Incremental Learning from Positive Examples

Grazia Bombini, Nicola Di Mauro, Floriana Esposito, and Stefano Ferilli

Università degli Studi di Bari, Dipartimento di Informatica, 70125 Bari, Italy
{gbombini,ndm,esposito,ferilli}@di.uniba.it

Abstract. Classical supervised learning techniques are generally based on an inductive mechanism able to generalise a model from a set of positive examples, assuring its consistency with respect to a set of negative examples. In case of learning from positive evidence only, the problem of over-generalisation comes into account. This paper proposes a general technique for incremental multi-class learning from positive examples only, which has been embedded in the learning system INTHELEX. The idea is to incrementally suppose the positive evidence for a class to be a negative evidence for all other classes until the environment explicitly declares the contrary.
An application of the proposed technique to the agent learning domain has been provided. The proposed framework has been used to simulate an agent learning and revising in an incremental way a logical model of a task by imitating skilled agents. In particular, demonstrations are incrementally received and used as training examples while the agent interacts in a stochastic environment. The experimental results prove the validity of the proposed approach on this application domain.

Key words: Inductive Logic Programming, Incremental Learning

1 Introduction

The goal of inductive Machine Learning systems is to discover a description of a target concept from a set of observations provided by the expert and possibly a background knowledge. Inductive learning can be cast as a search problem in the space of all possible hypotheses [1]. In the supervised setting a background knowledge and a set of positive and negative examples are exploited by the learning system in order to find a hypothesis explaining all positive examples and none of the negative ones. In particular, negative examples are used to bias the search in the space of all possible hypotheses.

However, in many cases it may be necessary to learn from positive examples only because no negative example is available, as in the case of a child learning to speak, or when an animal or an agent learns how to act by observing and/or by imitating other agents. In this scenario the learning system could produce an overly general hypothesis, such as "everything is true", due to the absence of contradicting evidence.

This paper proposes a general technique for incremental multi-class learning from positive examples only. The idea is to incrementally suppose the positive

evidence for the class c_i to be a negative evidence for all other classes c_j $(i \neq j)$, until the environment explicitly declares the opposite. In this paper, a first order logic description language to express possible hypotheses is used. First-order logic is a powerful symbolic representation language able to represents the intrinsic relations among the domain. It is also understandable by humans.

An application of the proposed technique will be provided, where an agent must learn and revise in an incremental way a logical representation of a task by imitating a skilled agent. In particular, the proposed technique has been used to learn a policy adopting a relational language to describe both the demonstrations and the policy. The relational description language makes the adoption of the background knowledge very simple. The agent actively interacts with the teacher by deciding which action to execute next and requesting demonstrations.

2 Related work

From an Inductive Logic Programming (ILP) [2] point of view, one of the most popular approaches to the problem of learning from positive examples only, is to exploit some form of clustering, adopting symbolic generalisations in the given representation language (as used by Kirsten and Wrobel [3]). Given some elements of the space of all possible observations, the learner tries to generalise over various subsets of the observed data yielding proper rules. Each subset, known as a "cluster", is made up of examples that are deemed near to each other. However, this is an unsupervised batch approach that does not fit the requirements of learning by imitation.

Other approaches to positive only learning include systems that adopt the Bayes theorem [4], like Progol [5]. In [4] the problem has been studied in a Bayesian framework where the distribution of the examples is assumed to be known. The solution provided is to generate random examples of the target concept by sampling the type range of each attribute, and then using these examples as negative. After selecting the positive instance to be generalised, the most specific clause within the language constraints that entails it is constructed. In order to find a concept description, a search in the hypothesis space of clauses that are more general than such as most specific clause is performed. In the induction phase, a measure to evaluate how well a clause explains all the examples is employed.

MERLIN 2.0 [6] uses Hidden Markov Models to facilitate learning from positive examples in situations where predicate invention is required, such as learning predicates that classify or generate sequences. MERLIN induces Hidden Markov Models from positive sequences only to determine when to invent new predicates. Due to the use of this induction technique, incremental learning is possible.

The following systems apply a positive-only learning approach to unlabelled data exploiting positive observations. In [7], Yu et al. proposed a technique based on Support Vector Machines (SVM), named PEBL, to classify Web pages given positive and unlabelled pages. This approach uses heuristics to identify unlabelled examples that are likely to be negative and applies a standard two-class

(positive and negative) learning method to these examples and the positive examples. Essentially their method consists of two steps: the first identifies a set of reliable negative documents from the unlabelled set (strong negative documents) i.e. those documents that do not contain any feature of the positive data, and the second builds a classifier using SVMs. The number of positive examples affects the performances of PEBL. Indeed, when the positive data is small, the results are often very poor.

In [8] Lee and Liu defined the problem of learning from a set of positive and unlabelled examples as a two-class learning problem with noisy negative examples. They heuristically identify some reliable negative examples in the unlabelled set using logistic regression. They assign weights to unlabeled examples and use weighted logistic regression to learn the classes of "positive" and "negative". The proportion of noise in the labeled examples can be approximated by the proportion of noise in a random sample from the overall set of examples. The unlabeled examples are interpreted as weighted negative examples.

3 Learning from positive examples only

The learning problem for ILP can be formally defined:

Given: A finite set of clauses $\mathcal{B}$ (*background knowledge*) and sets of clauses E^+ and E^- (positive and negative *examples*).

Find: A theory Σ (a finite set of clauses), such that $\Sigma \cup \mathcal{B}$ is *correct* with respect to E^+ and E^-, i.e.: a) $\Sigma \cup \mathcal{B}$ is *complete* with respect to E^+: $\Sigma \cup \mathcal{B} \models E^+$; and, b) $\Sigma \cup \mathcal{B}$ is *consistent* with respect to E^-: $\Sigma \cup \mathcal{B} \not\models E^-$.

Given the formula $\Sigma \cup \mathcal{B} \models E^+$, deriving E^+ from $\Sigma \cup \mathcal{B}$ is *deduction*, and deriving Σ from $\mathcal{B}$ and E^+ is *induction*. In the simplest model, $\mathcal{B}$ is supposed to be empty and the deductive inference rule $\models$ corresponds to *θ-subsumption* between clauses.

Common ILP approaches produce general clauses from positive examples and restrict their coverage by the help of negative examples. In domains where only positive examples are available, the learning systems may not be able to learn the concepts correctly because of the lack of restrictions induced by negative examples. Indeed, learning from positive examples only may cause over-generalisation that leads to inconsistent resulting hypotheses. In a multiple concept learning setting, given a positive example for a class, it is possible to exploit the other classes to assume negative examples. In particular, the given example is regarded as negative for the other classes.

Incremental learning is necessary when incomplete information is available at the time of initial theory generation. The learned theory is checked to be valid on any new available example. In case of failure, a revision process is activated on it in order to restore the completeness and the consistency properties. When a positive example which is inconsistent with the hypothesis biased drawn from previous negative examples is processed, all such negative examples are excluded from the learning process. When a candidate negative example is considered, the

Algorithm 1 IIL

```
 1: errors ← 0
 2: T ← ∅
 3: M ← ∅
 4: loop
 5:    (c, e) ← get a new positive example
 6:    N ← {(¬c_i, e)|c_i ∈ C \ {c}}
 7:    s ← classify(e,T)
 8:    if s ≠ c then
 9:       errors++
10:       for all m ∈ M do
11:          if m is negative for the class c with the same body as e then
12:             M ← M \ {m}
13:       generalize( T, c ← e, M)
14:    M ← M ∪ {c ← e}
15:    for all c_i ∈ N do
16:       if M does not contain the positive example c_i ← e then
17:          if s = c_i then
18:             errors++
19:             specialize( T, ¬c_i ← e, M)
20:          M ← M ∪ {c_i ← e}
```

system checks whether a previous positive example has already been processed for the same class. Hence, it is necessary that the system is capable of taking into account the examples in an incremental way and learn simultaneously several concepts, possibly related to each other.

Algorithm 1 reports the tuning procedure to refine the current theory. M represents the set of all positive and negative examples already processed, E is the current example to be examined, and T is the theory learned from the examples in M which are incrementally provided by an expert. It starts with an empty theory and an empty historical memory. Whenever a new example is taken into account, it is also stored in the historical memory. The training examples are exploited by the system to modify incorrect hypotheses according to a data-driven strategy.

Each evidence e belongs to a specific learning class c. When a positive example $E = \{c, e\}$ is provided to the learning system, it is considered as a negative example for all the other classes $c_i \in C$ with $c_i \neq c$ that must be learned. N represents the set of negative examples assumed. For each positive example, the system verifies the soundness of T trying to classify it using the learned model. If there is a misclassification, a theory revision is needed and all the negative examples for class c having the same description of e are excluded from historical memory.

A generalisation process on the current theory produces a revised theory obtained in one of the following ways:

1. generalise one of the definitions pertaining to the theory that relate to the concept of the example, by removing conditions. This must be done by ensuring that the revised theory covers the new example and is consistent with the all negative examples previous examined;
2. add a new definition to the theory to explain the example;
3. add the example as a positive exception.

All the candidate negative examples $(\neg c_j, e) \in N$ for which a positive example $c_j \leftarrow e$ has already been processed are ignored. Indeed, since that evidence was already taken into account as positive in the learning process, it can not be assumed as a negative example for c_j.

When a negative example is covered, a specialization process outputs a revised theory by performing one of the following actions:

1. add positive conditions to each definition that explains the example. These conditions must be able to characterize all positive examples already considered and exclude the current negative example;
2. add a negative condition to the faulty definitions to differentiate the current negative example from positive examples previously considered;
3. add the negative example as an negative exception.

In case of concepts that have many positive examples and few negative ones that distinguish them, the system learns different definitions. The negative examples are used for specialise the definitions that apply to specific cases. At the end of the learning process, T represents the learned model.

In order to explain the general procedure previously described, we present two simple examples belonging the blocks world domain. This domain consists of 4 blocks (a, b, c and d), where blocks can be on the floor (denoted by f) or can be stacked on each other. Predicate $on(X, Y)$ denotes that block X is on block Y, and that X and Y belong to the same stack. In this domain, the available actions are of the kind $move(X, Y)$, with $X \in a, b, c, d$ and $Y \in a, b, c, d, f$, $X \neq Y$, to be interpreted as the action to put X on Y. Predicate $block(X)$ means that X represents a block, $clear(X)$ means that X is clear, $floor(X)$ means that X represents the floor. Predicate $goal\_on(X, Y)$ means that the goal is achieve when block X is on block Y.

The following set of literals represents a possible state in this domain:
{*goal_on(a, b), clear(c), on(c, a), on(a, f), on(d, b), clear(d), on(b, f), block(a), block(b), block(c), block(d), floor(f)*}

In this state, it is correct to move both block c and block d on the floor to achieve the goal. If the positive example $move(c, f)$ with the previous observation is presented to the system, it adds the following definition to the theory to explain the example: *move(A,B) :- clear(A), block(A), floor(B), on(D, B), block(D), goal_on(D,E), block(E), on(F,E), clear(F), block(F), on(E,B), on(A,D).*

Then the system assumes *not(move(c,d))*, *not(move(d,c))*, *not(move(d,f))* as negative examples with the same evidence as the positive example. These

examples are not explained by the currently learned theory and thus is not necessary to specialise it.

Further examples concerning the same state may be presented to the system. In particular, a possible positive example is $move(d, f)$ with the evidence previously considered. In this case, the system assumes *not(move(c,d))*, *not(move(d,c))*, *not(move(c,f))* as negative examples.

When the system takes into account a positive example, is necessary to check whether a previous negative example (in this case *not(move(d,f))*) was previously processed and stored in the historical memory, in which case it must be removed.

Then the positive example is processed, and in this case a generalisation process is necessary on the learned clause. The negative examples *not(move(c,d))*, *not(move(d,c))* are not covered by the theory previously generalised, and therefore there is no need of further processing to revise the learned theory.

When the system takes into account the negative example $not(move(c, f))$, in the historical memory the corresponding positive example with the same evidence is found, and then the negative example is not processed.

The only clause learned from these few examples results: *move(A, B) :- clear(A), block(A), floor(B), on(D, B), block(D), on(E, B), block(E), on(F, E), clear(F), block(F), on(A, D).*

4 INTHELEX

The proposed algorithm , was integrated into INTHELEX (INcremental THeory Learner from EXample) [9], an incremental learning system for the induction of first-order logic theories from positive and negative examples. It can learn simultaneously several concepts, possibly related to each other. The approach used by this system may be defined as "multistrategic" since it combines different forms of reasoning in the same symbolic paradigm. It learns theories in form of sets of Datalog clauses, interpreted according to the Object Identity (OI for short) [10] assumption. Under the OI assumption, within a clause, terms (even variables) denoted with different symbols must be distinct.

INTHELEX is a fully and inherently incremental learning system. The learning phase can start by taking as input a previously generated version of the theory or an empty theory (from the first available example). When the theory is incorrect wrt an example, this is rejected and a process of theory revision starts. Algorithm 2 reports the general tuning procedure used in INTHELEX for refining the theory to be learned. INTHELEX uses two inductive refinement operators, one for generalising definitions that reject positive examples (completeness), and the other for specialising definitions that explain negative examples (consistency). In order to perform its task, the system exploits a previous theory (if any) and a *historical memory* of all the past (positive and negative) examples that led to the current theory.

The system can be provided with a background knowledge (i.e. some partial concept definitions know to be correct about the domain and hence not modifiable) in the same format as a theory rules.

Algorithm 2 tuning(E,T,M)

Input: E: example; T: theory; M: historical memory;
1: Add E to M
2: **if** E is a positive example not covered from T **then**
3: generalize(T, E, M)
4: **else**
5: **if** E is a negative example covered by T **then**
6: specialize(T, E, M)

5 Sample application: Learning from demonstration

Learning from demonstration or *learning by imitation* [11, 12] represents a promising approach in the area of human-robot interaction. Recent research in other fields considers the imitative learning as an essential part of human development [13]. Indeed, humans and animals use imitation as a mechanism for acquiring knowledge. Imitation-based learning [14] and learning by demonstration [15] are exploited to enable an unskilled agent, the *observer*, to learn tasks by simply observing performances of a skilled agent, the *teacher* [16, 17]. This approach can be viewed as a collaborative learning based on the interaction between the human and the agent. The agent gathers information about the task in the form of perceptual inputs and action results, and estimates the latent control policy of the demonstrator. The estimated policy can then be used to control the agent's autonomous behaviour. As reported in [18], this method can reduce learning time when compared to classical exploration-based methods such as reinforcement learning (RL) [17].

Learning from demonstration is strongly related to supervised learning where the goal is to learn a policy given a fixed set of labelled data [19]. From this point of view it is possible to collect the interactions between the teacher and the observer and to use them as learning examples. Furthermore, data should be gathered in an incremental way by minimising the number of required labelled data useful to learn the given policy.

In the scenario of an agent acting in a stochastic world, the *correct actions* of the agent may be considered as positive examples in a supervised learning task.

Here, we assume that an agent A aims at learning how to act in an environment by imitating an *expert agent* E. The environment in witch the agent acts is defined by a finite set of states S. For each state $s \in S$, the agent has available a finite set of actions $A(s) \subseteq \mathcal{A}$ which cause stochastic state transition. In particular, an action $a \in A(s)$ causes a transition to state s'_a when executed in state s, where $\mathcal{A}$ is the set of all primitive actions. Given a goal, each training example $e = \{a, o\}$ belongs to a specific learning class based on the action a. For each class a theory must be learned in order to be able to predict the corresponding action on unseen observations.

Hence, each action taken from E in a given state may be considered as a positive example. Given an action-state pair (a, s) and a set $A(s)$ of possible actions that can be taken in state s, all other actions in $A(s) \setminus a$ are assumed to

be negative examples until the expert agent actually takes any of them in the same state s. Actions represent the target concepts, and the state represent the evidence. At the end of the learning process, the learned theory T represents the optimal policy.

The agent is assumed to observe a demonstrator that performs the correct sequence of actions useful to reach a given goal by starting from an initial state of the environment. During each training sequence, the agent records the observation about the environment and the corresponding action performed by the demonstrator. An observation $o \in S$ is represented as a set of ground Datalog literals. Each training example, $e = \{a, o\}$, consists of an action $a \in \mathcal{A}$ selected by the demonstrator and an observation $o \in S$. Obviously, we assume that the demonstrator uses a good policy π to achieve the goal. Hence, the aim of the agent is to learn such as hidden policy $\pi : S \rightarrow \mathcal{A}$ mapping states (observations) to actions.

In case of imitative learning, given a state s, if the teacher takes the action $a_i \in A(s)$, then the observer can assume that a_i is a positive example and all other actions $a_j \in A(s)$ $1 \leq j \neq i \leq |A|$ are negative ones. A negative example a_j for the state s is considered reliable until the demonstrator performs a_j in s. Furthermore, the process of imitative learning is naturally modeled by an agent able to modify its theory in an incremental way, where each new incoming example may give rise a theory revision process. We represent the policy as a set of logical clauses where the *head* of the clause represents the action while the *body* represents the state. In particular, a clause represents an action that may be performed in a given state.

The proposed learning framework has been applied to a pursuit domain, generally used in the field of agent learning. The experiment regards the problem of learning a policy in a domain where a predator should capture a prey. This stochastic environment consists of a 4x4 grid surrounded by a wall, with a predator and a prey inside. We assume that the predator has caught the prey if the prey lies on the same square as the predator at the end of its move. The prey moves at random, while the predator follows a *good user-defined* strategy in order to capture the prey. Both the predator and the prey can move in four directions (north, east, south and west). The action of an agent consists in moving to the square immediately adjacent in the selected direction. In case the target square is a wall, the agent stays in the same square. The two agents move alternate turns.

An observation is made up of the agent's perception about the squares surrounding it in the four directions and the square under it. The state of each square may be *empty*, *wall* or *agent*. Starting from an initial state, once captured the prey the sequence of observations does not restart by placing agents in random positions, but it continues from the positions of catch. For example, a predator agent having a wall to the west and the prey to the east, the observation is represented by the following set of literals: {*observation(a, d1, e), observation(a, d2, e), observation(a, d3, p), observation(a, d4, w), under(a, e), direction(d1, named1), direction(d2, named2), direction(d3, named3), direction(d4, named4),*

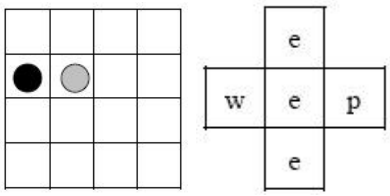

Fig. 1. A sample predator prey domain. The black circle represents the predator and the gray circle represents the prey. The figure on the right represents the predator agent percept.

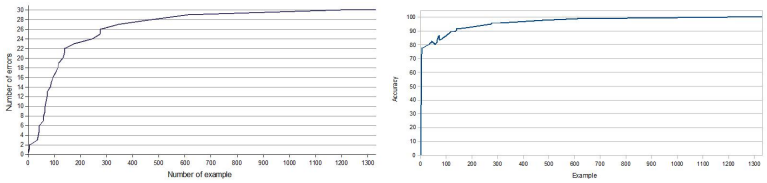

Fig. 2. Errors during the learning phase. The figure on the right represents the prediction accuracy (%) over the entire historical memory for each theory revision.

north(named1), south(named2), east(named3), west(named4), prey(p), predator(a), wall(w), empty(e)} where a stays for predator agent, p for prey agent, e for empty, and w for wall.

After fixing the strategy to capture the prey, we simulated a scenario in which a predator instructs another agent to capture the prey with a minimum number of steps. Hence, given an observation, the action taken from the predator represents a positive training example, while all other possible actions are supposed to be negative examples.

We have generated 10 sequences of observations. Each sequence, containing traces of prey's captures, is made up of 322,5 positive and 967,5 negative observation-action pairs on average.

A first experiment has been performed on the whole dataset without providing the system with any background knowledge. Over all the 10 sequences the system learned a theory (policy) made up of 9,4 clauses, obtained by 14,8 generalisations and 2,2 specialisations (26,4 errors) on average. Each clause is composed of 13.5 literals on average. The system needs 0.325 seconds per example to learn the correct policy.

In order to evaluate the behaviour of the learning process, we generated a sequence made up of 1332 observation-action pairs.

Figure 2, reports on the left, the number of errors (a generalisation or a specialisation request) the agent made during the learning phase. As we can see, the number of errors grows until the system learns the correct policy (i.e., the learned classification theory). Figure 2, on the right, reports the evolution

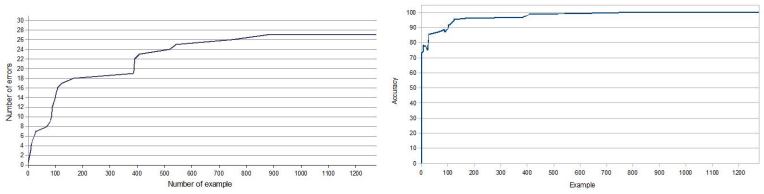

Fig. 3. On the left: errors during the learning phase. On the right: prediction accuracy (%) over the entire historical memory at each revision theory with the use background knowledge.

of prediction accuracy of the policy, learnt by the agent during the imitation process.

An example of learned rule is {*move(A, B) :- predator(A), direction(B, D), direction(F, H), observation(A, I, J), observation(A, B, E), observation(A, F, G), south(H), north(D), wall(E), prey(G), empty(J), under(A, J).*}

where a predator A moves toward north when at south there is the prey and at north there is a wall.

Another experiment has been performed with the system provided with a background knowledge containing several rules defining the concept of "next direction". An example is {*next(A,B):- north(A), east(B).*}. This rule denotes that north direction followed by the east direction. The learned theory abstracts the concept of direction. An example of learned rule is

{*move(A, B) :- predator(A), direction(B, D), direction(G, F), next(D, H), next(H, F), next(F, E), next(E, D), observation(A, B, I), observation(A, G, K), wall(I), under(A, J), empty(J), prey(K).* }

where predator A moves in direction D when it observes that in D there is a wall and opposite there is the prey K at any edge of the grid (D may be any of the four direction). This action applies to four different states.

On the same 10 sequences of the previous experiment, the system learned a theory made up of 6,4 clauses, obtained by 12,1 generalisations and 3,5 specialisations (22 errors) on average. Each clause is composed of 15.3 literals on average. The system needs 0.337 seconds per example to learn the correct policy. In order to evaluate the behavior of the learning process with the use of a background knowledge, we generated a sequence made up of 1276 observation-action pairs.

Figure 3 reports the number of errors (a generalization or a specialization request) the agent made during the learning phase and prediction accuracy of the policy.

It is interesting to note that the theory learned with the help of a background knowledge does not exploit specific directions, but rather the relationships between them. This allows an abstraction on states. Indeed, the policy learned tours out to be more compact (a lower number of clauses on average compared with the first experiment), where each clause is applicable to a subset of states.

6 Conclusion

In domains where only positive example are available, a classical supervised learning system may not be able to learn the concepts correctly because the lack restrictions induced by negative examples. Indeed, negative examples are used to bias the search in the space of all possible hypotheses. In this case, the problem of over-generalisation comes into account.

A general technique for incremental multi-class learning from positive examples only has been presented. The problem of over-generalisation has been avoided assuming as negative examples the positive example for the other classes, until the environment does not explicitly declare the contrary.

The proposed technique has been implemented and tested on a dataset belonging to the field of agent learning from demonstration. The proposed framework allows to quickly, accurately and incrementally train an unskilled agent to imitate a (human or artificial) demonstrator. The results confirm the validity of the technique.

Acknowledgements

This work is partially funded by Italian Ministry of University and Scientific Research FAR Project MBLab "The Molecular Biodiversity Laboratory".

References

1. Mitchell, T.M.: Generalization as search. Artif. Intell. **18**(2) (1982) 203–226
2. Lavrac, N., Dzeroski, S.: Inductive Logic Programming: Techniques and Applications. Ellis Horwood, New York (1994)
3. Kirsten, M., Wrobel, S.: Relational distance-based clustering. In: ILP. (1998) 261–270
4. Muggleton, S.: Learning from positive data. In: Inductive Logic Programming Workshop. (1996) 358–376
5. Muggleton, S.: Inverse entailment and progol. New Generation Comput. **13**(3&4) (1995) 245–286
6. Boström, H.: Predicate invention and learning from positive examples only. In: ECML. (1998) 226–237
7. Yu, H., Han, J., Chang, K.C.C.: Pebl: positive example based learning for web page classification using svm. In: KDD. (2002) 239–248
8. Lee, W.S., Liu, B.: Learning with positive and unlabeled examples using weighted logistic regression. In: Proceedings of the Twentieth International Conference on Machine Learning (ICML). (2003)
9. Esposito, F., Ferilli, S., Fanizzi, N., Basile, T., Di Mauro, N.: Incremental learning and concept drift in inthelex. Intelligent Data Analysis Journal, Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift **8**(3) (2004) 213–237
10. Semeraro, G., Esposito, F., Malerba, D.: Ideal refinement of datalog programs. In Proietti, M., ed.: Logic Program Synthesis and Transformation. Volume 1048 of LNCS., Springer (1996) 120–136

11. Billard, A., Siegwart, R.: Robot learning from demonstration. Robotics and Autonomous Systems **47**(2-3) (2004) 65–67
12. Schaal, S., Ijspeert, A., Billard, A.: Computational approaches to motor learning by imitation. Philosophical Transactions: Biological Sciences **358**(1431) (2003) 537–547
13. Meltzoff, A.: The "like me" framework for recognizing and becoming an intentional agent. Acta Psychologica **124**(1) (2007) 26–43
14. Schaal, S.: Is imitation learning the route to humanoid robots? Trends in cognitive sciences **3**(6) (1999) 233–242
15. Nicolescu, M., Mataric, M.: Natural methods for robot task learning: instructive demonstrations, generalization and practice. In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems (AAMAS03), ACM (2003) 241–248
16. Atkeson, C., Schaal, S.: Robot learning from demonstration. In Fisher, D., ed.: Proceedings of the 14th International Conference on Machine Learning (ICML). (1997) 12–20
17. Smart, W., Kaelbling, L.: Effective reinforcement learning for mobile robots. In: IEEE International Conference on Robotics and Automation (ICRA). Volume 4. (2002) 3404–3410
18. Chernova, S., Veloso, M.: Confidence-based policy learning from demonstration using gaussian mixture models. In: AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, New York, NY, USA, ACM (2007) 1–8
19. Bentivegna, D., Atkeson, C., Cheng, G.: Learning from observation and practice using primitives. In: AAAI Fall Symposium Series, 'Symposium on Real-life Reinforcement Learning'. (2004)