A Constraint-Based Approach to Learning and Explanation*

Gabriele Ciravegna,[1,2] **Francesco Giannini,**[2] **Stefano Melacci,**[2] **Marco Maggini,**[2] **Marco Gori**[2]
[1]Department of Information Engineering (DIINFO), University of Florence, Florence, Italy
gabriele.ciravegna@unifi.it
[2]Department of Information Engineering and Science (DIISM), University of Siena, Siena, Italy
{fgiannini, mela, maggini, marco}@diism.unisi.it

Abstract

In the last few years we have seen a remarkable progress from the cultivation of the idea of expressing domain knowledge by the mathematical notion of constraint. However, the progress has mostly involved the process of providing consistent solutions with a given set of constraints, whereas learning "new" constraints, that express new knowledge, is still an open challenge. In this paper we propose a novel approach to learning of constraints which is based on information theoretic principles. The basic idea consists in maximizing the transfer of information between task functions and a set of learnable constraints, implemented using neural networks subject to L_1 regularization. This process leads to the unsupervised development of new constraints that are fulfilled in different subportions of the input domain. In addition, we define a simple procedure that can explain the behaviour of the newly devised constraints in terms of First-Order Logic formulas, thus extracting novel knowledge on the relationships between the original tasks. An experimental evaluation is provided to support the proposed approach, in which we also explore the regularization effects introduced by the proposed Information-Based Learning of Constraint (IBLC) algorithm.

1 Introduction

We consider a generic multi-task learning problem in which the goal is to learn a set of unknown functions, also called task functions, given some labeled examples and knowledge about the way the tasks are related. While some of the relationships among tasks are known proprieties of the considered environment, new relation patterns, that are not known in advance, could also exist. This paper presents a learning framework in which, instead of only focusing on the development of the task functions accordingly to the available knowledge, we are also interested in determining new knowledge. We consider the case in which such knowledge is represented by a set of logic rules that compactly explain how the task functions are related in some regions of the input space (e.g., co-occurrence on some samples, exclusivity, ...). Given a test example, our approach can provide a logic-based explanation of the discovered relationships in the space region to which the example belongs. Another innovative aspect of the proposed approach is that the learned knowledge conditions the development of the task functions themselves. This introduces a regularization effect also improving the overall accuracy of the developed functions.

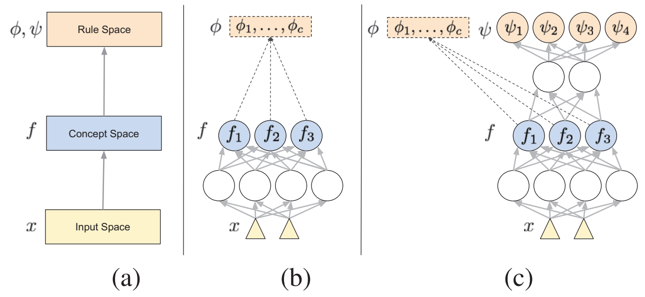


Figure 1: (a) Input, concept and rule spaces, that are associated to input data x, task functions $f(x)$ and constraints $\phi_j(f(x)) = 0$, $j = 1, \ldots, c$, and $\psi_z(f(x)) = 0$, $z = 1, \ldots, m$, respectively. (b) Learning task functions subject to *given* constraints $\phi_j(f(x)) = 0$, $j = 1, \ldots, c$ (dotted connections are identity mappings). (c) Learning task functions subject to given constraints and *learning* of constraints $\psi_z(f(x)) = 0$, $z = 1, \ldots, m$, where $m = 4$ in this example.

More formally, in this framework we distinguish among the problem of learning a set of task functions in the *input space*, subject to given constraints that implement the available knowledge on the problem at hand, and the problem of learning "new" constraints in the *concept space* to which the task functions belong. Constraints belong to what we refer to as the *rule space*, as sketched in Fig. 1 (a). Learning new knowledge is the outcome of a developmental process in which the task functions are progressively learnt in the input space (stage 1) and, afterwards, it is learnt how the task functions are related in the concept space (stage 2). An alternate optimization scheme is progressively iterated, refining the task functions subject to given-and-new constraints, and improving the newly devised constraining functions.

*This work was partly supported by the PRIN 2017 project RexLearn, funded by the Italian Ministry of Education, University and Research (grant no. 2017TWNMH2).

In this paper, we consider the generic framework of *learning from constraints* (Gnecco et al. 2015), that nicely generalizes the most popular learning settings by means of the unifying notion of "constraint" (see Table 4 in (Gnecco et al. 2015) for a list of examples). Learning is conceived as the problem of finding those task functions that are subject to a number of constraints that represent the available knowledge on the considered problem (Fig. 1 (b)). The optimal solution is the one that is the most parsimonious (regular, smooth), making the problem well-posed. A key feature of such a framework is that symbolic knowledge, unambiguously expressed by means of First-Order Logic (FOL) formulas, can be converted into constraints among the task functions and enforced on a subset of the training set or on all such data (whether they are supervised or not) (Diligenti, Gori, and Sacca 2017). Any supervision represents a simple form of pointwise constraint, i.e. $f_i(x_h) - y_h = 0$, where (x_h, y_h) is a supervised pair and f_i is a certain task function.

While it is pretty common to search for solutions that are consistent with a given set of constraints, learning new constraints to adapt to the environment is still an open challenge. In this paper we propose a novel approach to *learning of constraints* in the concept space, which is rooted on information theoretic principles. The basic idea consists in maximizing the transfer of information between the task functions and another set of learnable functions. While it is known that maximizing the Mutual Information from the input space to the task functions leads to an unsupervised process that is inherently related to data clustering (Melacci and Gori 2012), here we consider the case in which we maximize the information transfer from the concept space to the rule space, where the aforementioned learnable functions are neural networks (Fig. 1 (c)). This process yields the unsupervised discovering of new constraints among the tasks, that are fulfilled in almost-disjoint sub-portions of the concept space. Such newly devised knowledge is not immediately explainable using formal descriptions and, for this reason, we provide a simple procedure that can explain the learned constraints in terms of FOL formulas, thus extracting symbolic knowledge on the relationships between the original tasks. This procedure is based on the idea of reducing the number of paths in the networks that implement the new constraints, exploiting L_1 regularization. This choice allows us to produce compact FOL formulas and it is mostly motivated by simplicity. An experimental evaluation is provided to support the proposed approach, where we also explore the regularization effects introduced by the proposed Information-Based Learning of Constraint (IBLC) algorithm.

Related Work

There exists a number of works sharing the idea of learning constraints. They are usually intended to satisfy the available ground truth (supervisions) on a given dataset (De Raedt, Passerini, and Teso 2018). The learned constraints have to be expressed in some (mathematical) language and a common choice falls on logic formulas (Valiant 1984; Bessiere et al. 2017; Pawlak and Krawiec 2017; Kolb et al. 2018), such as in the case of FOL in Inductive Logic Programming (Muggleton 1991; De Raedt et al. 2016). The case of learning hard-constraints is generally plagued by large algorithmic complexity, thus several approximations were proposed. Approaches that are about learning soft-constraints usually aim at discovering the weights of a large set of clauses (Richardson and Domingos 2006; Bach et al. 2015), or both clauses (structure) and weights (Campigotto, Battiti, and Passerini 2015; Yang, Yang, and Cohen 2017). Differently, in our framework any constraint is implemented as a neural network and the learning setting can be carried out as any traditional learning of neural network parameters. Only at the end of the learning stage we provide a symbolic interpretation of the networks.

Symbolic interpretation of neural networks (rule extraction) has been the subject of many researches by several authors, especially in the nineties. Some approaches are about Fuzzy Logic (Kasabov 1996; Huang and Xing 2002; Castro and Trillas 1998; Di Nola, Gerla, and Leustean 2013), but are generally less straightforward in terms of explainability than the ones based on Boolean Logic (Fu 1991; Towell and Shavlik 1993; Tsukimoto 2000; Sato and Tsukimoto 2001). In the latter case, it is pretty common to rely on a discretization of the input and output values of the neurons, pruning the network to keep it simple. The interpretation we propose in this paper follows this approach, mostly in the spirit of (Zilke, Mencía, and Janssen 2016) where Boolean interpretation is applied neuron-by-neuron to a deep neural network that is trained to solve a specific task, decomposing the interpretation of the whole network to its subconstituents. However, we follow a different paradigm, both developing a set of task functions and another network that, in turn, learns how the tasks are related (constrained).

This paper is organized as follows. Section 2 introduces the learning framework, including the Information-Based Learning of Constraint (IBLC) algorithm, whose outcome is explained using the strategies of Section 3. Section 4 collects the experimental results and Section 5 concludes the paper.

2 Formulation of Learning

We consider a multi-task learning environment composed of n task functions, $f_1, \ldots, f_n$, compactly represented by the vectorial function $f = [f_1, \ldots, f_n]$, where $f : \mathcal{X} \subset \mathbb{R}^d \to \mathcal{Y} \subset \mathbb{R}^n$. The perceptual information $x \in \mathcal{X}$ is processed and it is mapped to the concept space by $f(x)$ (Fig 1 (b)). For example, in a multi-class classification problem, n is the number of classes, and f_i is the classifier associated to the i-th class. We assume that we are given a discrete collection of data $X = \{x_1, \ldots, x_N \,:\, x_k \in \mathcal{X}\}$, of task-specific knowledge (for example, supervisions on some of the available data), and of other knowledge that is about the environment in which the system operates, such as relationships among the task functions.

Learning from Constraints. Several ad-hoc solutions can be considered to inject the available knowledge into the learning process that yield the f_i's. The framework of *learning from constraints* (Gnecco et al. 2015) offers a generic environment in which the unifying notion of "constraint" is exploited to model the available knowledge and to enforce it into the learning process. Such knowledge is implemented using a number of constraints that involve (a subset of) the task functions f, and they are represented with $\phi_j(f(x)) = 0$, $j = 1, \ldots, c$,[1] where $\phi = [\phi_1, \ldots, \phi_c]$, $\phi : \mathcal{Y} \subset \mathbb{R}^n \to \mathcal{Z} \subset \mathbb{R}^c$, being c the number of constraints. It is pretty common to embed each ϕ_j into a non-negative penalty function, indicated with $\hat{\phi}_j$, that is evaluated over a subset of the input space and that expresses the cost for not respecting the equality constraint $\phi_j(f(x)) = 0$ (notice that sometimes the introduction of a penalty function could also not be needed, for example when ϕ_j is already non-negative, and then we have $\hat{\phi}_j = \phi_j$). If $X_{\phi_j} \subseteq X$ are the data points belonging to such subset, then we aim at minimizing the penalties together with a regularization term that enforces the smoothness. Formally[2],

$$f^\star = \arg\min_f U(f)$$
$$= \arg\min_f \left\{ \sum_{j=1}^{c} \sum_{x_k \in X_{\phi_j}} \hat{\phi}_j(f(x_k)) + \gamma_f \|f\| \right\} \quad (1)$$

where the generic regularization term based on the norm of f is weighted by $\gamma_f > 0$, and we assumed that customizable scaling factors are embedded into the penalty terms. For example, consider the classic supervision constraint $\phi_j(f(x)) = f(x) - y(x) = 0$, and suppose we want to enforce it on a single data point x_s. Then, we could embed it into a quadratic penalty function $\hat{\phi}_j(f(x)) = \|f(x) - y(x)\|^2$, and the second summation of Eq. 1 will involve x_k belonging to $X_{\phi_j} = \{x_s\}$. However, the role of ϕ_j in the considered learning framework is more general, since ϕ_j could implement different types of knowledge, for example the one expressed by means of a FOL formula. For further details we recommend (Gnecco et al. 2015).

In this paper we consider the case in which f is modeled by a feed-forward neural network with n output neurons, each of them associated to a component of f (Fig. 1 (b)).

Learning of Constraints. Enforcing the *given* constraints $\phi_j(f(x)) = 0$, $j = 1, \ldots, c$, results in enforcing regularities in the concept space that are dictated by ϕ. Learning *new* constraints $\psi_z(f(x)) = 0$, $z = 1, \ldots, m$ corresponds with discovering regularities in the concept space, modeled by the vectorial function $\psi = [\psi_1, \ldots, \psi_m]$, with $\psi : \mathcal{Y} \subset \mathbb{R}^n \to \mathcal{Z} \subset \mathbb{R}^m$, where m is the number of learnable constraints.

We propose to implement ψ using a feed-forward neural network with m output neurons, each of them associated to a component of ψ (Fig. 1 (c)). Such network is not the only element that the system is expected to learn, since also the subsets of data on which each constraint is enforced are *unknown* and must be estimated. We indicate with $X_{\psi_z} \subseteq X$ the subset associated to ψ_z, and $X_\psi = \{X_{\psi_z},\ z = 1, \ldots, m\}$.

[1]Inequality constraints could be available as well, but to simplify the descriptions we focus on equality constraints (inequality constraints $\phi_j(f(x)) \leq 0$ can be converted into an equality by $\max(0, \phi_j(f(x)) = 0)$.

[2]In order to simplify the notation, here and in the rest of the paper we will not make explicit the dependence of objective functions on the data X, and of eventually available supervisions.

We can define a problem similar to the one of Eq. 1, where the variables to be optimized are ψ and X_ψ, that is

$$\psi^\star, X_\psi{}^\star = \arg\min_{\psi, X_\psi} D(\psi, X_\psi, f)$$
$$= \arg\min_{\psi, X_\psi} \left\{ \sum_{z=1}^{m} \sum_{x_k \in X_{\psi_z}} \hat{\psi}_z(f(x_k)) + \gamma_\psi \|\psi\| \right\} \quad (2)$$

where $\gamma_\psi > 0$ and $\hat{\psi}_z$ denotes a penalty term associated to each ψ_z. This problem is clearly ill-posed, being solved by ψ constantly equal to 0 or by m equivalent constraints. Moreover, we are posing no conditions on X_ψ, that is now a variable of the learning problem, and on the portion of X covered by $\cup_{z=1}^m X_{\psi_z}$, so that several not-useful solutions are possible.

Information-Based Learning of Constraints

In order to devise a valid formulation of Eq. 2, we follow the intuition that ψ should maximize the information transfer from the concept space to the rule space, to capture the regularities of the concept space itself. In particular, maximizing the Mutual Information (MI) under a smoothness condition is a feasible way to learn membership functions associated to clusters (Melacci and Gori 2012).

For any ψ_z we introduce the probability distribution $P_{\Psi=z|Y=f(x)}(\psi, f(x))$, that can be regarded as the probability that ψ_z is fulfilled on a concept-space projection $f(x) \in \mathcal{Y}$ of the example $x \in \mathcal{X}$. The notation Ψ indicates a discrete random variable associated to the constraints, while Y is a random variable associated to the data in the concept space, respectively. The MI between the input data, projected into the concept space, and the rule space is computed by $I_{Y,\Psi}(\psi, f)$, where

$$I_{Y,\Psi}(\psi, f) = H_\Psi(\psi, f) - H_{\Psi|Y}(\psi, f)\ , \quad (3)$$

being H_Ψ the entropy associated to Ψ, while $H_{\Psi|Y}$ is the conditional entropy of Ψ given Y. In detail, $H_\Psi(\psi, f) =$

$$-\sum_{z=1}^{m} \left(\int_{\mathcal{Y}} P_{\Psi=z|Y=y}(\psi, y) dy \right) \log \left(\int_{\mathcal{Y}} P_{\Psi=z|Y=y)}(\psi, y) dy \right)$$

and $H_{\Psi|Y}(\psi, f) = -\sum_{z=1}^m \int_{\mathcal{Y}} P_{\Psi=z|Y=y}(\psi, y) dy$. If we assume $\mathcal{Y}$ to be uniformly distributed and discretized on the data $\{f(x) :\ x \in X\}$, we can easily replace the integrals with summations on $x \in X$, scaled by $\frac{1}{N}$.

The MI is maximized when, on average, all the m constraints are fulfilled in an unbiased way, that is $\int_{\mathcal{Y}} P_{\Psi=z|Y=y}(\psi, y) dy = \frac{1}{m}$, due to the maximization of the entropy H_Ψ. Moreover, due to the maximization of the negative conditional entropy $-H_{\Psi|F}$, only one of the m constraints is fulfilled on each $f(x)$, i.e., for each $f(x)$ there is only one $P_{\Psi=z|Y=f(x)}$ that is 1 for a certain z, while the other $P_{\Psi=h|Y=f(x)}$, $h \neq z$, are 0. As a result, maximizing MI leads to the development of m different constraints that are fulfilled in a mutually exclusive way on each $f(x)$, and so on each data sample $x \in X$. This allows us to inherently determinate the set X_ψ, since, for each z, we have that

$X_{\psi_z} = \{x \in X \ : \ \psi_z(f(x)) = 0\}$, avoiding the need of treating X_ψ as a variable of the learning problem (Eq. 2)

If we maximize the MI on the data X (indicated with the penalty term $\hat{I}_{Y,\Psi}$) in a soft way and under a smoothness assumption, then we end-up in a feasible objective that can be used to learn the constraining function ψ, that is a valid formulation of the problem in Eq. 2. In detail,

$$D(\psi, X_\psi, f) \approx \overline{D}(\psi, f) = \left\{ -\hat{I}_{Y,\Psi}(\psi, f, X) + \gamma_\psi \|\psi\| \right\} , \tag{4}$$

where the set X_ψ is not anymore a variable of the problem.

We are left with the problem of defining how $P_{\Psi=z|Y=f(x)}(\psi, f(x))$ is computed. We must ensure that $\sum_z P_{\Psi=z|Y=f(x)} = 1$ as well as that each $P_{\Psi=z|Y=f(x)} \geq 0$, but we also have to take care that the z-th constraint is fulfilled whenever $P_{\Psi=z|Y=f(x)} = 1$, i.e., $\psi_z(f(x)) = 0$. A probabilistic normalization can be built using the softmax function, inverting the signs of the activations,

$$P_{\Psi=z|Y=f(x)}(\psi, f(x)) = \frac{e^{-\beta\psi_z(f(x))}}{\sum_{h=1}^{m} e^{-\beta\psi_h(f(x))}} , \tag{5}$$

that, however, yields the same probabilities even if we add an offset to the activations, thus violating the aforementioned requirements (β is a customizable scaling parameter). Moreover, whenever $e^{-\beta\psi_z(f(x))}$ numerically dominates $e^{-\beta\psi_h(f(x))}$'s, $h \neq z$, we will end-up in having probability ≈ 1 even if $\psi_z(f(x)) = 0$ is not fulfilled. We can easily circumvent these problems by requiring each ψ_z to be bounded $[0, \eta]$ (e.g., using sigmoidal activation functions we have $\eta = 1$), and forcing conditions on the extremes of the interval, i.e., $P_{\Psi=z|Y=f(x)}(\psi, f(x)) = \theta \approx 1$ when $\psi_z(f(x)) = 0$ and $\psi_h(f(x)) = \eta$, $h \neq z$. This leads to: $\beta = \log(\theta(m-1)) - \log(1-\theta)$, and we set $\theta = 0.99$.

Finally, we notice that due to the soft enforcement of the MI in Eq. 4, the system can actually learn multiple constraints that are fulfilled on the same x, thus relaxing the aforementioned mutual exclusivity. Even if out of the scope of this paper, we could also include the functions ϕ in the summations of Eq. 5 and of the entropy terms, thus encouraging the system to learn new constraints ψ that are different from the given ϕ (a condition that, in general, is not necessarily met).

Stage-based Learning

The problem of learning from – and of – constraints can be formalized as,

$$f^\star, \psi^\star = \arg\min_{f,\psi} \left\{ U(f) + \overline{D}(\psi, f) \right\} , \tag{6}$$

where U and $\overline{D}$ are defined in Eq. 1 and Eq. 4, respectively. However, if we directly minimize Eq. 6 we actually enforce the maximization of the MI between the input space (instead of the concept space) and the rule space, due to the dependence of $\overline{D}$ by f. We will see in Sec. 4, that this configuration, that we will call Global Optimization, is difficult to be optimized and leads to unsatisfactory results. Therefore, we propose a Stage-based Optimization procedure (formalized in Algorithm 1) in which we first learn the task functions subject to the given constraints only (stage 1 – only $U(f)$ is involved), and then we learn new constraints in the concept space, keeping the task functions fixed (stage 2 – only $\overline{D}(\psi, f)$ is involved). Afterwards, we apply an iterative process that is based on optimizing the whole Eq. 6, alternately keeping fixed either the constraints or the task functions. In other words, we further refine the task functions by the learned constraints (emphasizing their relationships) and, in turn, we refine the constraints by means of the updated task functions (in this case, we do not consider the term $U(f)$, since it is not function of ψ). This procedure is repeated $T-1$ times, and it is guaranteed to converge since we keep minimizing the same functional.

Algorithm 1 Stage-based Optimization procedure (superscripts indicate the iteration number).

```
1: initialize: $f^{(0)}$, $\psi^{(0)}$
                                   ▷ initialization of the network weights
2: $f^{(1)} \leftarrow \arg\min_f U(f^{(0)})$
                                   ▷ learning the task functions (stage 1)
3: $\psi^{(1)} \leftarrow \arg\min_\psi \overline{D}(\psi^{(0)}, f^{(1)})$
                                   ▷ learning of constraints (stage 2)
4: for $t = 1$ to $t = T-1$ do
5:     $f^{(t+1)} \leftarrow \arg\min_f U(f) + \overline{D}(\psi^{(t)}, f^{(t)})$
                                   ▷ refinement of the task functions
6:     $\psi^{(t+1)} \leftarrow \arg\min_\psi \overline{D}(\psi^{(t)}, f^{(t+1)})$
                                   ▷ refinement of the learned constraints
7: end for
8: output: $f^\star = f^{(T)}$, $\psi^\star = \psi^{(T)}$
```

3 Interpretation of the Learned Constraints

The constraint learning procedure described so far leads to the development of constraints $\psi_z(f(x)) = 0$, $z = 1, \ldots, m$, that do not provide an explicit symbolic description of the discovered relationships among the involved task functions f. Neural networks are generally considered as *black boxes* and providing an explanation of their decision mechanism is not straightforward and generally requires complex techniques. In this paper we focus on possible interpretations of the network ψ from a logical point of view (as also considered in (Di Nola, Gerla, and Leustean 2013; Zilke, Mencía, and Janssen 2016)), motivated by the usefulness of FOL to represent knowledge. A logical formula can be seen as a symbolic description of a certain truth-function, in this case related to the functional behaviour of each ψ_z.

We propose to exploit a *Boolean interpretation* of the neural network that implements ψ. We consider task functions f that return values in $[0, 1]$, thus modeling the truth degree of a set of logic predicates. Similarly, we focus on functions ψ that are in $[0, 1]$ too, thus fulfilling the requirements on the bounds introduced in Section 2. The main idea consists in approximating the input of each neuron with the vertices of the Boolean hyper-cube, while the neuron output is approximated with a Boolean value. In detail, the hidden units are squashed in $[0, 1]$ (we used sigmoids) and, since the functions f are in $[0, 1]$ as well, each neuron of the ψ-network learns a linear separation surface in the space $[0, 1]^{q_i}$, being

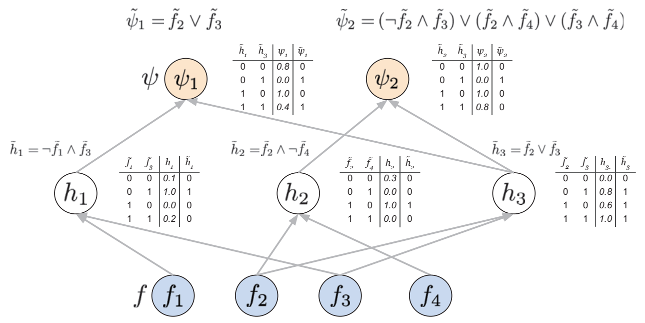
q_i the fan-in of the generic i-th neuron. We select a decision threshold (we used 0.75), so that the separation surface yields a $\{0,1\}$-valued function denoting the membership of any input pattern to the (false) 0-class or (true) 1-class. If we restrict the neuron input space to the vertices of the Boolean hyper-cube $\{0,1\}^{q_i}$, then any neuron can be interpreted as a Boolean function and therefore it can be described according to its associated truth-table.

The truth-table of each neuron can be easily converted into its logical representation in *Disjunctive Normal Form* (DNF) considering the disjunction ($\vee$) of conjunctions ($\wedge$) of the literals ($v, \neg v$) corresponding to the rows for which the Boolean function is true. By composing the truth tables accordingly to the network structure (see Fig. 2), any ψ_z can be represented by a Boolean formula $\tilde{\psi}_z$ that depends on the Boolean propositional variables $\tilde{f}$ of the task functions f. Any formula $\tilde{\psi}_z$ turns out to be true (evaluated to 1) for any pattern x satisfying the constraint $\psi_z(f(x)) = 0$, i.e., for the sake of clarity, the truth-function associated to $\tilde{\psi}_z$ corresponds to $1 - \psi_z$. In Fig. 2 we report an example of a neural network implementing ψ, where each neuron is paired with a truth table and a logic description by means of a Boolean formula. Finally, in order to get a FOL formula that describes $\psi_z(f(x)) = 0$, quantified with $\forall x \in X_{\psi_z}$, we replace the Boolean variables $\tilde{f}$ with the predicates whose truth value is computed by the task functions f.

Dealing with Complexity

Unfortunately, the size of such truth-table (and the complexity of its corresponding formula) is exponential in the number of its input variables and this can generally limit the scalability of this approach. In order to get low-level complexity formulas, we first notice that we can restrict the input space of each neuron by observing how the neuron behaves when the network makes predictions in the given training data. In fact, in the case of the input of the i-th neuron, we can reasonably consider only the subset of configurations of the Boolean hyper-cube $\{0,1\}^{q_i}$ that are actually triggered when making inference on the training data. Secondly, we assume that any neuron has only few edges with a not-null weight, whose number is defined by the custom parameter $q \geq 1$. In order to implement this assumption, we chose the L_1-norm ($||.||_1$) on the network weights as regularization term of Eq. 4, that facilitates the sparsity of the connections.

In order to reach the condition in which the fan-in of each neuron is q, for the Stage-based Optimization we propose to progressively prune the network, fixing eventually generated inconsistencies due to the possibly disruptive pruning operation. In detail, after step 3 and step 6 of Algorithm 1, where the ψ-network is updated, for each neuron, we prune the input weight with the smallest absolute value. Then, we refine the pruned network, repeating the ψ-update step (step 3 or step 6). At the end of Algorithm 1, in order to ensure that we preserve exactly q input connections for each neuron, we keep pruning and refining until all the fan-ins are equal to q. Regarding the Global Optimization, instead, the whole process is performed at the end of the learning epochs. In Fig. 2 we report an example of ψ-network with $q = 2$.



$\tilde{\psi}_1 = \tilde{f}_2 \vee \tilde{f}_3$
$\tilde{\psi}_2 = (\neg\tilde{f}_2 \wedge \tilde{f}_3) \vee (\tilde{f}_2 \wedge \tilde{f}_4) \vee (\tilde{f}_3 \wedge \tilde{f}_4)$
ψ
ψ_1
ψ_2
$\tilde{h}_1 = \neg\tilde{f}_1 \wedge \tilde{f}_3$
$\tilde{h}_2 = \tilde{f}_2 \wedge \neg\tilde{f}_4$
$\tilde{h}_3 = \tilde{f}_2 \vee \tilde{f}_3$
h_1
h_2
h_3
f
f_1
f_2
f_3
f_4

$\tilde{h}_1$	$\tilde{h}_3$	ψ_1	$\tilde{\psi}_1$
0	0	*0.8*	0
0	1	*0.0*	1
1	0	*1.0*	0
1	1	*0.4*	1

$\tilde{h}_2$	$\tilde{h}_3$	ψ_2	$\tilde{\psi}_2$
0	0	*1.0*	0
0	1	*0.0*	1
1	0	*1.0*	0
1	1	*0.8*	0

$\tilde{f}_1$	$\tilde{f}_3$	h_1	$\tilde{h}_1$
0	0	*0.1*	0
0	1	*1.0*	1
1	0	*0.0*	0
1	1	*0.2*	0

$\tilde{f}_2$	$\tilde{f}_4$	h_2	$\tilde{h}_2$
0	0	*0.3*	0
0	1	*0.0*	0
1	0	*1.0*	1
1	1	*0.0*	0

$\tilde{f}_2$	$\tilde{f}_3$	h_3	$\tilde{h}_3$
0	0	*0.0*	0
0	1	*0.8*	1
1	0	*0.6*	1
1	1	*1.0*	1

Figure 2: An example of ψ. Hidden and output neurons are paired with the truth table (right) and the corresponding logic description (top) devised as described in Section 3. The truth tables include the real-value neuron outputs (third column) and the Boolean approximation (last column). The logic descriptions of ψ_1, ψ_2 are the outcome of composing the descriptions of the hidden neurons.

As a possible drawback of this approach, the choice of keeping low the fan-in of each neuron could affect the capability of the neural network to learn global rules, e.g. a disjunctive clause among all the possible task functions. However, networks with stacked layers progressively compose the computations, allowing us to recover more general formulas involving several task functions still keeping low the size of the truth-tables. Even if the Boolean interpretation can be affected by some approximation mistakes (we removed those min-terms that are always false), some promising experimental results showing the effectiveness of this approach are reported in Section 4.

4 Experimental Results

Our experimental analysis aims at evaluating the automatically devised FOL-based explanations of the relationships among the learned task functions. The learned constraints, in turn, have regularization effects on the task functions, hence, we also evaluate how their generalization skills are affected. We compare: the Baseline model where only the task functions are learned (no constraints are learned); the Global Optimization algorithm, where the constraints ψ_z and the task functions are jointly learnt by directly optimizing Eq. 6; the Stage-based Optimization algorithm, where the learning process consists in multiple repetitions of a task-learning stage and a constraint-learning stage, as described in Algorithm 1. We considered two challenging settings built on two popular datasets, MNIST and PASCAL Part[3], that we refer to as MNIST Even/Odd and PASCAL Multi.

In all the experiments, the model parameters have been cross-validated ranging them in the following grids: learning rate of the Adam optimizer $\in \{0.01, 0.001, 0.0001\}$, $\gamma_\psi \in \{1e-2, 1e-3, 1e-4\}$, $\gamma_f = 1e-6$ (we used the squared norm of the weights to implement $\|f\|$). We

[3]The datasets can be downloaded at http://yann.lecun.com/exdb/mnist/ and https://www.cs.stanford.edu/~roozbeh/pascal-parts/pascal-parts.html

Table 1: Accuracy values ($\pm$ standard deviation) and learned rules in MNIST Even/Odd dataset in the case of two configurations of the ψ network (a-b, see the main text for details), varying the amount of labeled examples per class.

# Labeled Examples	Baseline	Stage-based Optimization	Global Optimization
1	31.7 ± 2.1 %	**46.5$\pm$ 1.4 %**	38.7 ± 3.7%
10	68.6 ± 0.4 %	71.5$\pm$ 0.9%	**71.6 $\pm$ 1.7%**
100	84.5 ± 0.4 %	**86.7$\pm$ 0.3 %**	86.0$\pm$ 0.1%

Learned Rules			Samples from X_{ψ_z}
(a)	$\forall x \in X_{\psi_1},\ Odd\ \dot{\vee}\ Even$	$\forall x \in X_{\psi_2},\ Eight\ \dot{\vee}\ Zero$	$\in X_{\psi_1}$ $\in X_{\psi_2}$
	$\forall x \in X_{\psi_3},\ Nine \wedge \neg Even$	$\forall x \in X_{\psi_4},\ Six \wedge \neg Odd$	$\in X_{\psi_3}$ $\in X_{\psi_4}$
(b)	$\forall x \in X_{\psi_1},\ Odd \wedge \neg Even \wedge [(One\ \dot{\vee}\ Five\ \dot{\vee}\ Seven\ \dot{\vee}\ Nine) \vee (\neg One \wedge \neg Five \wedge \neg Seven \wedge \neg Nine)]$		$\in X_{\psi_1}$
	$\forall x \in X_{\psi_2},\ Even \wedge \neg Odd \wedge \neg One \wedge \neg Five \wedge \neg Nine \wedge (Eight \vee \neg Eight)$		$\in X_{\psi_2}$

also considered different scaling factors $\in \{0.5, 1, 2\}$ of the penalty term associated to the MI (Eq. 4), and a convex combination of the two entropy terms, modulated by a coefficient $\in \{0.25, 0.5, 0.75\}$. All experiments are based on 1000 epochs. For the stage-based optimization, the task function learning stage lasts 200 epochs, while the constraint learning stage lasts 50 epochs. The same holds for the successive refinements. All the reported results are averaged over 5 runs.[4]

MNIST Even/Odd. Let us consider the task of handwritten digits classification, where the task functions are the predicates f_i, $i = 0, \dots, 9$, denoting the memberships to the digit classes, and the predicates f_{even}, f_{odd}, denoting whether the input digit is even or odd. The 12 task functions are implemented as a multilayer perceptron (MLP) with 30 hidden and 12 output units. Here and in the following experiments, we indicate the task functions with more readable labels, such that $Even$ for $f_{even}(x)$ and $Zero$ for $f_0(x)$ (and so on). In this experiment we consider two configurations with (a) $m = 2$ and (b) $m = 10$ constraints, implemented by single-output neural networks ψ_z, without hidden layers, and we selected our pruning procedure to end when the fan-in of each ψ_z is $q = 6$ and $q = 2$. Training, validation and test sets are composed of $20k$, $5k$, and $10k$ digits, respectively, taken from a (class-balanced) subset of the MNIST data. Only a small portion of the training set is labeled, and it is converted into cross-entropy-based supervision constraints ($\hat{\phi}_j$ in Eq. 1).

Table 1 reports both the evaluation of the performances in terms of accuracy for the task functions and some prominent examples of the extracted rules (in the case of 100 labeled examples). Predictions are considered correct only if they perfectly match the whole ground truth tuple of length 12. It is important to notice that the reported rules in all experiments are extracted from a single run of the stage-based optimization: the MI, in fact, is better maximized within this framework, leading to more informative rules. On the right-side of the table some samples which belong to the support X_z of the rules are also reported.

The improvements in terms of accuracy respect to the baseline are evident for both the stage-based and global-optimization. As expected, we observe stronger improvements with a smaller number of labeled examples per class, especially in the case of the stage-based optimization (15% better than the baseline with 1 labeled example). The FOL formulas in the first configuration of the ψ_z networks (a), nicely capture the mutual exclusivity of $Even$ and Odd of the digits ($Eight\ \dot{\vee}\ Zero$) and the negative correlation of some of the odd (even) digits with the $Even$ (Odd) class. The rules extracted from the second configuration (b), instead, gather the digits into two groups (Odd and Even) and attempt to provide a description of each group. However, they remark the importance of the fan-in reduction process (Sec. 3): as the fan-in increases, the readability of the rules decreases.

PASCAL Multi. PASCAL Part dataset is composed of 6103 images of objects and object parts which were split into 5492 training images, 305 validation images and 306 test images. They were rescaled to 128×128 pixels, and we extracted features using the ResNet50 network trained on ImageNet. Following the approach used in (Donadello, Serafini, and Garcez 2017), very specific parts (e.g. front-left-leg) were merged into unique labels (e.g. leg) leading to 62 classes, out of which 42 are related to parts. In order to evaluate the regularization effects of the learning of constraints when the number of labeled samples changes (the other training samples are unlabeled), we have repeated the analysis with 10, 50 labels per class and with the dataset completely labeled. Task functions have one hidden layer with 100 units. The ψ-networks include 50 output units, and we consider both the case of architectures with no-hidden layers (a) and with 1 hidden layer (20 hidden units) (b), respectively setting fan-in $q = 3$ and $q = 2$. We have tried to further increase ψ_z network deepness, but the regularization effects on the task function and the quality of the extracted rules did not improve.

Table 2 reports the macro F1 score for the task functions and examples of the best extracted rules (data samples are

[4]All experiments are reproducible by using the code available at https://github.com/gabrieleciravegna/Information-based-Constrained-Learning

Table 2: Macro F1 scores (±standard deviation) in PASCAL Multi dataset in the case of two configurations of the ψ network (a-b, see the main text for details), varying the amount of labeled examples per class.

# Labeled Examples	Baseline	Stage-based Optimization	Global Optimization
10	$54.0 \pm 0.3\%$	$\mathbf{56.8} \pm \mathbf{0.2\%}$	$55.8 \pm 0.7\%$
50	$60.5 \pm 0.2\%$	$\mathbf{62.0} \pm \mathbf{0.4\%}$	$61.3 \pm 0.5\%$
Whole dataset	$62.6 \pm 0.3\%$	$\mathbf{63.8} \pm \mathbf{0.2\%}$	$63.7 \pm 0.4\%$

Learned Rules

(a)	$\forall x \in X_{\psi_1},\ \neg Beak \wedge Hand \wedge Foot$	$\forall x \in X_{\psi_2},\ \neg Foot \wedge Dog$
	$\forall x \in X_{\psi_3},\ \neg Boat \wedge Handlebar$	$\forall x \in X_{\psi_4},\ \neg AirplaneBody \wedge Cow$
(b)	$\forall x \in X_{\psi_1},\ Beak \wedge \neg Ear \wedge \neg Nose$	$\forall x \in X_{\psi_2},\ (Cat \vee Dog) \wedge (\neg Foot \vee Paw)$
	$\forall x \in X_{\psi_3},\ TvMonitor \wedge (Screen \vee Table)$	$\forall x \in X_{\psi_4},\ Aeroplane \wedge (Engine \vee Stern)$

in Fig. 3). As in the previous experiment, they all come out from the stage-based optimization. We observe an evident improvement, in terms of F1, obtained by both optimization methods over the baseline.

Due to its complexity, this benchmark always benefits from the employment of the proposed approach, regardless of the number of supervision. If we evaluate the rules of configuration (a) and the first rule of configuration (b), they describe the incompatibility among certain objects or object parts in the same image. This type of reasoning is often used also by human to automatically exclude some classes while recognizing something. Let us consider the case of the first rule of (b): one would immediately rule out the possibility that an animal may have ears or nose, while he already recognized it has a beak. Rule numbers 2, 3 and 4 of configuration (b), instead, are related to the composition of object parts with respect to a certain main class, e.g. an aeroplane with its engines or its stern.

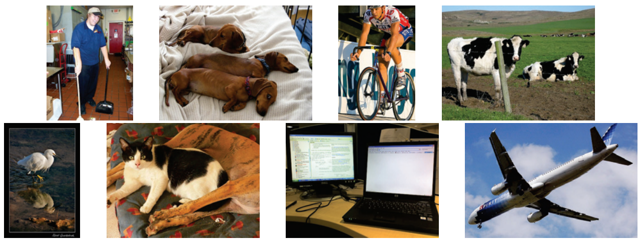


Figure 3: A sample from X_{ψ_z} for each rule of Table 2. First row, configuration (a). Second row, configuration (b).

Results Analysis

In order to further understand the regularization effects of the proposed approach, we have analysed the performances of the three configuration, Baseline, Stage-based Optimization and Global Optimization, under the same conditions, i.e. same hyperparameters, labeled data and network initialization. Fig. 4 reports the plots of two opposite scenarios: the first experiment with one supervised example per class and the second experiment when the whole training set is labeled. In both the cases we can observe an improvement

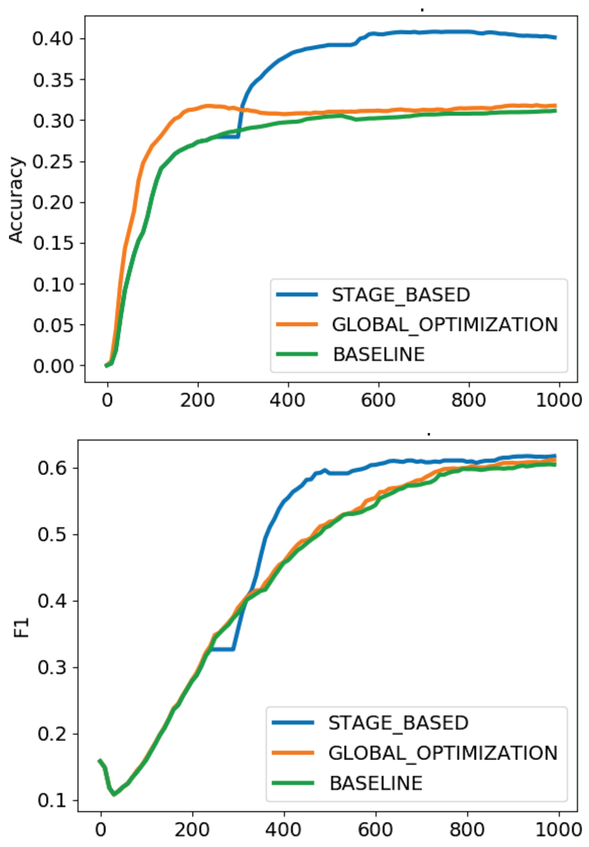


Figure 4: Performances on MNIST dataset with 1 labeled example per class (top plot) and on PASCAL Multi dataset with the training set completely labeled (bottom plot), with respect to the number of epochs.

of the performances over the baseline. The stage-based performance evolution is particularly interesting to inspect: after 200 epochs, the learning *of* constraints stage begins (the one that involves the MI), and for the following 50 epochs the accuracy (or F1) does not improve as the task functions are not modified. Nonetheless, in a few epochs the regularization effects imposed by the MI term in Eq. 6 boost the performances of the network overcoming those of the other two networks. This process is repeated every 250 epochs.

5 Conclusions and future work

Given a set of learnable task functions, we proposed a novel approach to learn constraints on such functions, which is based on information theoretic principles. Constraints are implemented by neural networks that, in turn, are trained to maximize the Mutual Information between the task functions and the rule space. At the end of the training step, the constraints (neural networks) are interpreted as Boolean FOL formulas and some promising results are discussed for different settings. In future work, we propose to analyse the effect of learning new constraints with respect to the robustness of the task functions, as in case of adversarial attacks.

References

Bach, S. H.; Broecheler, M.; Huang, B.; and Getoor, L. 2015. Hinge-loss markov random fields and probabilistic soft logic. *arXiv preprint arXiv:1505.04406*.

Bessiere, C.; Koriche, F.; Lazaar, N.; and O'Sullivan, B. 2017. Constraint acquisition. *Artificial Intelligence* 244:315–342.

Campigotto, P.; Battiti, R.; and Passerini, A. 2015. Learning modulo theories for preference elicitation in hybrid domains. *arXiv preprint arXiv:1508.04261*.

Castro, J. L., and Trillas, E. 1998. The logic of neural networks. *Mathware and Soft Computing* 5:23–37.

De Raedt, L.; Dries, A.; Guns, T.; and Bessiere, C. 2016. Learning constraint satisfaction problems: An ilp perspective. In *Data Mining and Constraint Programming*. Springer. 96–112.

De Raedt, L.; Passerini, A.; and Teso, S. 2018. Learning constraints from examples. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Di Nola, A.; Gerla, B.; and Leustean, I. 2013. Adding real coefficients to łukasiewicz logic: An application to neural networks. In *International Workshop on Fuzzy Logic and Applications*, 77–85. Springer.

Diligenti, M.; Gori, M.; and Sacca, C. 2017. Semantic-based regularization for learning and inference. *Artificial Intelligence* 244:143–165.

Donadello, I.; Serafini, L.; and Garcez, A. d. 2017. Logic tensor networks for semantic image interpretation. *arXiv preprint arXiv:1705.08968*.

Fu, L. 1991. Rule learning by searching on adapted nets. In *AAAI*, volume 91, 590–595.

Gnecco, G.; Gori, M.; Melacci, S.; and Sanguineti, M. 2015. Foundations of support constraint machines. *Neural computation* 27(2):388–480.

Huang, S. H., and Xing, H. 2002. Extract intelligible and concise fuzzy rules from neural networks. *Fuzzy Sets and Systems* 132(2):233–243.

Kasabov, N. K. 1996. Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems. *Fuzzy sets and Systems* 82(2):135–149.

Kolb, S.; Teso, S.; Passerini, A.; and De Raedt, L. 2018. Learning smt (lra) constraints using smt solvers. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence,{IJCAI-18}*, volume 104, 2067–2073. International Joint Conferences on Artificial Intelligence Organization.

Melacci, S., and Gori, M. 2012. Unsupervised learning by minimal entropy encoding. *IEEE transactions on neural networks and learning systems* 23(12):1849–1861.

Muggleton, S. 1991. Inductive logic programming. *New generation computing* 8(4):295–318.

Pawlak, T. P., and Krawiec, K. 2017. Automatic synthesis of constraints from examples using mixed integer linear programming. *European Journal of Operational Research* 261(3):1141–1157.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine learning* 62(1):107–136.

Sato, M., and Tsukimoto, H. 2001. Rule extraction from neural networks via decision tree induction. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 3, 1870–1875. IEEE.

Towell, G. G., and Shavlik, J. W. 1993. Extracting refined rules from knowledge-based neural networks. *Machine learning* 13(1):71–101.

Tsukimoto, H. 2000. Extracting rules from trained neural networks. *IEEE Transactions on Neural networks* 11(2):377–389.

Valiant, L. G. 1984. A theory of the learnable. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, 436–445. ACM.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, 2319–2328.

Zilke, J. R.; Mencía, E. L.; and Janssen, F. 2016. Deepred–rule extraction from deep neural networks. In *International Conference on Discovery Science*, 457–473. Springer.