically challenge this view (discussion later in Section 6.2).

This area of the literature is also interesting to our objectives because methods for drift-detection and adaptation are often treated as black-box and model-agnostic; typically a dynamic ensemble of models is maintained, without the need to explicitly specify the base model class Gomes et al. [2017] and members are reset (simply purged and reinitialized) over time. Maintaining models in a data stream means that, almost by definition (since streams are potentially infinite, whereas resources are usually not), source data cannot be kept for the source task; thus driving model-driven adaptation.

The assumption of that source data is not available has also been tackled in, e.g., the context of natural language processing by [Chen et al., 2019], an approach both multi-source and model-based (i.e., not requiring parallel data). And Davis and Domingos [2009] consider the case of different distribution and different domain, finding structural similarities in the form of Markov logic. Meanwhile, Hsu et al. [2018] perform cross-domain adaptation by focusing not only on a transfer of features, but rather of *metrics* of similarity. The importance of task diversity (as opposed to the importance of task similarity which is usually of focus) is approached by Tripuraneni et al. [2020], but they still rely on the common assumption that the main purpose is to learn a feature representation shared across different tasks (i.e., multi-task, rather than transfer learning).

Meta-learning is a kind of transfer learning that transfers from the *meta* source task in terms of hyper-parametrization, including particular architecture search. In other words, knowledge from the source domain itself is not directly transferred, but rather, knowledge of how to approach tasks. There are a number of modern examples (e.g., [Alet et al., 2018, Finn et al., 2017, Andreas et al., 2016, Chen et al., 2020, Zoph and Le, 2017]). However, despite often carrying the 'model-agnostic' label, these methods are still gradient-descent/neural-network centric and almost invariably assume access to source datasets (i.e., they are predominantly data-driven, rather than model-driven).

3.2 Modular learning and random projection methods

The approach we take, with no assumptions on task similarity, brings us close to that of modular random projections.

Random projections and untrained layers have been used in machine learning for decades, including textbook examples of polynomial regression [Hastie et al., 2001], random-basis function neural networks [Du and Swamy, 2013], the eccentrically-named 'extreme learning machines' [Huang et al., 2011] and random kitchen sinks [Rahimi et al., 2007], and – if we involve recurrent layers – reservoir computing [Lukoševičius and Jaeger, 2009]. Such techniques have been frequently criticized or, worse, largely ignored by the deep-learning community where gradient-descent and back-propagation are the dominant techniques yet, nevertheless, these other diverse methods continue to persist and are successful in many tasks.

In our work the source model is not a random projection in the traditional sense that the model parameters are drawn randomly, rather in the sense of the model being trained on a source task which was drawn randomly (rather than guided by a similarity metric). Nevertheless, we face potentially the same challenge of information bottlenecks: even if a large amount of data is moved easily in memory the amount of information (relevant to solving the target task) may be poor.

---

[1]Throughout, we use the notation $P(Y|x)$ as shorthand for $P(Y|X = x)$; the conditional distribution – conditioned on observation x; a realization of the random variable X

The model-agnostic setting of not being able to inspect source models or view the data they were trained on (the context we consider) lends itself naturally to a comparison to modular learning; building a web of structure across black-box models. Such methods are well known in the multi-label literature (e.g., by Gasse [2017], Read et al. [2021], and references therein), where modules are off-the-shelf classifier or regression models (a frozen neural network layer would be a particular case, as by Goutam et al. [2020]) also known as parameter-isolation. Indeed, this purely model agnostic and modular view comprises the launching pad for our study, from which we proceed in the following section.

4 Lessons from Multi-label Learning and the Case of Chaining Methods

Multi-label learning is a specific case of multi-task learning, and often can be approached as a specific case of transfer learning. Particularly, the case of chaining models. The multi-label literature has generally overlooked the potentially close connection to transfer learning. In this section we specifically study the area of overlap between these areas and extract insights to apply to the special case of model-agnostic transfer learning without task similarity, that we have detailed above.

In a multi-label task we learn from data $\{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^n$ where $\boldsymbol{x}^{(i)} \in \mathcal{X} = \mathbb{R}^d$ and $\boldsymbol{y}^{(i)} \in \mathcal{Y} = \mathbb{R}^m$ (including the most common case where $\mathcal{Y} = \{0,1\}^m$; m is the number of labels), with which we want build some model h to provide output/prediction, $\widehat{\boldsymbol{y}} = h(\boldsymbol{x})$ for test example $\boldsymbol{x}$, where target labels $\widehat{\boldsymbol{y}} = [\widehat{y}_1, \ldots, \widehat{y}_m]$ refer to a unique $\boldsymbol{x}$-in common. In various places (and in Figure 2) we generalize to multi-dimensional or meta-labels $\boldsymbol{y}_1, \boldsymbol{y}_2$; which bridges the notational gap to multi-label transfer learning: we can consider a model connectedness among blocks $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ as much as among individual labels y_1 and y_1 of a single block, without loss of generality.

4.1 Multi-label graph and chaining methods

The fact that (in transfer learning) only the target task is of interest wrt the loss metric, means any construction that the source task need not be considered beyond production of the target model; knowledge is to be transferred only in one direction, from source to target. Many aspects of multi-label learning (which supposes evaluation of all tasks/labels) are clearly not relevant, however, there are indeed classes of models which proceed in a sequential and/or hierarchical fashion, from one completed model to the next, and thus are ideally suited. For example, the family of *chaining methods* (a recent survey in [Read et al., 2021]), as deep architectures developed contemporaneously such as ADIOS [Cisse et al., 2016], and [LDL, 2017] that follow a chain-like approach. Consider the illustrations in Figure 2.

In its simplest rendition of training in chaining for the multi-label learning case, the method first learns

$$h_1 : \mathcal{X} \to \mathcal{Y}_1 \quad \text{and then}$$
$$h_2 : \mathcal{X} \times \mathcal{Y}_1 \to \mathcal{Y}_2$$

in that order (an order which can be inverted in the multi-label case, but not for transfer). Predictions are later obtained from the model as

$$\begin{aligned}
\widehat{\boldsymbol{y}}_1 &= h_1(\boldsymbol{x}) \\
\widehat{\boldsymbol{y}}_2 &= h_2(\widehat{\boldsymbol{y}}_1, \boldsymbol{x}) \\
&= h_2(h_1(\boldsymbol{x}), \boldsymbol{x})
\end{aligned} \tag{1}$$

We remark that under this chaining mechanism, training of h_2 does not require access to h_1 or any kind of joint training, because labels $\boldsymbol{y}_1 \in \mathcal{Y}_1$ are already available in the training data. Further, at test time, h_2 only requires access to h_1's predictions (not the data it was trained on, or any knowledge of its gradient or model class). Therefore, this is essentially a model-agnostic transfer from $\boldsymbol{y}_1$ to $\boldsymbol{y}_2$.

We also remark that such an approach is known to perform well (better than independent models) without any strict assumption on dependence/similarity among the label tasks [Dembczyński et al., 2012, Read et al., 2021]. There is only one reason why it does not already meet requirements for transfer learning: the fact that the source $\boldsymbol{x} \in \mathcal{X}$ is identical to both $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$; whereas we seek performance in the case that $\mathcal{X}_1 \neq \mathcal{X}_2$ (inputs from both tasks come from different distributions). Our question for the following is thus: is it possible to extend the chaining approach successfully to the transfer learning scenario without *any* particular assumption of similarity.
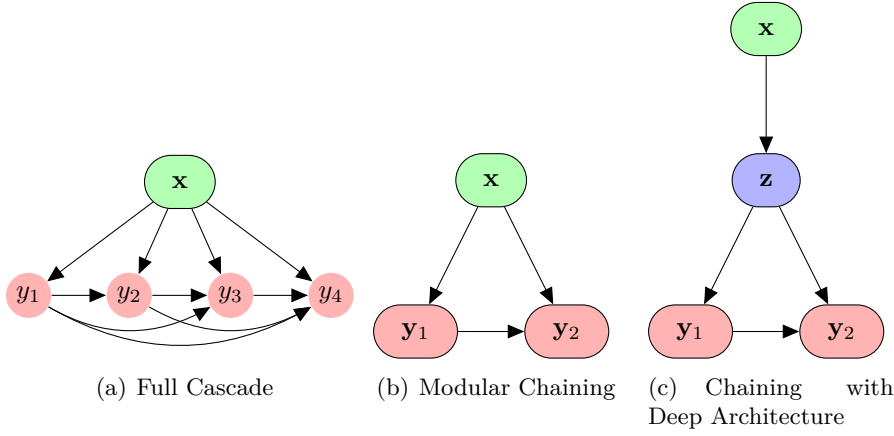
(a) Full Cascade   (b) Modular Chaining   (c) Chaining with Deep Architecture

Figure 2: A depiction of 'chaining'; (a) among each of the outputs of a single task, such that $\widehat{\boldsymbol{y}} = [\widehat{y}_1, \ldots, \widehat{y}_m] = h(\boldsymbol{x})$; and also as between two tasks; (b) with predictions invoked directly from the input, and (c) via a latent representation. This generalizes to multi-dimensional labels $\widehat{\boldsymbol{y}}_1, \widehat{\boldsymbol{y}}_2 = [\widehat{y}_{1,1}, \ldots, \widehat{y}_{1,m_1}, \widehat{y}_{2,1}, \ldots, \widehat{y}_{2,m_2}] = h(\boldsymbol{x})$.

4.2 The importance of modeling label dependence in multi-label learning

Following the seminal paper by Tsoumakas and Katakis [2007], the multi-label literature flourished, and over the years that followed many dozens of methods were presented motivated by the goal of modeling labels together under the assumption that this was necessary due to the presence of label dependence. This assumption was largely based on intuition and encouraged by empirical results, since indeed, these methods widely improved over the use of independent models.

This intuition of label relatedness was formalized in the setting of probabilistic dependence by Dembczyński et al. [2010a] (and Dembczyński et al. [2012]), in particular distinguishing between global label dependence,

$$P(Y_2|Y_1) \neq P(Y_2) \tag{2}$$

(for any two given labels Y_1 and Y_2); and conditional label dependence, after observing an input vector,

$$P(Y_2|Y_1, \boldsymbol{x}) \neq P(Y_2|\boldsymbol{x}). \tag{3}$$

But what was particularly interesting was their analysis wrt loss metrics, which revealed an apparent disconnect between the larger intuitions of the community and its empirical studies, and theoretical results. Namely, it was shown that some common multi-label metrics, such as Hamming loss, could be minimized without any consideration of label dependence, even though dozens of published results (including classifier chains) proclaimed and demonstrated advantages in practice from doing exactly that (as opposed to modeling problems individually).

Hamming loss (let us denote ℓ_H) is essentially an average of label-wise losses, thus due to the summation in the average, each label can in theory be tackled individually. On the other hand, the 0/1-loss (let us denote $\ell_{0/1}$) compares elements such that they must match exactly[2] to obtain 0 loss, and else a loss of 1; i.e., a label-vector prediction prediction is treated as a single label; and thus this encourages a model of label dependence, such that accurate combinations can be chosen even at the cost of poorer marginal predictions. The relationship between these two loss metrics is given as:

$$\ell_H(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = \sum_{j=1}^{m} [\![y_m \neq \widehat{y}_m]\!] = \ell_{0/1}(y_1, \widehat{y}_1) + \cdots + \ell_{0/1}(y_m, \widehat{y}_m) \tag{4}$$

$$\ell_{0/1}(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = [\![(\sum_{j=1}^{m} [\![y_m \neq \widehat{y}_m]\!]) > 0]\!] = [\![\ell_H(\boldsymbol{y}, \widehat{\boldsymbol{y}}) > 0]\!] \tag{5}$$

(where $[\![A]\!] = 1$ if condition A holds). Many other metrics are possible, but these two are a good illustration on the two extremes, and many methods perform well on both.

The fact that many methods perform well on both metrics is interesting. Of course, it is true that one metric can act well as a surrogate for another in the multi-label context [Dembczyński et al., 2010b,

---

[2]Note that the inverse of Hamming loss and 0/1 subset loss are known as Hamming score and *exact* match (or subset accuracy), respectively

Park and Read, 2018] (as in the general context of machine learning); i.e., a model that performs well on 0/1-loss should perform well also on Hamming loss (this is obvious in the extreme case of $\ell_H = \ell_{0/1} = 0$). But, if a classifier has been built to explicitly model label dependence, and performs better than independent classifiers, under Hamming loss which requires no such model, it suggests reasons to model labels together even if they do not demonstrate any form of statistically significant dependence. Indeed in classifier chains which is one such example, it is essentially equivalent to successfully modeling some task labels by taking into other labels whose decision is conditionally independent according to the underlying concept. It is exactly these reasons we wish to study. We do so, alongside novel development, in the following.

4.3 Beyond label dependence: other factors behind multi-label performance

We now ask: if a multi-label architecture that models labels together performs better than models which are fit independently of each other, on a task where labels are known to be independent of each other (both marginally and conditionally wrt $\boldsymbol{x}$), what explanation is there to this performance? We ask this question so that we might later exploit these mechanisms in a transfer learning setting where source and target tasks are dissimilar.

4.3.1 Non-linearity and extra capacity

For classifier chains in particular it was hypothesised that superior performance (wrt independent models) is partly due to the additional architecture of linking labels together [Dembczyński et al., 2012], much like the inner layers of a neural network (where labels in the chain 'replace' a latent/hidden-layer representation) [Read and Hollmén, 2017]. This idea has been recently further developed by Waegeman et al. [2019] (more generally for multi-output prediction) and by Read et al. [2021] (more specifically to classifier chain models).

This can be illustrated by the simplistic problem in Figure 3 (adapted from [Read and Hollmén, 2017]; and further developed in the following) and most particularly the model depicted in Figure 3(a), which is closely related to the classic solution of neural networks for the logical XOR function. However, unlike the classic example, this is done in a chain not by the use of a hidden node, rather by another label from another task – the AND function, in this case. We highlight: there is no conditional label dependence at all between these two task labels wrt the true function, i.e., $P(y_{\text{XOR}}|y_{\text{AND}}, \boldsymbol{x}) = P(y_{\text{XOR}}|\boldsymbol{x})$.

Essentially, in view of neural networks: each label acts as an extra hidden node, and all $y_j \to y_k$ arrows represent additional trainable parameters, and each predictive model (y_j-node) is an activation function. The difference is that, rather than employing back propagation (which is not possible due to the model-agnostic assumption; the 'activation functions' are classifiers though which we cannot back-propagate a gradient), we simply transfer predictive capacity from an existing trained function. Therefore, this is a kind of 'deep' neural network (admittedly in this example, in the most minimalistic definition possible), without the deep learning.

X_1	X_2	Y_{XOR}	Y_{AND}
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1

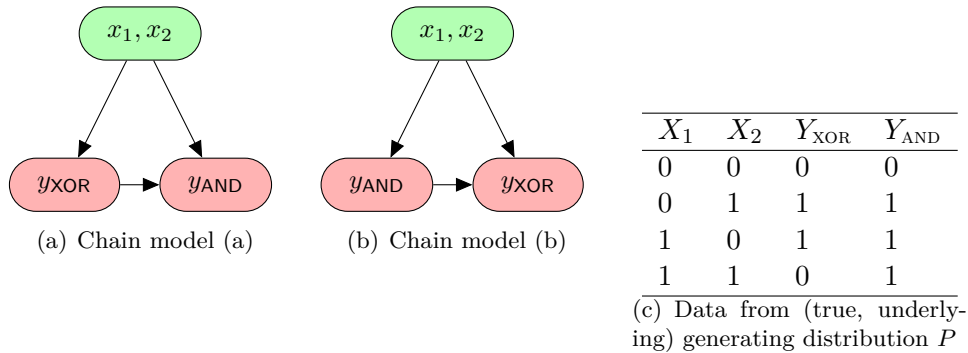(a) Chain model (a) (b) Chain model (b) (c) Data from (true, underlying) generating distribution P

Figure 3: A toy example where points of different class labels are not linearly separable (label XOR) and an example where they are (label AND). It is not a question of label dependence only, since with a linear base learner (e.g., logistic regression), the *order* of connectivity matters (yet statistical dependence/similarity metrics are symmetric). Rather than a successful model of conditional label dependence, we have gained network capacity from the AND node, providing the additional trainable parameter needed to model XOR.

4.3.2 Trading off variance for bias

Even several decades ago, the James-Stein estimator showed it was possible to benefit from modeling target variables together even if those variables are intrinsically independent from each other. This idea was taken up with regard to neural networks by Rojas [1996] but has only recently been revived in the analysis of multi-target learning by Waegeman et al. [2019] (including multi-label and multi-task classification), mentioned specifically with regard to classifier chains by Read et al. [2021], and to some extent in the deep-learning community [Havasi et al., 2020], and also in the context of multi-task learning [Feldman et al., 2014].

As Waegeman et al. [2019] explain, from the machine learning viewpoint the issue is that of regularization. Modeling multiple problems together, instead of independently, means that parameters are shared, which in turn discourages overspecialization and overfitting. We develop this view further as follows.

Suppose that we want an estimate of $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{Y}|\boldsymbol{x}]$. This task is relevant in the multi-label regression context, where $\boldsymbol{y} \in \mathbb{R}^m$, because using it as a prediction would minimize loss metrics $\ell(\boldsymbol{\mu}, \boldsymbol{y})$ based on squared-error, which are by far the most common in regression. And it is also relevant to multi-label classification; noting that $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{Y}|\boldsymbol{x}] = \sum_{\boldsymbol{y}} \boldsymbol{y} \cdot P(\boldsymbol{y}|\boldsymbol{x})$, i.e., a hypothetical weighted vote over all combinations $\boldsymbol{y} \in \{0,1\}^m$; and predictions based on this vote, i.e., $\widehat{y}_j = [\![\mu_j \geq 0.5]\!] \mid j = 1, \ldots, m$ will indeed minimize Hamming loss. Fig. 4 illustrates.
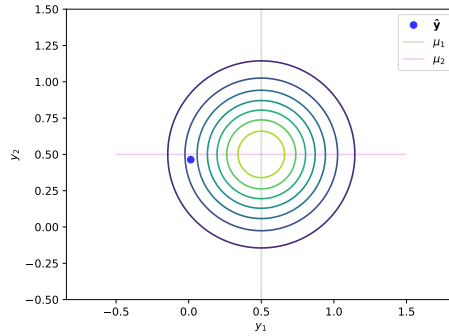

Figure 4: The multi-label/multi-target problem illustrated for $m = 2$ targets with a fictitious Gaussian to exemplify $P(\boldsymbol{y}|\boldsymbol{x})$ of mean $\boldsymbol{\mu} = [\mu_1, \mu_2]$, and estimate thereof, $\widehat{\boldsymbol{y}}$. Note that $\boldsymbol{\mu}$ is not necessarily the true value of $\boldsymbol{y}$ (for this particular $\boldsymbol{x}$) due to irreducible error; $\boldsymbol{y} = \boldsymbol{\mu} + \boldsymbol{\epsilon}$. However, on average (in expectation) predicting $\boldsymbol{\mu}$ would give us the best results under a squared- or Hamming-loss metric. Since $\boldsymbol{\mu}$ is unknown, we build a model and come up with estimate $\widehat{\boldsymbol{y}}$; exemplified in the figure. The shape of the Gaussian indicates no dependence among the two targets. Nevertheless, according to the James stein estimator it can be beneficial to model them together (at least, when $m > 2$).

Given model h we can make multi-label decision $\widehat{\boldsymbol{y}} = h(\boldsymbol{x})$, which we can think of as a sample from $P(\boldsymbol{y}|\boldsymbol{x})$. We can also bootstrap from the dataset or, for example, randomize the structure of a joint-modeling method, thus obtaining multiple models from which we obtain multiple estimates/predictions $\{\widehat{\boldsymbol{y}}^{(t)}\}_{t=1}^n$ [3]. We are essentially obtaining a number of samples $\widehat{\boldsymbol{y}}^{(t)} \sim P(\boldsymbol{y}|\boldsymbol{x}) \mid t = 1, \ldots, n$. A maximum likelihood estimate gives us unbiased estimate

$$\bar{\boldsymbol{y}} = \frac{1}{n} \sum_{t=1}^n \widehat{\boldsymbol{y}}^{(t)} \tag{6}$$

i.e., $\bar{\boldsymbol{y}} \approx \boldsymbol{\mu}$. And (differently) the James-Stein estimator:

$$\widehat{\boldsymbol{y}}_{\mathsf{JS}_n} = \frac{1 - (m-2)\frac{\hat{\sigma}^2}{n}}{\|\bar{\boldsymbol{y}}\|^2} \cdot \bar{\boldsymbol{y}} \tag{7}$$

$$= \lambda(\{\boldsymbol{y}^{(t)}\}) \cdot \bar{\boldsymbol{y}} \tag{8}$$

where we have explicitly denoted $\lambda(\cdot)$ as a mechanism to regularize $\bar{\boldsymbol{y}}$, pulling it towards 0 whenever $1 - (m-2)\frac{\hat{\sigma}^2}{n} < \|\bar{\boldsymbol{y}}\|$; shown graphically in Figure 5(a). Note that from $\{\boldsymbol{y}^{(t)}\}_{t=1}^n$ we have $n, \hat{\sigma}, m, \bar{\boldsymbol{y}}$ (enough to fully express Eq. (7)).

---

[3] Note that samples $\boldsymbol{y}^{(t)} = [y_1, \ldots, y_m] \mid t = 1, \ldots, n$ from a hypothetical posterior $\boldsymbol{y}^{(t)} \sim \widehat{P}(\boldsymbol{y}|\boldsymbol{x})$ should be distinguished from training samples $\boldsymbol{y}^{(i)} = [y_1, \ldots, y_m] \mid i = 1, \ldots, n$ from the data set, sourced originally as $\boldsymbol{y}^{(i)} \sim P(\boldsymbol{y}|\boldsymbol{x})$, as denoted elsewhere

Pulling estimates towards 0 makes particular sense in the multi-label classification case because label vectors are relatively sparse. In other words, for any given label j, the expected label $\mathbb{E}[Y_j]$ is indeed more likely to be closer to 0 than to 1. When modeling independent labels together, we impute this information in the form of a biased estimator.

Figure 5(b) and Fig. 5(c) confirm the potential for empirical gains by modeling unrelated labels together (via the James-Stein estimator) even though they are generated independently from each other. However, we also observe that these gains are very slight, except for large m and, more particularly, very small n. Which confirms earlier speculation (e.g., by Waegeman et al. [2019]).

(a) JS_1 properties: $\|\boldsymbol{y}\|^2$ vs λ ($m = 3, \sigma = 1$)   (b) JS_1 vs LS (wrt n; $m = 5$); multi-target regression

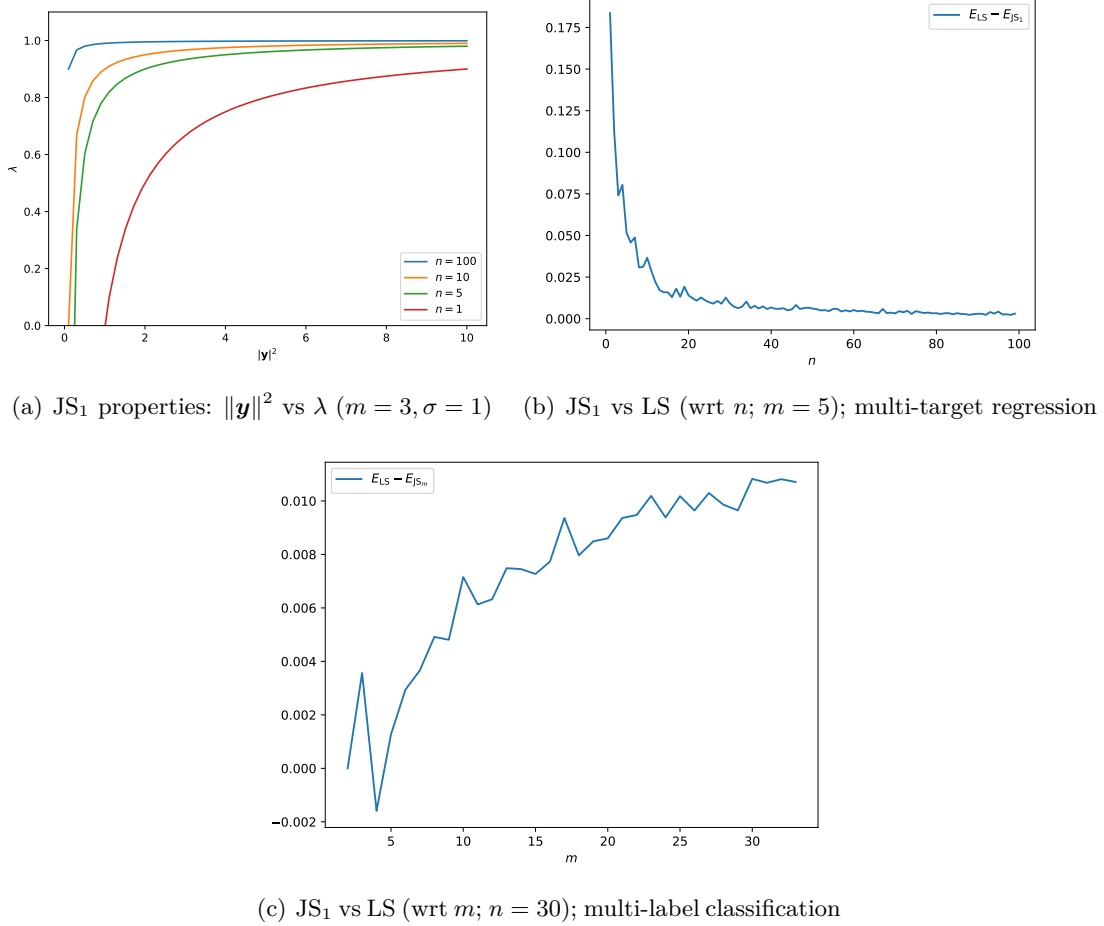(c) JS_1 vs LS (wrt m; $n = 30$); multi-label classification

Figure 5: Showing properties of (a) regularization effect of James Stein (analytical), (b) performance wrt number of training examples n and (c) wrt number of labels m (empirical). Data is synthetic, with label concepts generated completely independently of each other. E_A is error metric E (mean squared error, in these figures) on algorithm A, where JS vs LS is James Stein vs Least Squares (maximum likelihood estimate). Note that higher values of $E_{LS} - E_{JS_1}$ (improvement of JS over LS) is better. Each point has been generated from averaging over multiple runs. Stochastic gradient descent (SGD) is always the base learner (becoming logistic regression in (c) for classification).

4.3.3 The multi-label ensemble effect

Many multi-label methods are developed and presented in the context of an ensemble, then compared successfully to independent models. Success, measured as relative accuracy gain, is usually attributed to the design of the novel method component presented, and overall the proposed scheme of modeling label dependence. However, the effect of using an ensemble (vs single) model could, in many cases, be a significant component of such an accuracy; i.e., some gains are obtained regardless of the existence (or not) of dependence between the task labels. Essentially: if multi-label method X is presented in terms of 'ensembles of X' (a common phrasing in the literature), we cannot easily distinguish how effective X is, unless presented alongside ensembles of independent models; illustrated in Fig. 7(b). Indeed, ensembles of independent models have been recognized as competitive [Moyano et al., 2018].

It is true that adding more models (to an ensemble) also provides the opportunity to include more

predictive power (discussed in Section 4.3.1), as well as take into account more label dependence (discussed in Section 4.2), when making predictions. To unravel this overlap we carry out a study to specifically isolate the ensemble effect: first, we generate totally-independent regression tasks $\mathcal{X}_j \to \mathcal{Y}_j$ ($\forall j = 1, \ldots, m$) then append them together as if a single multi-output problem; and then we experiment with independent linear regression models vs a chain models of linear regression models vs ensembles of each of these. It is already established that extra capacity cannot be provided by the chains/joint modeling [Borchani et al., 2015] (because without non-linearities, the structure collapses); and similarly, we can easily see this is also the case in the multi-output ensemble: in the ensemble case, with regression weights $\boldsymbol{w}$ and ensemble averaging process (across n models, as above in Section 4.3.2):

$$\hat{\boldsymbol{y}} = \frac{1}{n} \left\{ \boldsymbol{x}\boldsymbol{w}^{(1)} + \cdots + \boldsymbol{x}\boldsymbol{w}^{(t)} + \cdots + \boldsymbol{x}\boldsymbol{w}^{(n)} \right\}$$
$$= \frac{1}{n} \boldsymbol{x}(\boldsymbol{w}^{(1)} + \cdots + \boldsymbol{w}^{(t)} + \cdots + \boldsymbol{w}^{(n)})$$
$$= \boldsymbol{w}\boldsymbol{x}$$

where, let $\boldsymbol{w}/n = \boldsymbol{w}^{(1)} + \cdots + \boldsymbol{w}^{(n)}$.

Therefore, the remaining improvement in error reduction should indicate regularization in the context of variance reduction, i.e., the 'ensemble effect'. Results are given in Fig. 6. Indeed, despite that none of the four methods have any theoretical advantage among them in terms of model capacity or the ability to model dependence (which is non existent in this case), there is a non-negligible difference in accuracy. We can also discard bias-reduction such as implied by, e.g., the James-stein estimator, since we are considering only Eq. (6).
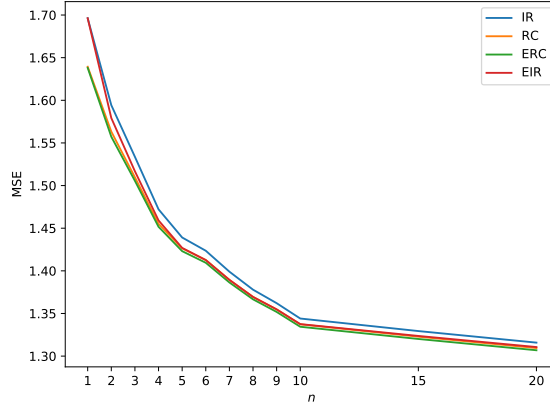

Figure 6: The mean squared error (MSE) of independent regression models (IR) vs regressor chains (RC) vs ensembles of regressor chains (ERC; each chain a random order) vs ensembles of independent regression models (EIR; standard bagging ensemble) wrt number of n, where $m = 10$. Hence, demonstrating the 'ensemble effect', which is slight, but visible. Data generation and general setup is equivalent to Figure 5; each point averaged over 100 simulations.

Vis-a-vis the James Stein estimator, this 'ensemble effect' concerns only the non-bias estimate in Eq. (6) (which appears in Eq. (8)). It is known to be the reduction of the variance component of the error, achieved by averaging votes of different estimators of the same target variable. For example, the well-established technique of bootstrap aggregating (bagging) Breiman [1996]. This effect is not specific to the context of a multi-output problem, unlike the James-Stein estimator, which will add bias to an estimate via $\lambda(\cdot)$ by considering the existence of other labels. Improvement can only be achieved on instability in the instantiation of base learner. Chained models produce this instability simply with random chain orders for each base learner which is known to produce ample diversity [Read et al., 2021], and thus the actual bagging procedure (of sampling the training data) is not needed. This is apparently (according to the results in Fig. 6 and the discussion herein) even the case when there is no dependence for such a chain to model; features in that case provide noise, which provide necessary diversity for ensemble voting to work.

4.3.4 Stacking, error propagation and correction

There is a second mechanism at play in Fig. 6. Chained models (RC) also provide improvement wrt independent regressors (RC) – despite not providing any advantage in either dependency-modeling or

predictive capacity. This is a different mechanism than vanilla ensembles. Whereas bagging expands horizontally across n models and then averages their results per label; in chained models, predictions are 'adjusted' wrt one another, in a vertical fashion of two ore more levels. This is similar to the general concept of stacking [Hastie et al., 2001]; where one prediction can affect another. Both approaches are contrasted in Fig. 7 in the multi-target context. Note that whereas in the vanilla ensemble averaging, predictions for a label/task j are averaged together (i.e., all paths leading to $\bar{y}_j$ pass only through $\{y_j^{(t)}\}_{t=1}^n$), and this is not the case in stacked models.

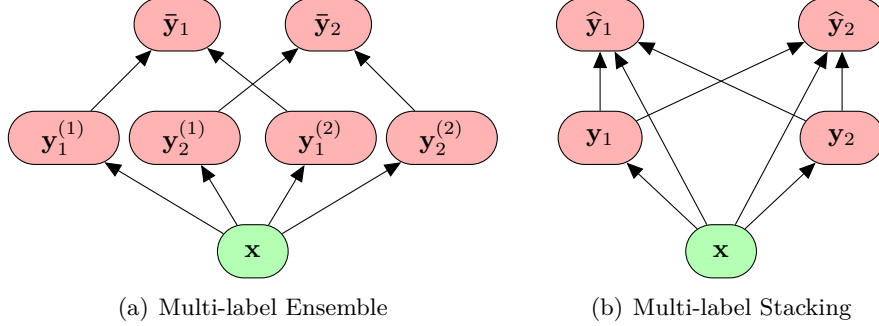(a) Multi-label Ensemble   (b) Multi-label Stacking

Figure 7: A (a) bagging ensemble (of $n = 2$ *independent* models/label concepts; not affecting each other's predictions) and (b) stacked prediction (also $n = 2$). We have kept the multi-dimensional label generalization in line with, e.g., Fig. 2; i.e., two multi-dimensional label concepts ($m = 1$); with an additional glyph over $\boldsymbol{y}$s of the second label layer – to distinguish from the first.

This mechanism of layering predictions (such as that of Fig. 7(b), but there are many similar varieties) has been credited for improvement over independent models several times in the multi-label literature, e.g., employed in [Cheng and Hüllermeier, 2009, Loza Mencía and Janssen, 2016], recently discussed by Waegeman et al. [2019] in a general multi-output context, and specifically in the context of chaining by Senge et al. [2019]. However, although stacking is known generally in machine learning, the presence of multiple labels/tasks creates a different dynamic where label predictions are corrected based on earlier predictions from *different* labels. In some contexts it is called error correction (or, equivalently claimed to reduce error propagation) but the mechanism being referred to is the same.

In reflection of the earlier development of the James-Stein estimator in Eq. (8), whereas JS biases/pulls an otherwise non-biased estimate $\bar{\boldsymbol{y}}$ according to term $\lambda(\cdot)$, bagging focusses only on the $\bar{\boldsymbol{y}}$ term, and stacking focusses only only $\lambda(\cdot)$. The relationship between the different methodologies can be expressed in the following among multi-label models $h^{(t)} : \mathcal{X} \to \mathbb{R}^m \mid t = 1, \dots, n$:

$$\widehat{\boldsymbol{y}}^{(t)} = h^{(t)}(\boldsymbol{x}) = [y_1^{(t)}, \dots, y_m^{(t)}]$$

$$\bar{\boldsymbol{y}} = \frac{1}{n} \sum_{t=1}^n \widehat{\boldsymbol{y}}^{(t)} = \left[\sum_{t=1}^n y_1^{(t)}, \dots, \sum_{t=1}^n y_m^{(t)} \right]$$

$$\widehat{\boldsymbol{y}}_{\mathsf{JS}_n} = \lambda(\{\widehat{\boldsymbol{y}}^{(t)}\}) \cdot \bar{\boldsymbol{y}}$$

$$\widehat{\boldsymbol{y}}_{\mathsf{JS}_1} = \lambda(\{\widehat{\boldsymbol{y}}^{(1)}\}) \cdot \widehat{\boldsymbol{y}}^{(1)}$$

$$= \lambda(\{h^{(1)}(\boldsymbol{x})\}) \cdot h^{(1)}(\boldsymbol{x})$$

$$\widehat{\boldsymbol{y}}_{\mathsf{Stacking}} = \lambda(\boldsymbol{x}, h^{(1)}(\boldsymbol{x}))$$

where $\lambda(\cdot)$ is, e.g., a multi-label meta model, such as the ones mapping $\lambda : \boldsymbol{y}_j \times \boldsymbol{x} \mapsto \widehat{\boldsymbol{y}}_j$ in Fig. 7(b).

How does such a mechanism (stacking) work if there is no label dependence? Although small, it is worth pointing out that the gain of using a regressor chain (no ensemble) offers already most of the gain in Fig. 6.

A single chain model can also be viewed as a particular variety of stacking. However, it is worth mentioning (as also discussed by Senge et al. [2019], Waegeman et al. [2019], Read et al. [2021]) here that stacking is not necessarily equivalent to chaining due to a potentially different training procedure concerning the source of the training label vectors $\boldsymbol{y}$s. In stacking, second-level models λ draw training data from the first layer models thus $\boldsymbol{y} \sim \widehat{P}(\boldsymbol{y}|\boldsymbol{x})$ where $\widehat{P}$ characterises the underlying probabilistic interpretation of the model h. In training chains, labels are lifted directly from the training data, i.e., sourced from the distribution $\boldsymbol{y} \sim P(\boldsymbol{y}|\boldsymbol{x})$. So, in vanilla stacking, the second layer λ reduces bias of the *individual* models h_j, but at a cost of eliminating bias in the form of label combinations (i.e., at

the cost of not modelling label dependence), and thus again the effect is mostly regularization via bias reduction, rather than benefiting explicitly from label dependence. This is confirmed by Read and Hollmén [2017], Senge et al. [2019] and others who empirically find that gains vs independent models were indeed under Hamming loss (which does not explicitly benefit from any model of existing label dependence) rather than 0/1-loss (which does).

Specific to model chains, the concept of *error propagation* has been evoked numerous times, e.g., [Senge et al., 2014] (and references therein). This refers to the idea that a poor prediction will cascade along the chain and negatively influence later predictions (i.e., more poor predictions for different labels). Interestingly, this is exactly the opposite affect that stacking is supposed to achieve (of *correcting* predictions from one label to another) and thus highlights the complexities induced by the multi-output chaining model. Obviously, one seeks to *minimize* the propagation of error along a chain of predictions.

Error propagation certainly exists under a probabilistic interpretation of chaining when modeling label dependence for the minimization of non-decomposable loss metrics such as 0/1 loss (indeed, as shown by Senge et al. [2014] under the term of 'label noise', caused by training labels coming from a different distribution to the true labels). However, in the feed-forward context, we argue that the idea of minimizing error propagation contradicts with the idea of providing predictive capacity to models further down the chain (discussed above in Section 4.3.1). One does not study whether an activation function of a neural network provides a 'correct prediction' in order to be a useful feature to the next neuron. Empirical investigations also offer little to no support for this effect of error-propagation reduction: putting easier/reliable labels first does not appear to increase chain performance [Read et al., 2021] (and references therein). And most importantly, we note that the probabilistic minimization is not relevant to transfer learning; where we are only interested and in control of the performance of the second (target) task.

4.4 A study on effect interaction

Having so far discussed the hypothetical benefits of modeling labels (i.e., tasks) together even when they bear no measurable statistical dependence among each other, the question we ask now is – to what extent these effects can be translated into gains on real-world and benchmark multi-label datasets? (in view of translating those effects into cross-domain transfer learning).

In order to investigate, we repeat the following sequence of experiments on several standard multi-label datasets; results displayed in Figure 8 (experimental details such as framework and the datasets are described in Section 5.4). The effects that we expect to capture are denoted in bold.

Exp 1. **Models of label dependence, non-linearity, regularization**: We compare independent models of vanilla logistic regression vs an ensemble of classifier chains with logistic regression as base model, evaluated under the 0/1 loss metric (Eq. (5)).

Exp 2. **Non-linearity, regularization**: As above, except we change the loss metric to a decomposable one: Hamming loss (Eq. (4)); for which no model of label dependence is required for optimum performance. Therefore any difference in performance must be explained by other mechanisms, namely extra capacity/non-linearity (given in the form of the classifier chain[4]) and regularization thereof; with variance reduction possible via both ensembles, and bias-reduction possibly via the chain structure among models. In other words: models of label-dependence should play no role here.

Exp 3. **Regularization:** As above, except now replacing logistic regression as a base classifier with multi-layer perceptrons (MLP) in both cases. MLPs provide sufficient architecture for non-linearities, so any additional capacity provided by the classifier chains should be superfluous. At this point the only gain of the ensemble of classifier chains (vs independent models) must be regularization.

Exp 4. **Spurious effects only:** As above but we now add regularization (in the form of a typical L2 weight penalty) to all MLP instantiations. According to our hypotheses, any gain of chaining found here should be treated as spurious; not likely to be replicated in general.

Results in Figure 8 confirm the widely-held belief in the literature that modeling label dependence is a core issue in multi-label learning (depending on which loss metric is under consideration), and support the idea that studies and models of such dependence can lead to strong gains over a benchmark method

---

[4]Logistic regression provides a linear decision boundary, a non-linearity is produced by converting the regression score $\in [0, 1]$ into label $\in \{0, 1\}$

that treats each label independently. However, in addition, as we hypothesized, other techniques can have a non-negligible influence. Consider the Music dataset (Figure 8(d)), one of the most widely-used benchmarks in hundreds of multi-label papers: effects *other* than label-dependence (e.g., additional architecture, and regularization) have produced on average a model around 1.2 times more performant in terms of accuracy than a typical baseline of independent models. This is highly significant. Even gains of 1.05 times the benchmark are very interesting results when there is no reason otherwise to expect any benefit at all from label dependence. The 'spurious effects' of Experiment 4 under Bird should not be regarded necessarily as suspicious. The classifier chain still represents a different model, and although theoretically the architecture of an MLP should be powerfully non-linear, for example, the additional neurons of prior nodes are small in number but may be qualitatively useful as opposed to the generic and default activation functions.

In the remainder of this paper, we look to translate these results to the domain of transfer learning.

5 Transfer Learning without Task Similarity via Transfer Chains

We infer from the results of the previous section that task similarity is only one of several aspects to consider when modeling multiple tasks together. In this section we develop the setting of model-driven model-agnostic (black-box) cross-domain transfer by removing precisely the assumption on task similarity, and nevertheless specifically leveraging the chaining mechanism. In this section we outline, justify, and explain our approach of 'transfer chains', with some empirical experiments to examine its performance.

5.1 An outline of Transfer Chains

Figure 9(a) depicts a simplified representation of our proposal. Our goal is to produce a target prediction $\widehat{\boldsymbol{y}}_T$ for given target test instance $\boldsymbol{x}_T$ (we look specifically at the classification case). We aim to make use of a source model h_S, but we have no access to any of its source training data pairs $(\boldsymbol{x}_S, \boldsymbol{y}_S)$; and we specifically denote $\tilde{\boldsymbol{x}}_S$ and $\widehat{\boldsymbol{y}}_S$ (with tilde, and hat, respectively for the input and output) to remind of this. The pipeline as follows (for notational simplicity, but without loss of generality, we depict scalar output variables rather than vectors for the remainder of this section):

$$\tilde{\boldsymbol{x}}_S = f(\boldsymbol{x}_T) \tag{9}$$

$$\widehat{y}_S = h_S(\tilde{\boldsymbol{x}}_S) = \underset{y \in \mathcal{Y}_S}{\operatorname{argmax}} \widehat{P}(y|\tilde{\boldsymbol{x}}_S) \tag{10}$$

$$\widehat{y}_T = h_T(\widehat{y}_S, \boldsymbol{x}_T) = \underset{y \in \mathcal{Y}_T}{\operatorname{argmax}} \widehat{P}(y|\boldsymbol{x}_T, \widehat{y}_S) \tag{11}$$

where h_S and h_T are the source and target task models, respectively; and f is a function mapping from the source input to target input space:

$$f : \mathcal{X}_T \rightarrow \mathcal{X}_S$$

The probabilistic interpretations $\widehat{P}$ implied by models h are not strictly part of the framework, but will be used to discuss insights from a statistical perspective in the following. They could be a parametric distribution such as in logistic regression or a distribution of votes as in, for example, a random forest.

5.2 Theoretical insights: manufactured dependence

Consider Figure 10(a) which illustrates as a probabilistic graphical model transfer-learning under the assumption that tasks Y_S and Y_T are intrinsically independent of each other (our assumption). We could conclude that the decision for Y_T does not require or benefit from joint-modeling with Y_S; since

$$P(Y_T|\boldsymbol{x}_T) = P(Y_T|\boldsymbol{x}_T, y_S)$$

$$= P(Y_T|\boldsymbol{x}_T)P(x_T)/P(x_T)\sum_{y_S} P(y_S|x_S)P(x_S)P(x_S)$$

(Y_S is marginalized out, and observed $\boldsymbol{x}$-terms cancel out); i.e., it is clear that the predictive distribution $P(Y_T|\boldsymbol{x})$ is not influenced in any way by the source task.
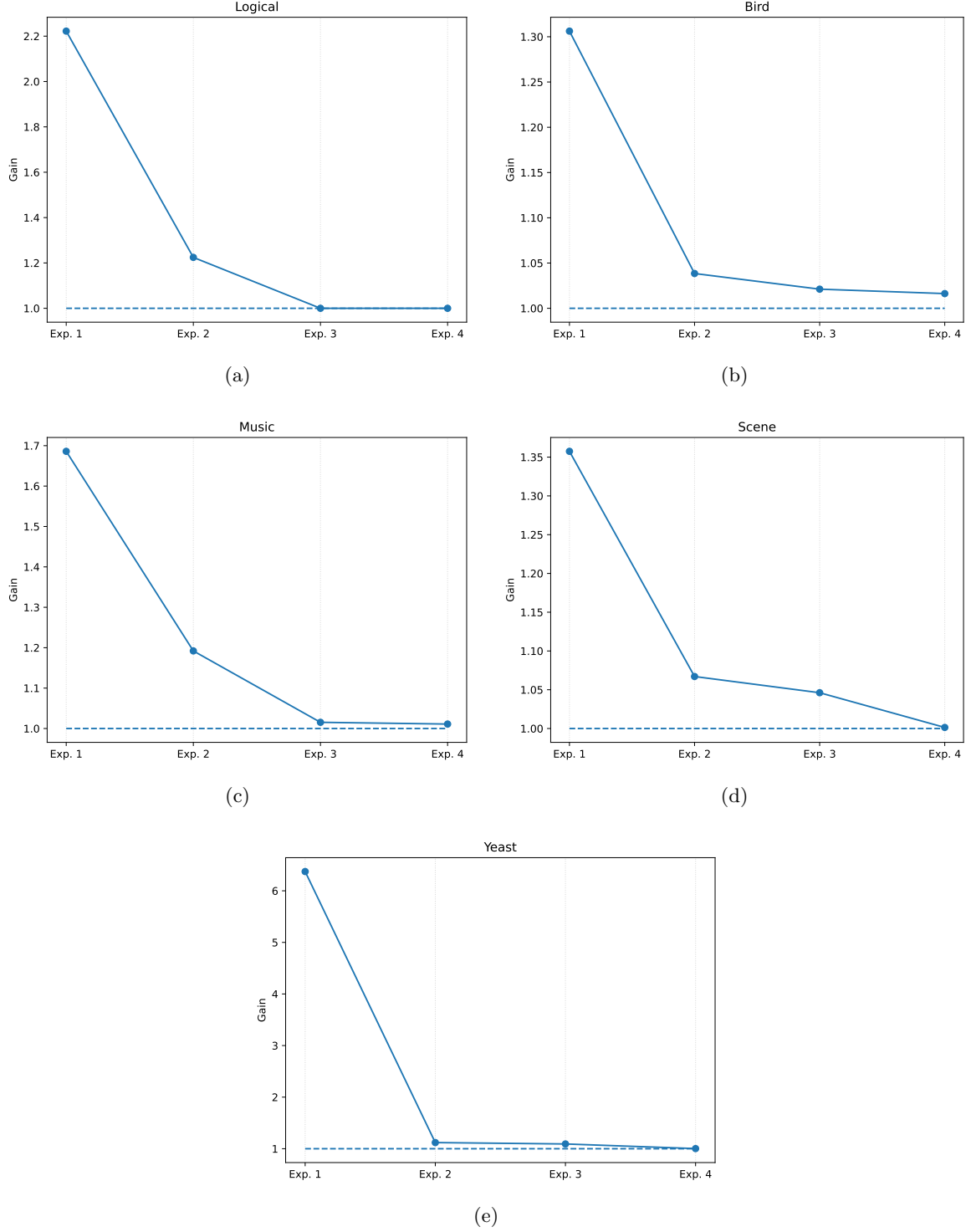
Figure 8: The gain of modeling labels together (across a cascade/chain) over independent models on well-known benchmark multi-label data sets (listed in Table 2); indicating how many times better the predictive performance is: $\mathsf{Gain} = \frac{1-\ell(\text{chains})}{1-\ell(\text{indep.})}$ where $\ell(h) \in [0,1]$ is the loss obtained by employing model h. Therefore a gain of 1.2 indicates that a chain-based model obtains 20% better predictive performance that independent models applied on the same problem. Each point is obtained as an average over 5-fold cross validation, on experiments 1.–4. as listed above.

The key observation is that at prediction time we are not dealing with true distributions P, but (as indicated in Eq. (10)) approximations $\widehat{P}$. Further, we do not approximate $P(X_S)$ or have any direct means to do so (observations $\boldsymbol{x}_S$ are never available). Rather, we map $\boldsymbol{x}_T$ into $\mathcal{X}_S$ and then to $\widehat{y}_S$ deterministically using f and h_S, respectively. This is shown in the deterministic graphical model of Fig. 9(a). Probabilistic inference can then be represented by Figure 10(b) where conditioning on both observations renders the other connectivity irrelevant.
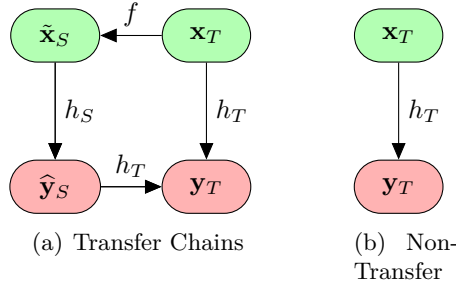
(a) Transfer Chains   (b) Non-Transfer

Figure 9: A simplified schema of (a) our proposed approach vs (b) traditional data-driven approach that does not make use of external models. Edge labels are explained via Eqs. (9)–(11). In particular, note that the connection from $\boldsymbol{x}_T \to \tilde{\boldsymbol{x}}_S$ implies an imputation/mapping step of $\boldsymbol{x}_T$ into $\mathcal{X}_S$-space; and does not involve source-task data at any point. The main difference from Figure 1 is that this is the model-agnostic approach where we cannot observe hidden layers.

The link $\widehat{Y}_S \to \widehat{Y}_S$ appears due to the fact that, usually, $\widehat{P} \neq P$ and thus potentially

$$\widehat{P}(Y_T|\boldsymbol{x}_T) \neq \widehat{P}(Y_T|\boldsymbol{x}_T, \widehat{y}_S) \tag{12}$$

This tells us that we should take into account $\widehat{y}_S$ as there is clearly conditional dependence. This dependence has been manufactured artificially by a modeling failure. Figure 10(c) provides an an example of this mechanism in the transfer learning context (extending Fig. 3(c) from earlier, which was only a multi-label problem; this time $P(\boldsymbol{X}_S)$ and $P(\boldsymbol{X}_T)$ may have different distributions): predicting the XOR label becomes conditionally dependent on the prediction of the AND label, if minimal linear modeling is used; in order to obtain best results. This example is further extended in Fig. 11.
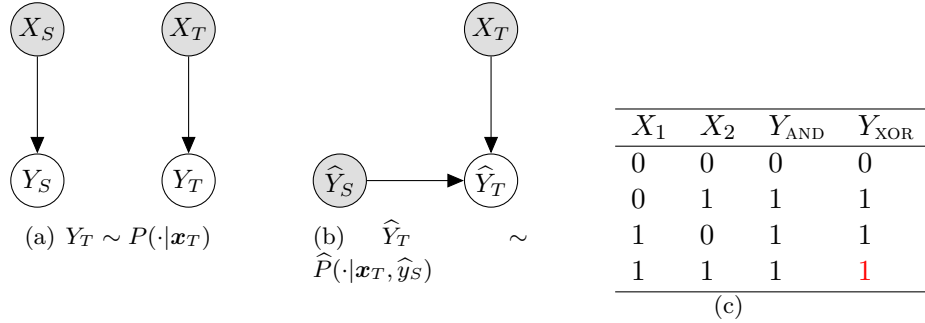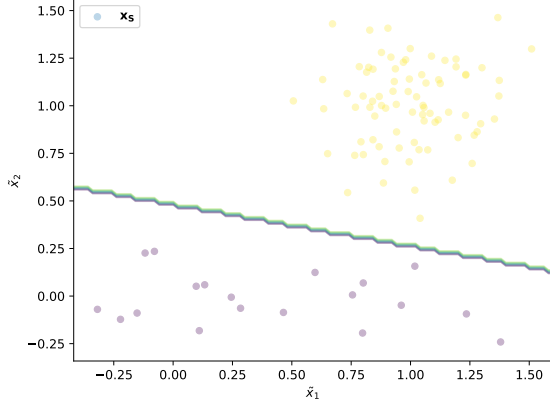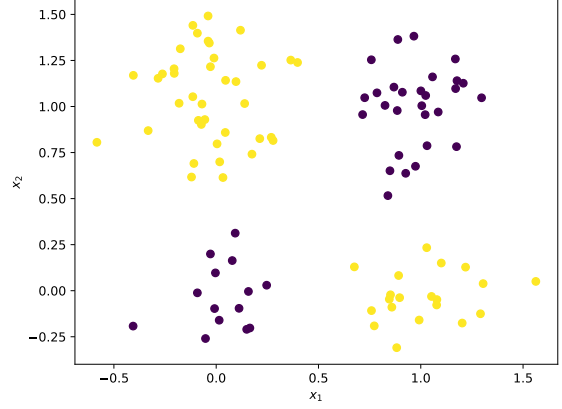

(a) $Y_T \sim P(\cdot|\boldsymbol{x}_T)$ (b) $\widehat{Y}_T \sim \widehat{P}(\cdot|\boldsymbol{x}_T, \widehat{y}_S)$

X_1	X_2	Y_{AND}	Y_{XOR}
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

(c)

Figure 10: Probabilistic graphical models illustrating (a) the generating distribution of Y_T, and (b) the approximated predictive distribution of $\widehat{Y}_T$. An example (c) illustrates where this conditional dependence arises due to insufficient capacity of the base model; extended from Fig. 3(c): supposing a linear base model class h_T, conditional dependence (and thus, a motivation to model these tasks together) has been created (the affected bit is highlighted in red), even though it is not present in the original problem. However, note that in this setting $\boldsymbol{x}_S \sim P$ is from a different and unknown distribution than $\boldsymbol{x}_T$ (as shown by (a)); the input *spaces* are however both of the same dimension (namely, $\{0,1\}^2$).

Naturally, this artificial dependence only arises when h_T has insufficient capacity or otherwise is not sufficiently effective. To the contrary, then $\widehat{y}_S$ may be uninformative or otherwise have no influence on the decision of h_T. This may also happen if h_S performs no useful function (e.g., outputting a constant or random noise), or otherwise fails to add any predictive power above what h_T already offers. Although note that this may occur even in if the tasks were intrinsically *dependent*, such as in low-noise scenarios when the target task can be solved easily and efficiently by h_T alone.

Hence the main assumption: that h_S provides some useful predictive capacity for some [source] task. And thus the challenge: find f such that $\widehat{y}_S = h_S(f(\boldsymbol{x}_T))$ is an effective feature to predicting y_T, *in addition to* $\boldsymbol{x}_T$. We are particularly interested and likely to benefit from multi-label/multi-output problems where $\widehat{\boldsymbol{y}}_S$ is a feature/label vector, simply because we are able to obtain a larger number of features from a single problem. Thus, henceforth we generalize to this case where $\widehat{\boldsymbol{y}}_S \in \mathbb{R}^m$.

(a) *Source Problem* ($y_S \equiv$ logical AND function)

(b) *Target Problem* ($y_T \equiv$ logical XOR function)

Figure 11: Toy (b) *T*arget and (a) *S*ource data sets; both generated independently of each other, as in Fig. 10(c), where both source domains are $\in \mathbb{R}^2$. Points in (a) (i.e., $\boldsymbol{x}_S \in \mathcal{X}_S$; $\boldsymbol{x}_S \sim P_{\mathcal{X}_S}$) have been faded to emphasise that we do not observe them; nor the decision boundary shown in that figure. We only have classifier/decision-rule $h_S : \mathcal{X}_S \mapsto \mathcal{Y}_T$ (the decision rule). The goal is to map $\boldsymbol{x}_T$ (points in (b), i.e., $\in \mathcal{X}_T$) into $\mathcal{X}_S$ by building an appropriate mapping $f : \mathcal{X}_T \to \mathcal{X}_S$. This mapping can be considered a success if feature $\widehat{y}_S = h_S(f(\boldsymbol{x}_T))$ leads to increased performance for a fixed target model class $h_T : \mathcal{X}_T \times \mathcal{Y}_S \to \mathcal{Y}_T$.

5.3 Considerations for the mapping-function

In Transfer Chains (recall: Eq. (9)–(11)) we use a fixed and unobserved source model $h_S : \mathcal{X}_S \to \mathcal{Y}_S$ already trained on a data to which we no longer have access and which we assume has no underlying measurable similarity to our target data set. We only know that h_S takes $|\mathcal{X}_S|$ inputs and $|\mathcal{Y}_S|$ outputs. We rely on a mapping function $f : \mathcal{X}_T \to \mathcal{X}_S$ to put $\mathcal{X}_T$ into the right shape, and place it into a space that will provide a useful transformation. This is a crucial matter, since if we cannot make use of h_S in any way, there can be no justification for our approach.

According to the development so far, once removing the assumption of task similarity, the main utility left in h_S is as its potential use as a non-linear function to improve overall predictive capacity. Existing deep neural networks approaches are an obvious choice, providing any amount on non-linear capacity, and training them is an extensively studied task. However, as motivated earlier, that is still an inherently data-centric view and ignores a vast wealth of knowledge and processing capacity stored in the form of innumerable existing models.

Our point of departure is the minimal assumption that the source model provides a useful map for *some* problem. Supposing this, we wish to put it to work for our target problem. The first obstacle is that both the distribution and dimensions of this source problem are already fixed.

We might try to 'disguise' our target input instance $\boldsymbol{x}_T$ as a source instance, suggesting the use of a transport map [Villani, 2009] between the target and source distributions, $P(X_T) \mapsto P(X_S)$. Alas, we have neither those distributions nor the data with which we can form an approximation (at least, not $P(X_S)$). And even if we did, transport maps are an expensive undertaking that do not directly address our focus point which is the predictive performance of target model h_T.

Therefore, we directly seek out the functional part of the $\mathcal{X}_S$-space wrt decision-making, which is near the decision boundary in classification (in the case of regression models, and more generally – the decision surface).

Consider Figure 11. The figure frames the problem in a visual way: how to map points $\boldsymbol{x}_T^{(i)}$ from the right (target) space into the left (source) space, in a way that $\widehat{\boldsymbol{y}}_S = h_S(f(\boldsymbol{x}_T))$ is a useful feature for predicting $\boldsymbol{y}_T$. In the context of Transfer Chains, the problem can be posed in another way: how to design map f to maximize the performance J of target model h_T on test data (supposing a fixed target and source models h)? We can cast this as a generic optimization problem

$$f^* = \underset{f \in \mathcal{F}}{\text{argmax}} \, J(f) \tag{13}$$

with performance measure

$$J(f) \equiv J(\widehat{\boldsymbol{y}}_S, \boldsymbol{y}_T | \boldsymbol{x}_T) \propto \mathbb{E}[\ell^{-1}(\boldsymbol{Y}_T, f)]; \tag{14}$$

in other words, the importance of features $\widehat{\boldsymbol{y}}_S$ generated via map f and source model h_S, wrt predicting the values $\boldsymbol{y}_T$ (alongside original inputs $\boldsymbol{x}_T$).

Since $\widehat{\boldsymbol{y}}_S$ can be treated as a vector of features for the target model h_T, we can consider existing tools for assessing feature importance, e.g., filter metrics like mutual information as well as more computationally-intensive wrapper methods (involving the (re)-training of h_T). In our experiments we indeed experiment with such measures: namely mutual information, denoted $J_I(f) = I(\widehat{\boldsymbol{y}}_S, \boldsymbol{y}_T)$; and internal cross-validation returning an average loss according to loss function ℓ on predictions $\widehat{\boldsymbol{y}}_T$ vs $\boldsymbol{y}_T$, denoted as $J_\ell(f)$ (in experiments, we use 0/1-loss (Eq. (5)) but this choice is problem-dependent). Both are fairly standard filter- and wrapper-approaches (respectively) for feature selection. Essentially we are trying to maximize and measure how much 'artificial' dependence has been manufactured during the training process.

Since the main purpose of Transfer Chains is to leverage existing non-linearities in h_S, we restrict mapping function $f \in \mathcal{F}$ to a simple linear transformation:

$$\tilde{\boldsymbol{x}}_S = f(\boldsymbol{x}_T) = \boldsymbol{U}^\top \boldsymbol{x}_T + \boldsymbol{u} \tag{15}$$

Even with this simplification, finding the best search mechanism for $f \in \mathcal{F}$ is an extensive problem beyond the scope of this paper. In experiments we use vanilla hill-climbing methodologies; with proposal function adding Gaussian noise to $\boldsymbol{U}$ and a given budget of iterations (specified in the experiment section below).

A high-level overview of the framework is given in Algorithm 1. Figure 12 provides a demonstration and insight on a toy example.

Algorithm 1 Transfer Chain: A high-level overview

1: **procedure** TRAIN($h_S, \{(\boldsymbol{x}_T^{(i)}, \boldsymbol{y}_T^{(i)})\}_{i=1}^n$)
2: $\quad f \leftarrow \arg\max_{f \in \mathcal{F}} J(f)$ $\qquad \triangleright$ Search; Eq. (13)
3: $\quad \tilde{\boldsymbol{x}}_S^{(i)} = f(\boldsymbol{x}_T^{(i)})$ $\quad \forall i = 1, \ldots, n$ $\qquad \triangleright$ Mapping; Eq. (15), and Eq. (9)
4: $\quad \widehat{\boldsymbol{y}}_S^{(i)} = h_S(\tilde{\boldsymbol{x}}_T^{(i)})$ $\quad \forall i = 1, \ldots, n$ $\qquad \triangleright$ Feature creation; Eq. (10)
5: $\quad$ Fit $h_T : \mathcal{X}_T \times \mathcal{Y}_S \to \mathcal{Y}_T$ using dataset $\{[\boldsymbol{x}_T, \widehat{\boldsymbol{y}}_S]^{(i)}, \boldsymbol{y}_T^{(i)}\}_{i=1}^n$
6: $\quad$ **return** f, h_T

7: **procedure** PREDICT($\boldsymbol{x}_T, h_S, f, h_T$)
8: $\quad \tilde{\boldsymbol{x}}_S = f(\boldsymbol{x}_T)$
9: $\quad \widehat{\boldsymbol{y}}_S = h_S(\tilde{\boldsymbol{x}}_S)$
10: $\quad$ **return** $\widehat{\boldsymbol{y}}_T = h_T(\boldsymbol{x}_T, \widehat{\boldsymbol{y}}_S)$ $\qquad \triangleright$ Eq. (11)
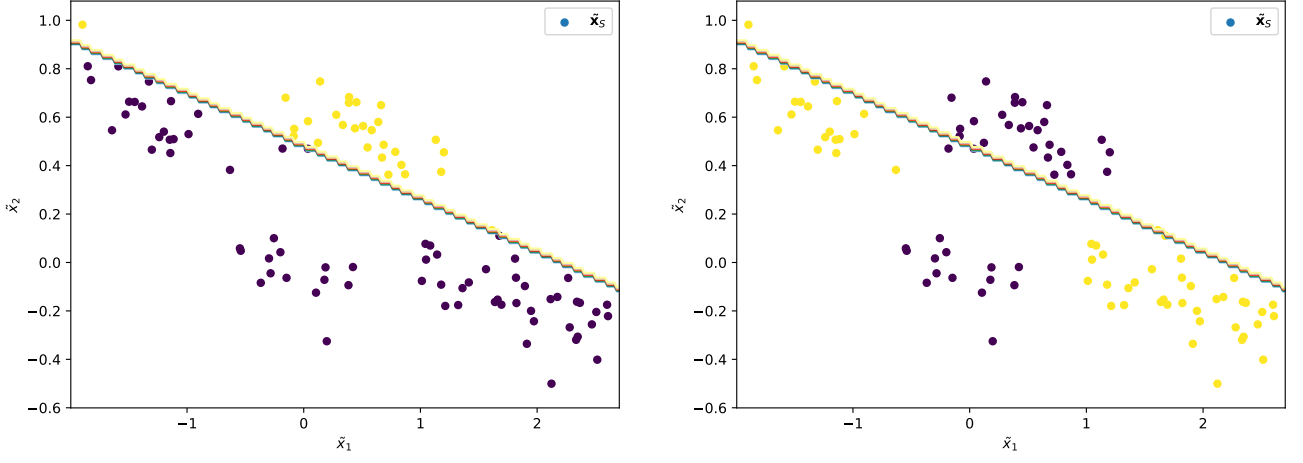

5.4 Experimentation

We conduct experiments to investigate if Transfer Chains can offer promise in practice for model-agnostic cross-domain transfer learning. As target problems we use benchmark multi-label classification datasets, listed in Table 2 alongside their corresponding source problems (drawn from the same pool of benchmark sets) It is important to recall that only black-box source models are used (not the source data) for the target problems. To that end, we train multi-label random-forest classifiers on source problems and, without loss of generality, pool them together as a single source model h_S. For example, the source model we have available when using Music as a target data set, is as

$$\widehat{\boldsymbol{y}}_S = h_S(f(\boldsymbol{x}_T)) = [h_{\mathsf{Scene}}, h_{\mathsf{Yeast}}](\tilde{\boldsymbol{x}}_S) = [\widehat{y}_{1,1}, \ldots, \widehat{y}_{1,6}, \widehat{y}_{2,1}, \ldots, \widehat{y}_{2,14}]$$

We do not use Birds or Music in the same experiment as each other's source or target because their domains are related; namely both are audio. Even though there is no other similarity beyond the audio domain, we wish to test in the context of cross-domain transfer learning in the strict sense. Multi-dimensional source datasets are important to our approach, in order to broaden the information flow towards target models (more source outputs implies more features for those models).

Transfer learning should offer some advantage vs standalone fully data-driven target models h_T. In particular, we will look for: best predictive performance, best initial predictive performance, and fastest increase in predictive performance (mentioned by Torrey and Shavlik [2010] and others as three desiderata for transfer learning). Target model classes h_T used in the experiments are described as follows where we use the concept of *step* to be able to compare at different points of training:

(a) Linear map f into $\mathcal{X}_S$, points colored as $\widehat{y}_S^{(i)}$ (AND)

(b) Linear map f into $\mathcal{X}_S$, points colored as $\widehat{y}_T^{(i)}$ (XOR)

Figure 12: Proof of concept of Transfer Chains on target problem XOR using source problem AND (continuing from Figure 11). Points $\tilde{\boldsymbol{x}}_S^{(i)} \in \mathcal{X}_S$, obtained from $\tilde{\boldsymbol{x}}_S^{(i)} = f(\boldsymbol{x}_T^{(i)})$, where $f \in \mathcal{F}$ according to a simple hill-climbing search, from Eq. (13) (via f in Eq. (15)) using performance measure $J_{0/1}(f)$ (exact match, i.e., –equivalently– inverse 0/1 loss). In (a), the color/label corresponds to $\widehat{y}_S^{(i)} = h_T(\tilde{\boldsymbol{x}}^{(i)})$. In (b), the color/label corresponds to $y_T^{(i)}$.

Table 2: Multi-label data sets (d attributes and m binary labels) and their corresponding source problems used in experiments (as black-box models h_S). Available from `https://www.uco.es/kdis/mllresources/`.

Target Dataset	n	d	m	Domain	Source Problem
XOR	200	2	1	Logical/Synthetic	AND
AND	100	2	1	Logical/Synthetic	
Music	593	72	6	Audio/Music	Scene, Yeast
Scene	2407	294	6	Image/Scene	Music, Yeast
Birds	645	260	19	Audio/Birdsong	Scene, Yeast
Yeast	2417	103	14	Microbiology	Music, Scene, Birds

- SLP: **Single-Layer Perceptron**, one logistic regression model per output trained via SGD
  At each step: An additional 100 iterations of SGD for each base learner.

- ECC: **Ensemble of Classifier Chains**, each chain in a random order
  At each step: A new base model (random chain) is added;

- MLP$_v$: **Multi-Layer Perceptron** (architectures v denoted below)
  At each step: An additional 100 iterations of SGD.

- RLP$_v$: **Random Layer Projection**: Initialized as MLP$_v$ but *not trained*
  At each step: trial new random weights; keep them if performing better (hill climbing on J_ℓ);

- TC$_v$: **Transfer Chain**. Subscripts and layers as per MLP. 100 initial search iterations $f \in \mathcal{F}$
  An additional 100 iterations of SGD for base learners

- ETC: **Ensemble of Transfer Chains**, Ensemble of random TC
  At each step: A new transfer chain (random $f \in \mathcal{F}$; then 100 iterations SGD) is added.

Where v determines the architecture, such that

- $v = 2$: 2 hidden layers of 100 units each,
- $v = 1$: 1 hidden layer of 400 units, and
- $v = 0$: 0 hidden layers (thus SLP $\equiv$ MLP$_0$).

As model-agnostic methods, we can consider any base model. We select stochastic gradient descent (SGD) since it provides a clear view of running time divided up into iterations, and thus provide a clearer picture of accuracy vs computational time invested. We use a weight decay penalty $\lambda = 0.05$

throughout. For all ensemble learners a new model (rather than additional iterations) is added at each step.

Experimental methodology: we run each method over 50 steps on the datasets (listed in Table 2), recording accuracy (exact match; inverse 0/1 loss) and computational expenditure at each step. Results are given in Figure 13.

All methods are implemented in Python making use of the scikit-learn framework [Pedregosa et al., 2011] for base models (SGD and classifier chains) where default parameters are as such unless otherwise specified above. All experiments carried out on a laptop with Intel 1.80GHz processors.

We do not compare to state-of-the-art methods in deep transfer learning which requires model introspection and manipulation and is thus not compatible with our assumptions and experimental setup. The main intention of these experiments is to provide a proof of concept supporting the discussion and development we put forward in this article, studying in particular the question of model-agnostic cross-domain transfer learning in a strict sense, and its link to multi-label learning.

Results are given in Figure 13. We see that one of variety of Transfer Chains (either ECT, or TC_v) performs best on all datasets (equal best in the case of Scene in terms of best performance). There are a large number of effects to decode, from overfitting to capacity to the influence of the underlying learning algorithm (SGD, in all cases) and the source model. Certainly results are not overall better on all datasets in all settings; yet we can come back to the main message: taking a model-based approach, even though they are black-box models, rather than a purely-data driven approach, holds promise – and this is a very interesting concept adding weight to the idea that a shift towards model-driven modeling rather than a purely data-based approach, can be an interesting opportunity to explore further. Further detailed discussion is given in the following section.
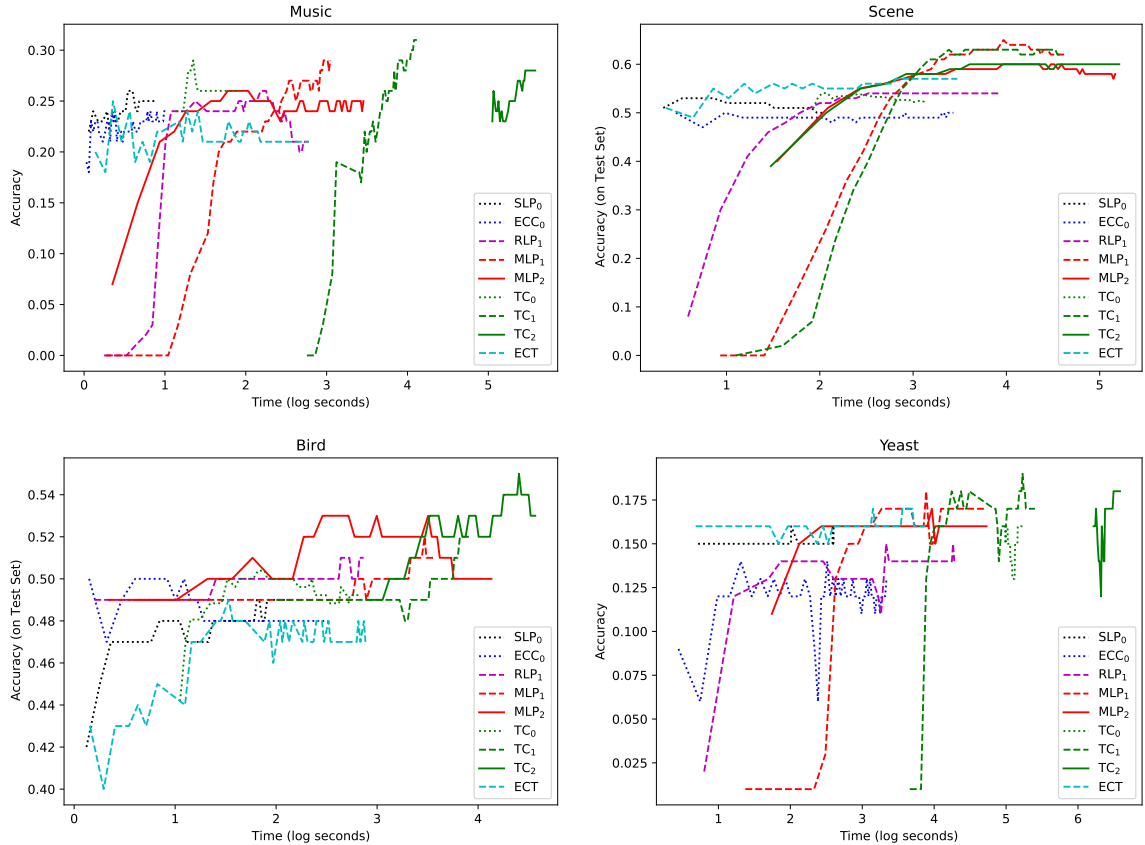

Figure 13: The performance of target models on multi-label datasets with a shuffled 60:40 train-test split, over 50 steps. Subset accuracy/exact match is given on the test set versus training time.

6 Discussion, Insights, and Implications

There is an important tradeoff between training the base models (i.e., the parameters of target model $h \in \mathcal{H}$), and finding a good mapping function (i.e., $f \in \mathcal{F}$), alongside the question of how many models (ensemble, or not) versus how much training time to invest in each of those models. These considerations are highly interdependent, since the effectiveness of the mapping function depends on

the effectivness of the target model and vice versa.

It is evident in Fig. 13, as is also well known in the literature, that simple hill-climbing has a limited over a small number of iterations on a high number of dimensions. At the same time, although the effect is limited it may nevertheless be significant.

Recall that our main aim is not to provide a new state-of-the-art method for transfer learning, because such methods assume task similarity, available source training data, or transparency of the source model. Our approach assumes none of these. It is already clear that task similarity is an important ingredient to success of transfer learning, just as label dependence is a major factor in the success of many multi-label and multi-task methods. However we have sought (in both cases) to highlight and demonstrate that it is not the *only* ingredient and to develop a study around the question of other aspects.

Therefore, although one can easily argue that transfer chains does not constitute an immediately attractive option in consistent efficient performance, its performance – alongside the development and ablation studies earlier – nevertheless raises interesting points of discussion and here we reflect upon these and the implications they can have in future research.

6.1 The potential of model-agnostic cross-domain transfer chains

Our experimental results of Section 5.4 support the hypothesis derived from our analysis (in Section 4): in the absence of task similarity, the gain for considering tasks together is arguably slight in some circumstances, yet non-negligible and even important in other cases.

One argument can be that gains could be met by more careful fine tuning and hyper-parametrization of existing data-driven approaches, but there is another argument that we must take a step to the side in order to take a step forward: massive data-driven learning approaches are not tenable to those who do not have the data, and computational expensive to those who do. And even in the context of enormous reserves of data and computational resources to process it, it unrealistic to rely on this approach, and it is certainly not the case in human learning or in nature.

From our developments, it appears apparent that much of the gain from the chain transfer is an addition of structural capacity and predictive power from the base model. Multi-layer networks with non-linear activation functions (represented by MLP) can provide enormous model capacity. However, providing some of this capacity via model-agnostic transfer appears to convey certain advantages. It some cases this is sufficient to reduce the need for (or at least depth of) back-propagation.

If predictive architecture non-linear capacity is an important factor, it is true that random projections are cheap method to obtain this, and are well known to the machine learning literature in many forms. However, although their inner layers are produced instantly as random weight matrices (and, because of this) they are notoriously wasteful, and indeed in Fig. 13 exhibited limited success in the form of RLP.

There is a qualitative difference in Transfer Chains, which uses random models only in the sense of selected from a pool of source models that are not likely to be related to the target task more than any other given task; the models themselves are not randomly initialized. Rather, since we make the assumption they were constructed by some machine learning framework we know their structure is efficient (formally: maximizing likelihood with some form of regularization) for at least some task and that they are already available. Hence also, we by this they are not wasteful; no additional creation or additional memory costs. The only challenge is having to find a relatively useful mapping to and from such models.

We showed that this process is similar to artificially creating task dependence. This also revives the question: what is similarity between two tasks. Drawing two tasks randomly from the pool of all available tasks much have some underlying structural connection, even if it is just by accident. Naturally overfitting is then a risk (as with random projections and basis function expansion and so on) but with sufficient regularization (another inherent benefit from multi-task learning), a satisfactory fit can be obtained, thus recycling capacity, rather than learning it from scratch.

6.2 Implications for learning from data streams with abrupt shifts in concept

Earlier (Section 3.1) we mentioned the case of data streams; a well-known area of research building methods to deal with data arriving constantly in a theoretically-infinite stream. Within this area, adapting to concept drift is the widely-known and relevant challenge – essentially a special case of transfer learning – of identifying and jettisoning knowledge which is no longer relevant and learning

the new concept as quickly as possible (at the same time avoiding catastrophic forgetting, i.e., of failing to retain knowledge which will be useful again in the future (a reoccurring concept))).

While a full consideration is beyond the scope, we can concisely say (references provided in Section 3.1) that the state of the art methods in the data-stream literature will typically look to detect drift, destroy existing structure which represents knowledge from the previous concept in order to free up memory, and allow it to regrow/reform based on the new concept – i.e., to actively promote forgetting. Following this argument, when drift is abrupt and complete (old concept bears no similarity to current concept), one could justify abandoning the existing models completely.

Our study indicates that such a reaction (destroying models referring to a no-longer valid concept) would be a major mistake even in cases of total and abrupt drift. Rather, one should aim to incorporate the old models into the current decision process for the new concept. This is precisely the best opportunity for methods such as Transfer Chains: following the start of a new concept, there is a small amount of target data, meaning that both other mechanisms we looked at – the 'James-Stein effect' and the 'chaining effect' are likely to render most benefit, and further more the presence of an model is already available and in place from the previous concept.

Furthermore, using a model in this way would be immediately relevant in the case of a recurrent concept. Recurrent concepts are already studied, but what our results indicates in that using models can be reused even if not explicitly appearing again in the stream in the same form. Yet we could suspect this mechanism could already be in play in data streams. Concept drift detection is notoriously imprecise, and so models are kept around much longer than they need to be. Further analysis in the context of data streams would be needed to properly develop this hypothesis; even if such a study is outside the scope of this article.

6.3 A fresh look at connections to human and biological transfer

There is no novelty in drawing connections to human learning when we discuss transfer learning, as an immense amount of literature exists (much of it since before machine transfer learning became known or popular). But we can highlight that the kind of transfer learning in humans is, like the approach we investigate, also mainly model-driven learning. Indeed, it is now well known that the human brain is not a data storage device in the sense of storing original data for faithful recall later [Pinker et al., 1997], at least not in the same sense as a database or csv file. On observation and interaction our brains develop neural pathways for processing information, and these pathways can be developed regardless of objective task similarity. For example, learning a foreign language may benefit achievement in mathematics [Stewart, 2005], and the study of mathematics is commonly motivated beyond contexts where it can be directly applied. This raises the same question we have studied: there is a non-trivial overlap between the concepts of transfer based on knowledge-based similarity, or of structural-similarity, or simply a transfer of capacity.

Indeed, one view is that chain transfer essentially repurposes a source model, and gain an effect for which that model was not initially designed (again: transfer – or simply making use of – existing capacity). We can find this throughout biology, and it is clearly observed in the Covid19 pandemic: global behavior was modified with planet-wide impact via a relatively simple mechanism (a virus). Admittedly, the ensuing behaviour was not a target model of the virus, and much alteration of global activity was specifically against its own payoff metric (of reproduction), but it provides ample demonstration of the scale of leverage on/repurposing of a complex pre-existing structure by a comparatively simple vector.

6.4 Limitations versus future potential: a place for model-agnostic cross-domain transfer

With the development and study of transfer chains we pushed towards the development of more model-driven learning approaches, moving a way from purely data-driven learning. Such an approach provides a novel methodology towards model re-use and recycling, and away from single-use models involving memory-wasteful and computationally-expensive from-scratch techniques. It is worth restating the main benefit: transfer learning that is employable using off-the-shelf black-box modules, not necessarily of a particular model class or framework.

We have also acknowledged that explicitly denying aspects of task similarity comes potentially at a major cost in terms of accuracy during transfer learning. As a short-term goal it may not be the most practical way to achieve improvement in a target model. However, under a larger and more long-term view, its potential is greater, and one could envision a large library of source models to select from.

Due to the model-agnostic and model-driven nature, this could foster more easily collaboration among practitioners without compromising privacy of data sources.

There are many avenues for improvement. We have so far only used relatively simplistic off-the-shelf base learners and hill-climbing methodologies. We can speculate that increasing the complexity of these methodologies as well as increasing the number of source tasks have improve the accuracy on the target task via a transfer chain, but further research would be needed to confirm this.

We have only looked at multi-label classification for the source and target tasks. Having multiple outputs is useful in overcoming an information bottleneck, but there is a vast array of task that could be considered: multi- and structured-output regression, time series, data streams, reinforcement learning, etc. In particular, we did not yet consider recurrent architectures, but these architectures are still used in practice with learning regimes other than those based on gradient descent, including an area of research known as reservoir computing [Lukoševičius and Jaeger, 2009]. This would be a clear candidate for future investigation, alongside reinforcement learning. In transfer chains, not only does the source data not need to be similar to the target data, but the task definition does not need to be the same either; such that recurrent connections among decision tree-regression models could be used as part of the representation for reinforcement learning. Future work will investigate such possibilities.

7 Conclusion

In this article we studied, developed, and leveraged mechanisms in multi-label and transfer learning other than task similarity, which is usually seen as a fundamental motivating assumption in these areas; removing it was a central point of our study. Namely, we looked initially in the multi-label case (which involves a large corpus of literature), where task similarity takes the form of label dependence. By removing the assumption of label dependence, we brought to surface theoretical insights on other mechanisms, such as effects of regularization and additional capacity that arise naturally in connecting different tasks. And we provided an empirical study to isolate and examine these effects in practice in benchmark multi-label datasets. We showed that these mechanisms, which have been hitherto underappreciated or even disregarded completely, can significantly impact the performance in multiple task systems. We made use of these findings to develop an approach to the extremely challenging goal of model-agnostic (black-box) model-driven (no source data) cross-domain (no task similarity) transfer learning. Despite this difficult setting, with our methodology of *transfer chains* inspired from related concepts in the multi-label literature, that arose from our study, we were able to demonstrate attractive results on a number of standard data sets, in most cases out-competing the established methods.

We did not question the important positive association between task similarity and predictive accuracy in transfer learning. Thus, by removing the assumption of similarity, our goal was not to produce a new state-of-the-art in multi-label or transfer learning, but rather we provided greater understanding on the significance and potential of effects other than task similarity; we developed some theoretical insights behind these effects, and demonstrated and isolated them in practice. This is a relatively small step, but we argue it has important implications for future development. We discussed a few of these implications; in particular, we elaborated a discussion on adapting to concept drift in data streams, among other possibilities.

We identified many areas for promising future developments of model-agnostic transfer learning, including using recurrent modular structures and reinforcement learning.

References

Learning deep latent spaces for multi-label classification. 2017. URL `https://arxiv.org/pdf/1707.00418v1.pdf`.

F. Alet, T. Lozano-Perez, and L. P. Kaelbling. Modular meta-learning. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 856–868. PMLR, 29–31 Oct 2018. URL `http://proceedings.mlr.press/v87/alet18a.html`.

J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48, 2016.

H. Borchani, G. Varando, C. Bielza, and P. Larrañaga. A survey on multi-output regression. *Wiley Int. Rev. Data Min. and Knowl. Disc.*, 5(5):216–233, Sept. 2015. ISSN 1942-4787. doi: 10.1002/widm.1157. URL `http://dx.doi.org/10.1002/widm.1157`.

L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. ISSN 0885-6125.

R. Chandra and A. Kapoor. Bayesian neural multi-source transfer learning. *Neurocomputing*, 378: 54–64, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.10.042. URL `https://www.sciencedirect.com/science/article/pii/S0925231219314213`.

X. Chen, A. H. Awadallah, H. Hassan, W. Wang, and C. Cardie. Multi-source cross-lingual model transfer: Learning what to share. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3098–3112, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1299. URL `https://aclanthology.org/P19-1299`.

Y. Chen, A. L. Friesen, F. Behbahani, A. Doucet, D. Budden, M. Hoffman, and N. de Freitas. Modular meta-learning with shrinkage. 33:2858–2869, 2020. URL `https://proceedings.neurips.cc/paper/2020/file/1e04b969bf040acd252e1faafb51f829-Paper.pdf`.

W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009. doi: 10.1007/s10994-009-5127-5.

M. Cisse, M. Al-Shedivat, and S. Bengio. Adios: Architectures deep in output space. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 2770–2779, New York, New York, USA, 20–22 Jun 2016. PMLR.

J. Davis and P. Domingos. Deep transfer via second-order markov logic. In *Proceedings of the 26th annual international conference on machine learning*, pages 217–224, 2009.

K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On label dependence in multi-label classification. In *Workshop Proceedings of Learning from Multi-Label Data*, pages 5–12, Haifa, Israel, June 2010a.

K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. Regret analysis for performance metrics in multi-label classification: the case of hamming and subset zero-one loss. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 280–295. Springer, 2010b.

K. Dembczyński, W. Waegeman, W. Cheng, and E. Hüllermeier. On label dependence and loss minimization in multi-label classification. *Mach. Learn.*, 88(1-2):5–45, July 2012. ISSN 0885-6125. doi: 10.1007/s10994-012-5285-8.

K. Dembczyński, W. Waegeman, and E. Hüllermeier. An analysis of chaining in multi-label classification. In *ECAI: European Conference of Artificial Intelligence*, volume 242, pages 294–299. IOS Press, 2012. ISBN 978-1-61499-097-0.

K.-L. Du and M. N. Swamy. *Neural Networks and Statistical Learning*. Springer Publishing Company, Incorporated, 2013. ISBN 144715570X, 9781447155706.

S. Feldman, M. R. Gupta, and B. A. Frigyik. Revisiting stein's paradox: multi-task averaging. *J. Mach. Learn. Res.*, 15(1):3441–3482, 2014.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44:1–44:37, 2014. doi: 10.1145/2523813. URL `http://doi.acm.org/10.1145/2523813`.

M. Gasse. *Probabilistic Graphical Model Structure Learning : Application to Multi-Label Classification*. Theses, Université de Lyon, Jan. 2017. URL `https://hal.archives-ouvertes.fr/tel-01442613`.

H. M. Gomes, J. P. Barddal, F. Enembreck, and A. Bifet. A survey on ensemble learning for data stream classification. *ACM Comput. Surv.*, 50(2):23:1–23:36, Mar. 2017. ISSN 0360-0300. doi: 10.1145/3054925. URL `http://doi.acm.org/10.1145/3054925`.

K. Goutam, S. Balasubramanian, D. Gera, and R. R. Sarma. Layerout: Freezing layers in deep neural networks. *SN Computer Science*, 1(5):1–9, 2020.

O. Grisel. All about scikit-learn, with olivier grisel. 2021. Accessed April 2021.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

M. Havasi, R. Jenatton, S. Fort, J. Z. Liu, J. Snoek, B. Lakshminarayanan, A. M. Dai, and D. Tran. Training independent subnetworks for robust prediction. *arXiv preprint arXiv:2010.06610*, 2020.

Y. Hsu, Z. Lv, and Z. Kira. Learning to cluster in order to transfer across domains and tasks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https:// openreview.net/forum?id=ByRWCqvT-`.

G.-B. Huang, D. Wang, and Y. Lan. Extreme learning machines: a survey. *International Journal of Machine Learning and Cybernetics*, 2(2):107–122, 2011. ISSN 1868-8071. doi: 10.1007/ s13042-011-0019-y.

A. Karbalayghareh, X. Qian, and E. R. Dougherty. Optimal bayesian transfer learning. *IEEE Transactions on Signal Processing*, 66(14):3724–3739, 2018.

E. Loza Mencía and F. Janssen. Learning rules for multi-label classification: a stacking and a separate-and-conquer approach. *Machine Learning*, 105(1):77–126, Oct 2016. ISSN 1573-0565. doi: 10.1007/ s10994-016-5552-1. URL `https://doi.org/10.1007/s10994-016-5552-1`.

M. Lukoševičius and H. Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

J. M. Moyano, E. L. G. Galindo, K. J. Cios, and S. Ventura. Review of ensembles of multi-label classifiers: Models, experimental study and prospects. *Inf. Fusion*, 44:33–45, 2018. doi: 10.1016/j. inffus.2017.12.001. URL `https://doi.org/10.1016/j.inffus.2017.12.001`.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10): 1345–1359, Oct. 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2009.191. URL `http://dx.doi.org/ 10.1109/TKDE.2009.191`.

L. A. F. Park and J. Read. A blended metric for multi-label optimisation and evaluation. In *ECML-PKDD 2018: 29th European Conference on Machine Learning*, pages 719–734, 2018. URL `http: //www.ecmlpkdd2018.org/wp-content/uploads/2018/09/608.pdf`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

S. Pinker et al. *How the mind works*, volume 524. New York Norton, 1997.

A. Rahimi, B. Recht, et al. Random features for large-scale kernel machines. In *NIPS*, volume 3, page 5. Citeseer, 2007.

J. Read and J. Hollmén. Multi-label classification using labels as hidden nodes. Technical Report 1503.09022v3, ArXiv.org, 2017. URL `http://arxiv.org/abs/1503.09022v3`. ArXiv.

J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains: A review and perspectives. *Journal of Artificial Intelligence Research (JAIR)*, 70:683–718, 2021. URL `https://jair.org/index.php/ jair/article/view/12376`.

R. Rojas. *Modular Neural Networks*, pages 411–425. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996. ISBN 978-3-642-61068-4. doi: 10.1007/978-3-642-61068-4_16. URL `https://doi.org/10. 1007/978-3-642-61068-4_16`.

R. Senge, J. J. del Coz, and E. Hüllermeier. On the problem of error propagation in classifier chains for multi-label classification. In M. Spiliopoulou, L. Schmidt-Thieme, and R. Janning, editors, *Data Analysis, Machine Learning and Knowledge Discovery*, pages 163–170, Cham, 2014. Springer International Publishing. ISBN 978-3-319-01595-8.

R. Senge, J. J. del Coz, and E. Hüllermeier. Rectifying classifier chains for multi-label classification. *arXiv preprint arXiv:1906.02915*, 2019.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

J. H. Stewart. Foreign language study in elementary schools: Benefits and implications for achievement in reading and math. *Early Childhood Education Journal*, 33(1):11–16, 2005.

C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

L. Torrey and J. Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

N. Tripuraneni, M. Jordan, and C. Jin. On the theory of transfer learning: The importance of task diversity. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7852–7862. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/59587bffec1c7846f3e34230141556ae-Paper.pdf`.

G. Tsoumakas and I. Katakis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3(3):1–13, 2007.

C. Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

W. Waegeman, K. Dembczyński, and E. Hüllermeier. Multi-target prediction: a unifying view on problems and methods. *Data Mining and Knowledge Discovery*, 33(2):293–324, Mar 2019. ISSN 1573-756X. doi: 10.1007/s10618-018-0595-5. URL `https://doi.org/10.1007/s10618-018-0595-5`.

K. Wang, X. Gao, Y. Zhao, X. Li, D. Dou, and C.-Z. Xu. Pay attention to features, transfer learn faster cnns. In *International Conference on Learning Representations*, 2019.

P. Zhao, L.-W. Cai, and Z.-H. Zhou. Handling concept drift via model reuse. *Machine Learning*, 109 (3):533–568, 2020.

B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. URL `https://arxiv.org/abs/1611.01578`.