Bias Learning, Knowledge Sharing

Joumana Ghosn and Yoshua Bengio

***Abstract*—Biasing properly the hypothesis space of a learner has been shown to improve generalization performance. Methods for achieving this goal have been proposed, that range from designing and introducing a bias into a learner to automatically learning the bias. Multitask learning methods fall into the latter category. When several related tasks derived from the same domain are available, these methods use the domain-related knowledge coded in the training examples of all the tasks as a source of bias. We extend some of the ideas presented in this field and describe a new approach that identifies a family of hypotheses, represented by a manifold in hypothesis space, that embodies domain-related knowledge. This family is learned using training examples sampled from a group of related tasks. Learning models trained on these tasks are only allowed to select hypotheses that belong to the family. We show that the new approach encompasses a large variety of families which can be learned. A statistical analysis on a class of related tasks is performed that shows significantly improved performances when using this approach.**

***Index Terms*—Bias learning, knowledge sharing, knowledge transfer, learning to learn, multitask learning.**

I. Introduction

MODEL-FREE or nonparametric inference has been shown to lead to poor generalization performance when attempting to learn complex problems using small training sets [9]. Improving the generalization performance can be achieved by using prior knowledge defining known properties of the problem that must be learned. This knowledge can be used to define a preference for a certain class of solutions in the space of all possible solutions or hypotheses that can be selected by a learner. Many techniques have been developed to incorporate prior knowledge in a learning model or in the learning process. For example, *artificial examples*, representing known properties about a problem, can be generated and added to the training set, in order to emphasize the need to select a solution that incorporates the properties embedded in the additional examples [1]. Penalty terms can be designed that penalize, during the learning process, the selection of solutions that do not represent known properties of the problem at hand [23]. And model-based learners can be created that satisfy constraints or characteristics specific to a problem (such as convolutional neural networks which were designed to tackle pattern recognition problems) [14], [15]. All these techniques require on the one hand, the availability of experts who can provide accurate and detailed knowledge, and on the other hand, the possibility of "translating" the provided knowledge into a form that can be used by learning models. Despite many successes, designing an appropriate bias (i.e., defining a preference for a particular class of solutions) remains a complex problem.

The training examples of a problem and the prior knowledge provided by experts include or encompass two distinct but complementary categories of knowledge: on the one hand, they define properties that are specific to the problem being learned, and on the other hand, they define properties that are more general and belong to the domain or environment in which the problem evolves (i.e., properties that are common to all the problems that evolve in the same environment). While the first group of properties can only be used to learn a particular problem, the second group can be used to learn any problem evolving in the same domain or environment. Hence, the greater usefulness of the second group of properties.

While prior knowledge defining properties of an environment can be used to learn any problem evolving in the environment (by using some of the methods aforementioned), the training examples of a particular problem could not until recently be used to learn other problems evolving in the same environment. This was due to the lack of methods capable of analyzing the training examples of a problem and distinguishing between the properties specific to the corresponding problem and the properties specific to the environment in which the problem evolves. *Multitask learning* was developed to overcome this shortcoming. The purpose of this discipline is to use the training examples of a group of related tasks evolving in the same environment, to define, in the space of all possible solutions, a class of solutions that represent or embody the domain or environment-related properties extracted from the training examples of all the related tasks. It achieves this aim by searching in the hypothesis space (or the space of all possible solutions) for a class of solutions that is suited for learning all the tasks in the environment. Multitask learning therefore attempts to *learn a bias* by trying to identify the properties common to a set of related problems.

Different approaches to multitask learning have been proposed over the last decade. In Section II, we review previous work in this field. We then describe in Section III a new multitask learning method that is simple and efficient. As all other multitask learning methods, this method assumes that all the problems or tasks being considered for learning belong to the same domain or environment. But contrary to several multitask learning methods that can only consider specific classes of solutions, the new method can be used to define a wide variety of classes. Also, the new method can be applied to a large variety of learning models while some methods can only be applied to neural networks. The basic principle of the new method is simple: the class of solutions embedding the properties of an

Manuscript received November 30, 2000; revised March 20, 2002 and December 12, 2002.

The authors are with the Department of Informatique et Recherche Opérationnelle, Université de Montréal, Montréal, QC H3C 3J7, Canada (e-mail: ghosn@iro.umontreal.ca; bengioy@iro.umontreal.ca).

Digital Object Identifier 10.1109/TNN.2003.810608

environment is represented by a manifold or a mixture of manifolds in the hypothesis space. The manifold can be affine or not. Its dimensionality can vary. We will argue when presenting this concept that the smoothness and the dimensionality of the manifold define the capacity of the class of solutions. The position of the manifold in the hypothesis space is chosen based on an analysis of the training examples of a group of related problems or tasks. Learning a particular problem amounts to searching the area defined by the manifold to select a solution representative of the specific properties of the problem. This solution is represented by a point on the manifold. The results of experiments using this new method as well as three other multitask learning methods are presented in Section IV. The generalization performances of these methods are compared to the performances of single-task learning whereby each task or problem in a group of problems is learned separately and no class of solutions is defined in the hypothesis space (in that case, a learning model is free to explore the entire hypothesis space instead of being restricted to a confined area). The experiments presented in Section IV have been designed to illustrate the impact of several factors on the generalization performance of multitask learning. A statistical analysis shows that the new multitask learning method leads in all the learning contexts explored in the experiments to significant improvements in the generalization performance when compared to single-task learning. The statistical analysis also shows that learning a bias using the new multitask learning method leads to generalization performances that are either better or at least comparable to the performances obtained by other multitask learning techniques. We conclude Section IV by describing tests that evaluate the "quality" of a bias or of a class of solutions learned by multitask learning techniques. This evaluation can be accomplished by using the learned bias to learning novel tasks derived from the same environment. An analysis of the experimental results shows that when learning a new task, the new method outperforms single-task learning in most learning contexts and that its performance is either better or is comparable to another multitask learning technique. In Section V, we discuss the notion of task similarity or task relatedness and we conclude in Section VI.

II. Multitask Learning

Several multitask learning methods have been developed over the last few years. Although most of them are concerned with improving the generalization performance of learning models trained on related tasks, some methods have different objectives. Indeed, some methods are concerned with reducing the training time of learning models, while other methods aim at guiding the decision-making process encountered when learning a task.

Improving the generalization performance is generally performed when a group of related tasks evolving in the same environment is available. The training examples of all the tasks are used to define, in the hypothesis space, a class of hypotheses that is suitable for learning all the tasks in the group (or in general, for learning all the tasks in the environment). The internal representation learning method [3], [5], [10], [12], [13], [24], [25] improves the generalization performance of neural networks trained to learn related tasks by simultaneously training all the networks and forcing them to share their first layers (i.e., the weights of the first layers are the same across all the networks while the weights of the last layers are different across the networks). The weights of the shared layers are updated using training examples sampled from all the tasks. And the weights of the nonshared layers of each network are updated using only the training examples of the task being learned by the network. This method, which has been successfully applied to neural networks, can only define a particular class of hypotheses or solutions. Indeed, as shown in Fig. 1, this method can only learn environments or families which can be represented, in the hypothesis space, by an affine and axis-aligned manifold. It should be noted that learning a family consists in identifying a position of an affine and axis-aligned manifold that is suitable for learning the tasks being used to define the family [e.g., in Fig. 1(b), learning a family S corresponds to finding a position $w^*_{.1}$ such that the corresponding manifold contains solutions for the tasks that evolve in the environment represented by the family]. Once a family of solutions has been learned, this family can be used to learn new tasks that evolve in the environment represented by the family, and that might become available in the future: to learn a new task evolving in the same environment, a neural network is trained. The weights of the first layers of the network are fixed and set to those of the shared layers when the family was learned. Only the weights of the last layers are modified to adapt to the training signals of the new task (i.e., in Fig. 1, learning a new task is achieved by searching for a solution that belongs to the family of solutions S. The position of the family S is not modified when learning a new task).

In the family discovery method [19], a parameterized family of models is built. The family is represented in hypothesis space by an affine manifold or a mixture of affine manifolds. And the class of solutions is represented by the neighborhood of the family (i.e., a solution to a problem has to be chosen as close as possible to the family). The dimensionality of the family depends on the number of related tasks being learned. Indeed, the manifolds are defined by the top eigenvectors obtained in a principal component analysis of the parameters (or solutions) of a group of learning models trained on a group of related tasks. A variant of the expectation–maximization algorithm is used to refit the parameters of the learners in the context of the family, and the family in the context of the parameters. As with the internal representation learning method, the manifolds learned using the family discovery technique can be used to learn novel tasks that arise in the future: the position of the family in the hypothesis space is kept fixed and a learner trained on a new task is forced to select a solution close to the family. Examples of families are given in Fig. 2.

Other methods for improving the generalization performance have been developed to learn weighted distance metrics [26] (for classification problems) and to learn symbolic rules [8], [20].

Reducing the training time of a learning model is in general achieved when a task or a problem that needs to be learned is similar or related to a task that has previously been learned. The solution of the previously learned task can be used to define the "initial state" of the learning process for the new task [7], [17],

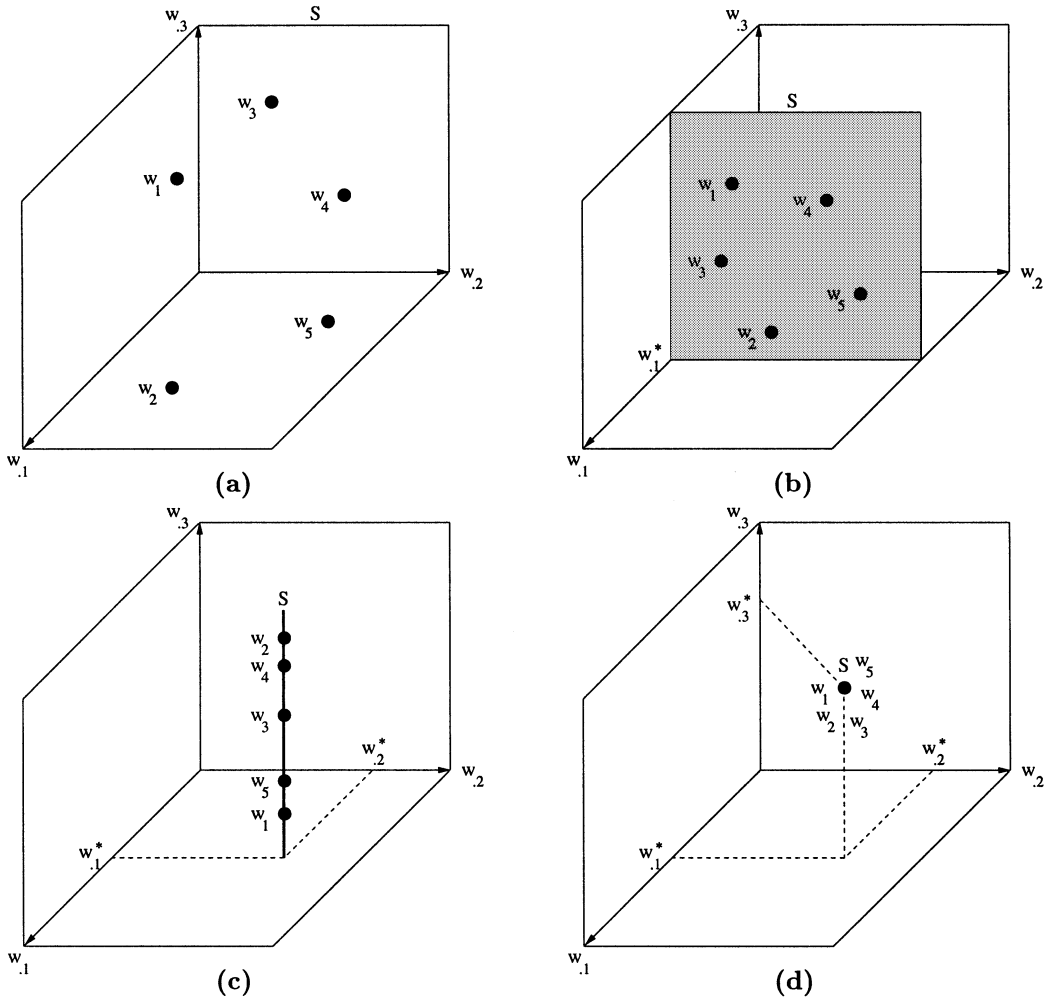


Fig. 1. Evolution of the class of solutions or hypotheses as a function of the number of shared layers in the internal representation learning method: the hypothesis space corresponds to the space of all possible parameters or weights of a neural network. Weight space is formed here of 3 axes, each representing the set of weights of a layer in a neural network containing two hidden layers and an output layer: $w_{.1}$, $w_{.2}$, and $w_{.3}$ represent the weights of the first hidden layer, the second hidden layer, and the output layer, respectively. The points $(w_1, \ldots, w_5)$ represent the solutions chosen for five tasks evolving in the same environment. The class of solutions is represented by S in the figures. The figures show the evolution of this class when (a) no layer is shared between the five networks trained on the five available tasks; (b) the first hidden layer is shared; (c) both hidden layers are shared; and (d) both hidden layers and the output layer are shared.

[21], [22]. For example, in [21], the parameters of a neural network trained on an old task are used to initialize the parameters of another neural network that needs to be trained on a new related task.

Guiding the decision-making process when learning a task consists in benefiting from previously acquired experience to make some choices concerning the learning process of new tasks. For example, in [4], a method is presented to choose, among several decision tree pruning techniques, one technique that is suited for a particular problem. Instead of trying all available pruning techniques (which can be a time-consuming process) to prune a newly built decision tree, this method suggests that the choice of the pruning technique should be based on the generalization performances of the different pruning techniques when these techniques were used to prune a previously built decision tree for a related task. The technique that led to the best generalization performance on the previously built decision tree should be used to prune the newly built decision tree.

III. Bias Acquisition Through Hypothesis Space Restriction

A. Manifold Learning: General Approach

Domain-related knowledge defines properties that learning models, trained on tasks derived from the corresponding domain, should comply with. These properties can be used to identify, in the hypothesis space of a learner, a family or a class of hypotheses that implement or verify them. Given that family, a learner should only explore hypotheses that belong to it.

When the domain-related knowledge is implicitly coded in the training examples of several tasks derived from the same domain, this knowledge needs to be extracted in order to define the family that will be used to guide the learning process. The extraction of the knowledge and the definition of the family can be performed by analyzing the training examples of all the tasks to determine or identify the properties shared by all the tasks. These properties can be used to learn a restricted class of hypotheses.

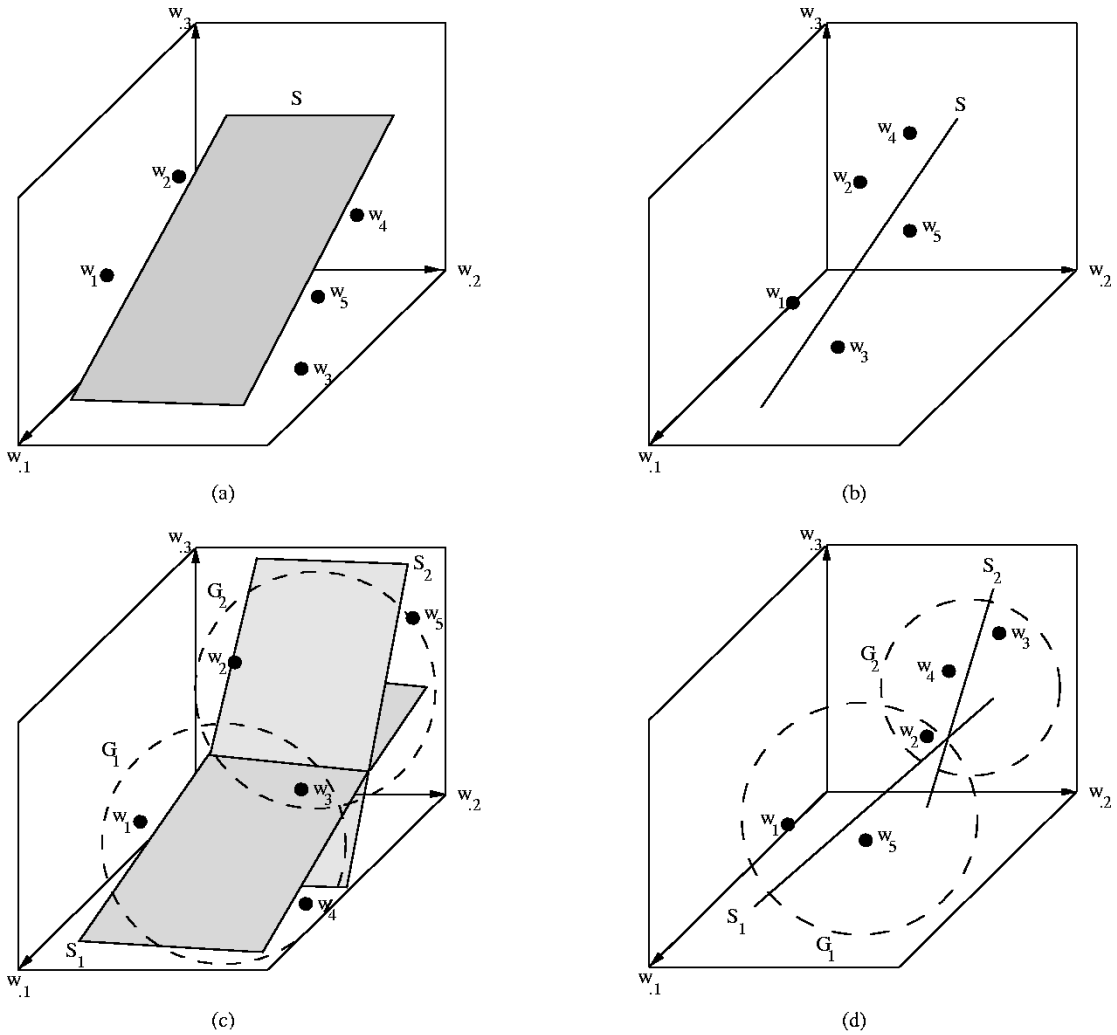


Fig. 2. Examples of families learned by the family discovery method: (a) two-dimensional affine family S, (b) one-dimensional affine family S, (c) nonaffine family represented by a mixture S of two affine manifolds (S_1, S_2), the manifolds being two-dimensional and having Gaussian influence functions (G_1, G_2) (representing the contribution of the manifolds to the mixture), and (d) nonaffine family represented by a mixture S of two affine manifolds (S_1, S_2), the manifolds being one-dimensional and having Gaussian influence functions (G_1, G_2). The class of solutions corresponds to the neighborhood of the family S. The size of the neighborhood increases with the dimensionality of the family. And the "refinement" or "quality" of the family increases with the number of manifolds in a mixture. The points ($w_1, \ldots, w_5$) represent the solutions chosen for five tasks evolving in the environment represented by the family.

A family of hypotheses can be represented in the hypothesis space by a manifold or a mixture of manifolds on which lie all the hypotheses that belong to the family. In this context, learning a task or a problem that belongs to the domain represented by the family requires the selection of a point on the manifold or on the mixture of manifolds. The chosen point must correspond to a hypothesis or a solution that embeds the problem-specific properties implicit in the training examples of the corresponding problem. The form of the family depends on the domain or environment. Different environments will require the use of different families. Different environments will therefore require the use of different manifold surfaces.

1) Family Definition: Let W represent the hypothesis space of a learning model (e.g., for a neural network containing M parameters, $W = \mathbb{R}^M$). When learning a particular task T_n, the learning model needs to choose a point w_n in this space. If this point has to be selected on a surface defining a family, then w_n should be parameterized as follows:

$$w_n = g(\varphi, \alpha_n) \tag{1}$$

where g represents the type or the form of the surface, φ defines the location of the surface in the hypothesis space, and α_n represents the position of the task-specific point w_n on the surface.

Learning the family requires choosing g and estimating φ, and learning a task T_n requires estimating α_n (which can be used to generate the corresponding value of w_n according to (1)).

When the family is a mixture of manifolds, it is defined as follows:

$$w_n = g(\varphi, \alpha_n) = \sum_{v=1}^{V} p_{n,v} w_{n,v} = \sum_{v=1}^{V} p_{n,v} g_v(\varphi_v, \alpha_{n,v}) \tag{2}$$

where V is the number of manifolds, $p_{n,v}$ is the contribution of the vth manifold to the generation of the point w_n, g_v is the form of the vth manifold, φ_v is the location of the vth manifold in hypothesis space, and $\alpha_{n,v}$ is the position of a point $w_{n,v}$ on the vth manifold. The contribution $p_{n,v}$ of the vth manifold should be positive and the sum of the contributions of all manifolds for a

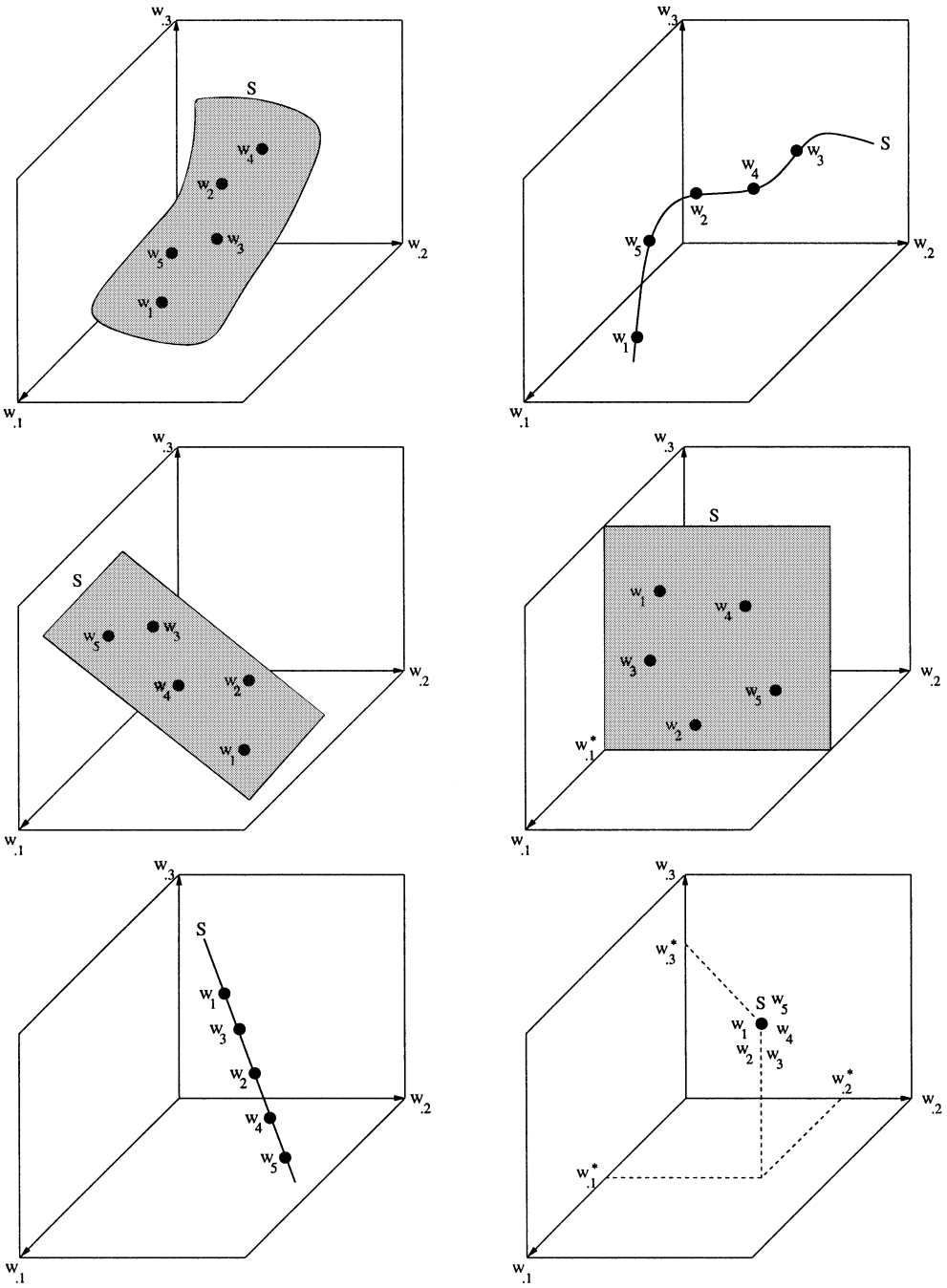


Fig. 3. Examples of families learned by the manifold learning method using the model defined in (1). A family S is represented by one manifold which can be affine or nonaffine. The dimensionality and the position of the manifold in the hypothesis space depend on the environment being modeled. The points $(w_1, \ldots, w_5)$ represent the solutions chosen for five tasks evolving in the environment defined by the family. These points are chosen on the surface defining the family.

particular point w_n should be equal to one. These two conditions can be met by defining $p_{n,v}$ as follows:

$$p_{n,v} = \frac{\exp(\xi_{n,v})}{\sum_{i=1}^{V} \exp(\xi_{n,i})}. \tag{3}$$

It should be noted that $\varphi = \{\varphi_v\}$ and $\alpha_n = (\{\alpha_{n,v}\}, \{\xi_{n,v}\})$.

Examples of possible families that can be learned using the models defined in (1) and (2) are presented in Figs. 3 and 4.

2) Family Learning and Multitask Learning: Learning the location φ of a surface implies examining the training examples of several related tasks to determine a location that captures the underlying structure of the environment corresponding to these tasks. In order to understand how this is accomplished, we consider a set of N related tasks $\{T_n\} = (T_1, \ldots, T_N)$. All these tasks evolve in the same environment. For supervised learning, each task T_n is represented by a set of K_n training examples $\{(x_{nk}, y_{nk})\}$ where x_{nk} is an input and y_{nk} is the corresponding desired output. Learning the family as well as learning solutions

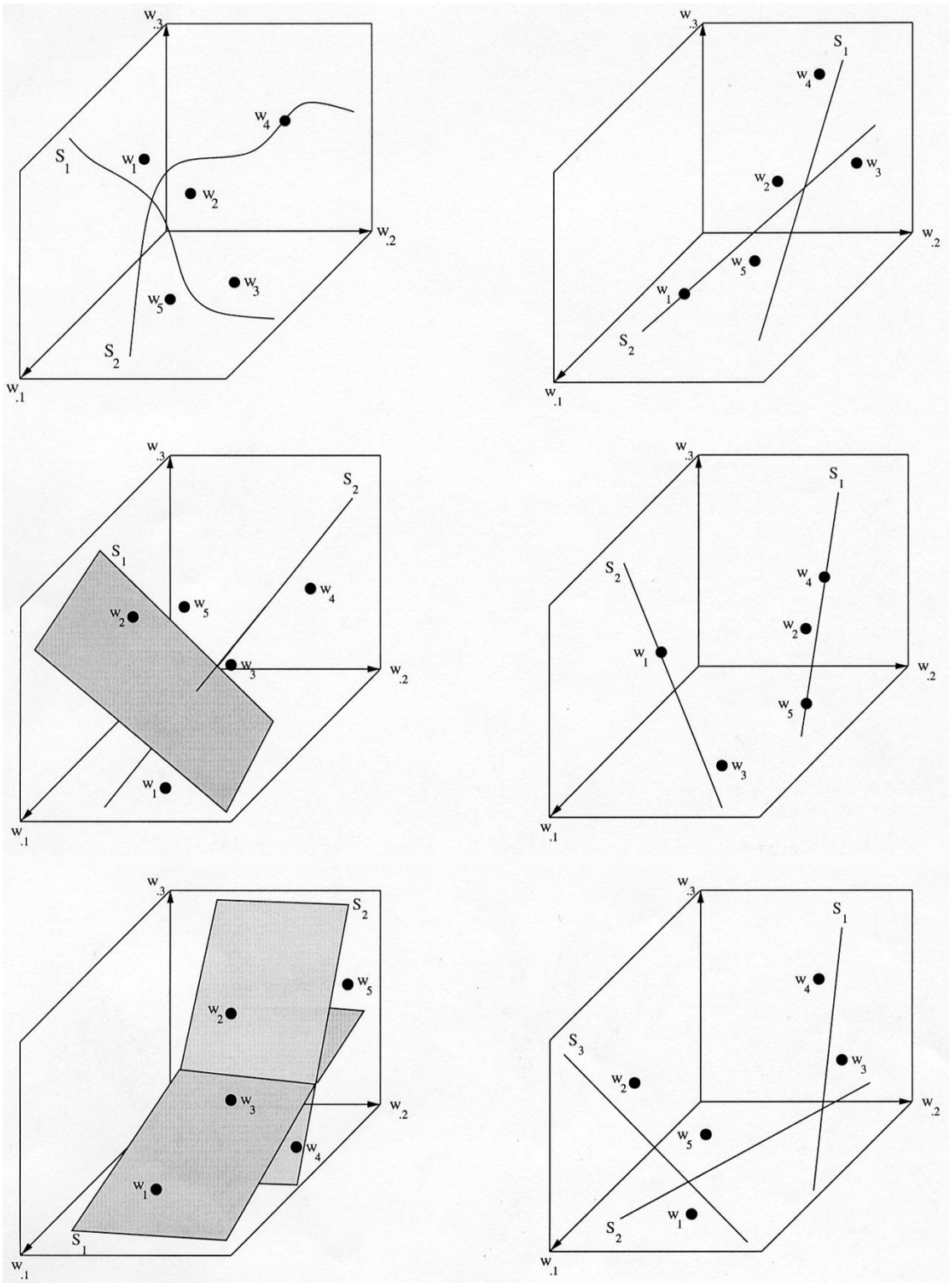


Fig. 4. Examples of families learned by the manifold learning method using the model defined in (2). A family S is represented by a mixture of manifolds $\{S_i\}$. The number of manifolds, their dimensionalities and their positions in the hypothesis space depend on the environment being modeled. The points $(w_1, \ldots, w_5)$ represent the solutions chosen for five tasks evolving in the environment defined by the family.

for all N tasks can be achieved by minimizing the following empirical risk:

$$R_{emp}(\alpha_1, \ldots, \alpha_N, \varphi) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{K_n} \sum_{k=1}^{K_n} C\left(f\left(g\left(\varphi, \alpha_n\right), x_{nk}\right), y_{nk}\right) \quad (4)$$

where $(\alpha_1, \ldots, \alpha_N, \varphi)$ corresponds to the set of parameters that need to be evaluated. Each α_n defines the position of the solution for task T_n on the surface defined by $g(\varphi, \cdot)$. C is a loss function that evaluates the "quality" of the solution learned for a task, and $f(w_n, x_{nk}) = f(g(\varphi, \alpha_n), x_{nk})$ defines the prediction output for task T_n when the input is x_{nk} (e.g., if the learning

model is a neural network, $f(w_n, x_{nk})$ is the output provided by the network when its parameters are equal to w_n and when the input is equal to x_{nk}).

The minimization of the empirical risk leads to the determination of parameters $(\alpha_1^*, \ldots, \alpha_N^*, \varphi^*)$ such that

$$(\alpha_1^*,\ldots,\alpha_N^*,\varphi^*) = \operatorname{argmin}_{\alpha_1,\ldots,\alpha_N,\varphi} \frac{1}{N}\sum_{n=1}^{N}\frac{1}{K_n}\sum_{k=1}^{K_n} C(f(g(\varphi,\alpha_n),x_{nk}),y_{nk}). \quad (5)$$

In order to better grab the difference between multitask learning and single-task learning, we define the equivalent of (1), (4) and (5) in the case of single-task learning

$$w_n = \alpha_n \quad (6)$$

$$R_{emp}(\alpha_1,\ldots,\alpha_N) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{K_n}\sum_{k=1}^{K_n} C(f(\alpha_n,x_{nk}),y_{nk}) \quad (7)$$

$$\alpha_n^* = \operatorname{argmin}_{\alpha_n} \frac{1}{K_n}\sum_{k=1}^{K_n} C(f(\alpha_n,x_{nk}),y_{nk}) \quad (8)$$

A comparison of (5) and (8) underscores the difference between multitask learning and single-task learning: whereas in the first case, the training examples of all the tasks are needed to define a family from which a solution for a particular task can be chosen, in the second case, each task is learned separately and no transfer or exchange of knowledge is performed between the tasks (i.e., the solution for each task can be chosen in the entire hypothesis space instead of being chosen from a restricted class of hypotheses).

3) Learning New Tasks: Once a family has been learned using the training examples of N tasks $(T_1, \ldots, T_N)$ [according to the model defined in (5)], this family can be used to learn new tasks that belong to the same environment as the N tasks used to generate the family. Learning a new task T_{N+1} consists of selecting a point w_{N+1} on the surface defining the family. This point should correspond to a solution that is representative of the properties specific to task T_{N+1}. Learning a new task thus consists in determining a position α_{N+1}^* such that

$$\alpha_{N+1}^* = \operatorname{argmin}_{\alpha_{N+1}} \frac{1}{K_{N+1}}\sum_{k=1}^{K_{N+1}} C(f(g(\varphi^*,\alpha_{N+1}),x_{N+1,k}),y_{N+1,k}) \quad (9)$$

where φ^* defines the location of the family, as computed in (5).

4) Manifold Learning versus Internal Representation Learning: In Section II, we described the internal representation learning method and we gave in Fig. 1 examples of the classes of hypotheses that can be represented by this method. A closer look at the class of solutions that can be considered by this method reveals that it corresponds to a particular choice of the surface defined by $g(\varphi, \cdot)$. Indeed, the class of solutions can be defined as follows:

$$\begin{bmatrix} w_{n,1} \\ w_{n,2} \\ \vdots \\ w_{n,p} \\ w_{n,p+1} \\ \vdots \\ w_{n,M} \end{bmatrix} = \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_p \\ \alpha_{n,p+1} \\ \vdots \\ \alpha_{n,M} \end{bmatrix} \quad (10)$$

where $w_n = (w_{n,1}, \ldots, w_{n,M})$, M is the dimensionality of the hypothesis space W, and p is the number of parameters that are shared by the learning models (i.e., by the neural networks) trained on the tasks $\{T_n\}$. $\varphi = (\varphi_1, \ldots, \varphi_p)$ and $\alpha_n = (0, \ldots, 0, \alpha_{n,p+1}, \ldots, \alpha_{n,M})$.

Internal representation learning, therefore, considers that tasks are related if they evolve in a common environment and if they have a common internal representation. It is a special case of manifold learning in which the manifold shape is affine and axis-aligned. Manifold learning is less restrictive: tasks are considered related only if they evolve in a common environment. The flexibility of the family definition in (1) allows for the representation of various families (and therefore of various environments). The families can be affine or nonaffine. They can also be axis-aligned or not aligned.

5) Manifold Learning versus Family Discovery: Family discovery was introduced in Section II and examples of the classes of solutions that this method can represent were illustrated in Fig. 2. The objective function being minimized in family discovery can be defined as follows:

$$E(\{w_n\},S) = \frac{1}{N}\sum_{n=1}^{N}\left((w_n - \operatorname{Proj}(S,w_n))^2 + \frac{1}{K_n}\sum_{k=1}^{K_n} C(f(w_n,x_{nk}),y_{nk})\right) \quad (11)$$

where $E(\{w_n\}, S)$ is the error corresponding to a particular choice of $\{w_n\}$ and S. S is the family (corresponding to an affine manifold or a mixture of affine manifolds), and $\operatorname{Proj}(S, w_n)$ is the projection of w_n on S.

There are two main differences between family discovery and manifold learning. The first difference lies in the fact that family discovery uses mixtures of affine manifolds to represent nonaffine families while manifold learning can implement nonaffine families without having to resort to the mixture of affine manifolds model. The second difference is related to the dimensionality of the families learned by both methods: in family discovery, the surface S is learned by applying a principal component analysis to the points $\{w_n\}$. The dimensionality of the surface therefore depends on the number N of tasks and is always smaller that N. In manifold learning, the dimensionality of the surface is not constrained by the value of N. This flexibility allows the choice of dimensionalities that depend on the "type" of properties defining an environment. Stringent environment-related properties call for the use of low-dimensional families (because few hypotheses respect these properties) while lax proper-

ties can be represented in hypothesis space by high-dimensional families (given that many hypotheses respect these properties).

6) Capacity and Capacity Control: There are two different capacities that need to be considered when discussing multitask learning. The first one is the capacity of a learning model when this model is forced to select a hypothesis that belongs to a family of hypotheses (i.e., the richness of f when varying w_n). The second one is the capacity of the family (i.e., the richness of g when varying φ). Both capacities depend on the level of smoothness of the family and on the dimensionality of the family. They increase when the smoothness of the family decreases and/or when the dimensionality increases. The control of both capacities is therefore undertaken by controlling these two factors. A control of the capacities is necessary when the number of training examples representing a group of related problems is small and/or when the number of problems is small.

When the dimensionality of the surface defining the family increases, the capacity of the learning model increases. This is the effect of having a family containing a large number of hypotheses among which the learning model can select a hypothesis suitable for a particular problem. A large capacity for the learning model can be considered when the number of training examples is large. But when this number is small, it is preferable to consider a smaller capacity. The number of training examples also affects the quality of the family. If the training sets of a group of related problems are small, it is recommended to control the smoothness of the surface, because it will be difficult to identify the underlying structure of the environment from a limited amount of information. The difficulty of learning environment-related properties when the training sets are small can be alleviated when the number of available problems increases.

The capacity of the learning models trained on a group of related tasks can also be controlled by forcing the models to choose hypotheses that are geometrically close to each other. This principle, inspired from the "soft-weight sharing" method [18], can be implemented by considering a penalty term that needs to be minimized along with the minimization of the empirical risk:

$$\begin{aligned} &E(\alpha_1,\ldots,\alpha_N,\varphi,\gamma) \\ &=\frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{K_n}\sum_{k=1}^{K_n}C\left(f\left(g\left(\varphi,\alpha_n\right),x_{nk}\right),y_{nk}\right)\right. \\ &\qquad\left.+\lambda\left\|g\left(\varphi,\alpha_n\right)-g(\varphi,\gamma)\right\|^2\right) \end{aligned} \tag{12}$$

where E is the objective function and γ is the position of a point on the surface defining the family. Learning in this context forces the selection of solutions that are close to the point whose position is γ. λ is a constant whose value defines the "weight" of the penalty term. This method can easily be extended to define several neighborhoods on the surface instead of defining only one neighborhood [18].

The capacity of a family needs especially to be controlled when the family, learned using a group of related problems, might be used to learn new problems evolving in the same environment. In that case, it might be particularly important to control the smoothness of the surface defining the family in order to avoid "task-over-fitting" situations: if the number of problems used to learn the family is very small, a family having a low level of smoothness might adapt to these problems and will not be able to generalize properly when new problems are to be learned. This happens when the family contains hypotheses that are specific to the original set of problems and that are not general enough to be representative of the environment. Another factor to consider when learning new problems is the dimensionality of the surface defining the family. Indeed, the dimensionality of the surface needs to be large enough in order to have a family that contains solutions that are representative of the properties of the environment instead of only containing solutions that are specific to the properties of the original set of problems.

B. Definition of an Affine Family

In the experiments presented in the next section, we only considered affine manifolds to represent families of hypotheses. We limited ourselves to this form of surface for two reasons: we wanted to verify if this simplest type of surface could be used to learn an environment. And given that we wanted to perform a thorough analysis of multitask learning by comparing the manifold learning method to four other methods and by examining the impact of several factors on the generalization performance of multitask learning, we had to limit the number of experiments performed with each method.

A d-dimensional affine manifold defined in an M-dimensional hypothesis space can be represented as follows [11]:

$$\begin{bmatrix} w_{n,1} \\ w_{n,2} \\ \vdots \\ w_{n,d} \\ w_{n,d+1} \\ \vdots \\ w_{n,M} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ \theta_{d+1,1} & \theta_{d+1,2} & \theta_{d+1,3} & \cdots & \theta_{d+1,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \theta_{M,1} & \theta_{M,2} & \theta_{M,3} & \cdots & \theta_{M,d} \end{bmatrix} \begin{bmatrix} \alpha_{n,1} \\ \alpha_{n,2} \\ \vdots \\ \alpha_{n,d} \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_d \\ \beta_{d+1} \\ \vdots \\ \beta_M \end{bmatrix} \tag{13}$$

where θ defines the direction of the manifold and β represents the offset of the manifold wrt the origin of the hypothesis space. The manifold can therefore be defined as: $w_n = g(\varphi,\alpha_n) = \theta\alpha_n + \beta$. The location of the manifold is $\varphi = (\theta,\beta)$.

The capacity of a learning model that needs to select a hypothesis belonging to an affine family can be controlled by controlling the dimensionality of the family. When $d = 0$, the family is represented by one point in the hypothesis space. This point corresponds to β. In that case, all learning models trained on related problems are forced to select the same hypothesis: $w_n = \beta$. This is complete parameter sharing. At the other extreme, when $d = M$, the family represents the entire hypothesis space. This is single-task learning.

Given that β is a point that is located on the affine manifold (the position of this point on the manifold is $\gamma = 0$ [i.e.,

$\beta = g(\varphi, \gamma) = g(\varphi, 0)]$), we used this point as the one defining the neighborhood in the model introduced in (12). Given this choice, the objective function defined in (12) becomes

$$\begin{aligned} E(\alpha_1, \ldots, \alpha_N, \theta, \beta) &= \frac{1}{N} \sum_{n=1}^{N} \left(\frac{1}{K_n} \sum_{k=1}^{K_n} C\left(f\left(g\left(\varphi, \alpha_n\right), x_{nk}\right), y_{nk}\right) + \lambda \left\| g\left(\varphi, \alpha_n\right) - \beta \right\|^2 \right) \\ &= \frac{1}{N} \sum_{n=1}^{N} \left(\frac{1}{K_n} \sum_{k=1}^{K_n} C\left(f\left(\left(\theta \alpha_n + \beta\right), x_{nk}\right), y_{nk}\right) + \lambda \left\| \theta \alpha_n \right\|^2 \right). \end{aligned} \tag{14}$$

IV. Experiments

The manifold learning method described in the previous section as well as the internal representation learning method [2], [5] and the family discovery technique [19] described in Section II are used to learn *invariant* Boolean functions. The results obtained when applying each method are analyzed and compared to single-task learning. They are also compared to a stringent form of multitask learning whereby an identical solution is learned for all the related tasks. This form is called learning a common solution. It is a special case of the manifold learning method and the internal representation learning method where the family is represented by a *single point* in hypothesis space. For a thorough comparison to be conducted, it is important to determine all the factors that could have an impact on each of these five methods. The simplest way to determine these factors is to examine (4) for the manifold learning method, the internal representation learning method and the common solution learning method (given that the last two methods are a special case of the first one), (7) for single-task learning, and (11) for the family discovery method. The performances of these methods depend on the number N of tasks and the number K_n of training examples for each task T_n. They depend on the quality of the training set $D_n = \{(x_{nk}, y_{nk})\}$ representing each task T_n. They also depend on the particular choice of the group of tasks $G = \{T_n\}$ (those tasks being chosen from the set of all tasks that define the environment). Finally, they depend on f, the type of function implemented by a learning model to learn a task T_n (e.g., in the case of feedforward neural networks, f corresponds to the output of a network and depends on the architecture of the network). The five learning methods differ in the type of family they use to define the domain-related properties of the tasks. In the case of single-task learning, the family corresponds to the entire hypothesis space. For multitask learning, the type of the family depends on the multitask learning method that is used.

In order to compare all five learning methods, and in order to analyze the influence exercised by N, K_n, D_n, and G, different values for each of these four factors were considered and each possible combination of the values of these factors was used to train each learning method. This design allows to test the interactions that exist between these factors. A statistical analysis was then performed to evaluate the differences in the generalization performance of all five learning methods.

In the following section, we describe the data used to perform the experiments. We then describe the experimental setting and the results obtained when applying single-task learning and the four multitask learning techniques.

A. Problem Description

We work with *invariant* Boolean functions that have Boolean inputs and Boolean outputs such that *the output of any invariant function depends only on the number of "1"s in the input, regardless of their positions*. The input examples were defined in $\{0,1\}^{14}$. Only those inputs containing 4 to 10 "1"s were considered. The remaining examples were discarded because the number of available examples containing 0 to 3 "1"s and 11 to 14 "1"s was too small for performance evaluation. The number of invariant Boolean functions with inputs containing four to ten "1"s is 128. Two trivial functions were discarded: the function whose output is always 0 and the function whose output is always 1.

All these Boolean functions evolve in an environment defined by the invariance property: whatever the function considered in this environment, the output of the function depends only on the number of "1"s in the input, whatever their position in the input vector.

B. Selection of the Data Sets and Groups of Tasks

The generation of the groups of tasks and of the datasets is illustrated in Fig. 5. 30 functions were chosen without replacement among the 126 available Boolean functions. These functions were used to generate on the one hand, ten groups of three functions each and on the other hand, five groups of six functions each. The groups containing six functions correspond to the pairing of groups of three functions.

Five thousand input examples were chosen without replacement from the set of inputs containing 4 to 10 "1"s (a nearly equal number of inputs was selected from each input category where a category is defined by the number of "1"s in the input). These examples were divided into five input sets of 1000 examples each. Each function (among the 30 chosen functions) had five different datasets corresponding to the five input sets and their corresponding desired values.

Each data set containing 1000 examples was divided as follows: a test set of 500 examples, a validation set of 200 examples, and a training set of at most 300 examples. In fact, three different training set sizes were considered: 50, 100 and 300 examples. The size of the validation set was not modified in order to control the setting of the experiments. The validation set was used to perform model selection.

This setup can be considered to analyze the impact of using different groups of tasks ($\{G_{ij}\}$, $\{G_i\}$) and the influence of the size N of the groups (the G_{ij} groups contain three functions or tasks and the G_i groups contain six functions). It also allows to analyze the effect of using five different datasets (each task T_n has five datasets ($D_{n,1}, \ldots, D_{n,5}$)), and varying amounts of training examples ($K_n = 50$, 100 and 300 examples).

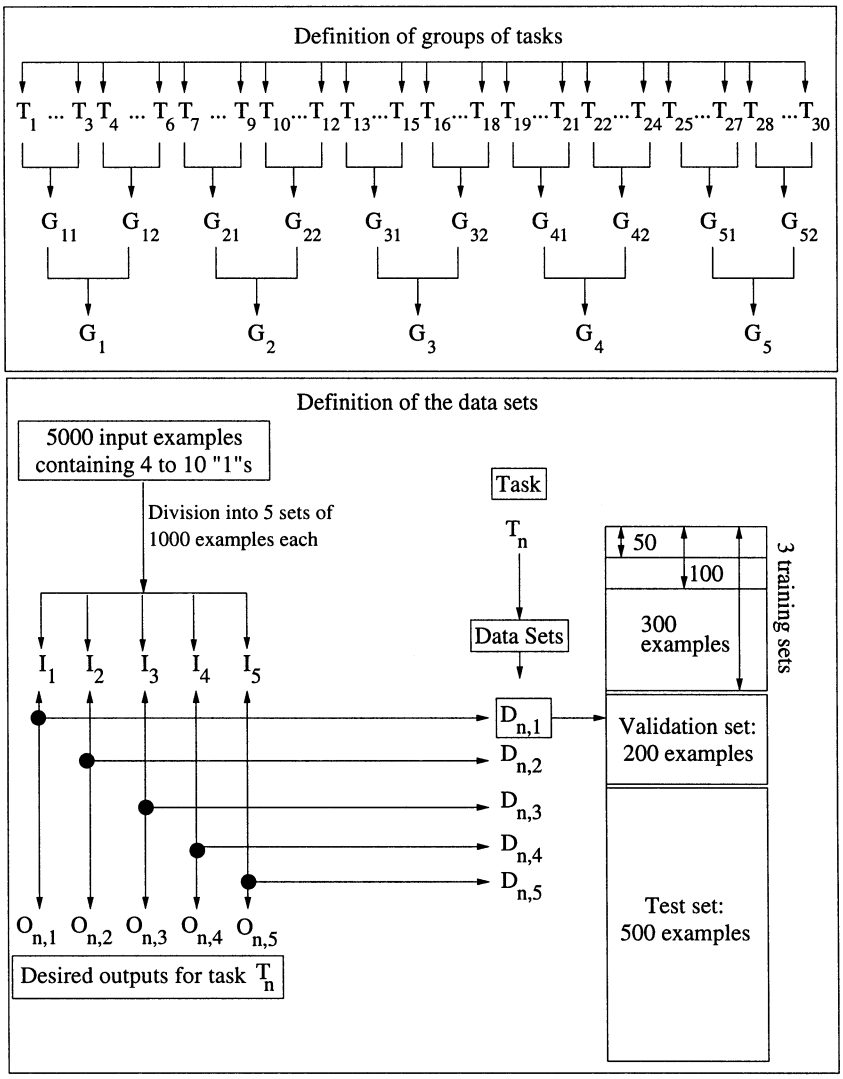


Fig. 5. Experimental setting used to learn 30 invariant Boolean functions ($T_1, \ldots, T_{30}$). The functions were used to generate ten groups of three tasks each, and five groups of six tasks each. For each function T_n, five different datasets were generated: ($D_{n,1}, \ldots, D_{n,5}$). Each data set contains 1000 examples, formed of 200 validation examples, 500 test examples, and 300 training examples which were used to form three training sets containing 50, 100, and 300 examples.

C. Experimental Setting

1) Single-Task Learning: Single-task learning consists in separately learning each task in a group of tasks. Neural networks were used to learn the tasks. For each combination of the values of N, K_n, D_n, and G, 23 different feedforward neural network architectures were considered. The architectures contained zero to four hidden layers with varying numbers of units per hidden layer. This large number of architectures was considered in order to find a suitable architecture for each function or task. Note that the apparently optimal architecture can vary for different functions or tasks in a group. At no time were the test sets used. Only the validation sets were used for early stopping and for model selection based on the proportion of classification errors (and on the mean squared error in cases of equal proportions of classification errors).

Once a separate model or neural network architecture was chosen for each task in a group of tasks, the generalization performance of the group was computed as the mean of the classification generalization performance of the selected models for all the tasks in the group.

2) Affine Manifold Learning: The first multitask learning method to be tested is the affine manifold learning method described in Section III. The model used in the experiments corresponds to the one defined in (14). In order to apply this method, two hyper-parameters need to be specified: the dimensionality d of the affine manifold representing the family, and the "weight" λ of the penalty term. Both parameters were chosen based on the validation sets performances. Several manifold dimensionalities were tested which ranged from one to M (where M is the number of parameters defining a neural network) by steps of ten. And four different values for λ were chosen: 0, 0.01, 0.1, and 1.0. 8 neural-network architectures among the set of 23 architectures used in single-task learning were selected. The experimental setting can, therefore, be summarized as follows: for each combination of the values of N, K_n, D_n, and G, try eight different neural-network architectures, and for each architecture, try each possible combination of d and λ. The validation

sets of the tasks in G are used to choose the optimal triplet (*network architecture, d, λ*).

3) Internal Representation Learning: The internal representation learning method [2], [5] was implemented by training several identical neural networks on the tasks in a group and by forcing the networks to share their first layers. For each combination of N, K_n, D_n, and G, all 23 neural network architectures used in the single-task learning experiments were tested. For each architecture, L experiments were performed which consisted in sharing the first l layers among the networks with $l = 1, \ldots, L$ and L is the number of hidden and output layers in the chosen architecture. Sharing all the hidden and output layers is equivalent to setting $d = 0$ in the affine manifold learning method. For each combination of N, K_n, D_n, and G, the validation sets of the tasks in G were used to choose the optimal pair (*network architecture, number l of shared layers*).

4) Family Discovery: Before explaining the experimental setting used to test the family discovery method [19], the iterative process applied to generate a family in this method is going to be described. We will only consider the case in which the family is formed of one manifold (mixtures of manifolds were not used because of the small size of the groups: N is either equal to three or six). The family discovery method was applied to a group of tasks according to the following general framework:

1) Initialization: train separately each task T_n contained in the group of tasks (early stopping based on the validation performance is used). At the end of the training process, the learning model trained on task T_n is defined by a set of "optimal" parameters w_n^*.
2) Family definition: perform a principal component analysis (PCA) on the "optimal" parameters $\{w_n^*\}$ and select the eigenvectors corresponding to the largest eigenvalues obtained in the PCA. Use the chosen eigenvectors to define an affine manifold S^* in hypothesis space.
3) Update of the solutions: relearn each task T_n by determining a new value of w_n^* chosen according to the following model:

$$w_n^* = \operatorname{argmin}_{w_n} \left(\lambda \left(w_n - \operatorname{Proj}\left(S^*, w_n \right) \right)^2 + \frac{1}{K_n} \sum_{k=1}^{K_n} C\left(f\left(w_n, x_{nk} \right), y_{nk} \right) \right) \quad (15)$$

   where λ is a constant representing the weight of the penalty term, and S^* is the manifold learned at the previous step. Early stopping is used when learning T_n.
4) Compute the mean validation performance of the group using the parameters $\{w_n^*\}$ learned at the previous step, and compare this mean performance to the previous mean validation performances observed the last times the previous step was applied. If early stopping needs to be performed, stop the learning process. Otherwise, go to step 2.

When applying the family discovery method, we were confronted by a problem at Step 3). We needed to choose a method of initialization of the parameters w_n. Initially, we decided to use the "optimal" parameters w_n^* obtained at a previous step of the iterative process to initialize w_n. This method led to a very slow (in some cases extremely slow) convergence of the iterative process. Choosing small values of λ somewhat alleviated the slowness of the process but it did not eliminate it. We therefore decided to try another method of initialization which consisted in choosing "random" parameters. Although the learning process became faster in general, there still were several cases or several experiments that were characterized by a very slow convergence. We do not fully understand the factors provoking this slowness. In some cases, we noticed that the "optimal" parameters w_n^* obtained at a previous step of the iterative process had some large values which caused a saturation problem when using neural networks. Initializing with random parameters helped eliminate part of the problem associated to having large initial values. But the problem was not completely solved because the manifold S^* was defined using the "optimal" parameters w_n^* obtained at a previous step, and its location in the hypothesis space was therefore influenced by the presence of large values. This is why using small values for λ was helpful. But it should be noted that these observations were limited to the examination of a limited number of experiments. A thorough analysis should be undertaken to understand the problem. The observed slowness forced us to limit the number of experiments performed with the family discovery method.

The following experiments were performed using the family discovery method: for each combination of N, K_n, D_n, and G, the neural network architecture that led to the smallest mean validation error (i.e., the smallest group validation error) when each task is learned separately in Step 1), was chosen. This architecture was chosen from the set of the 23 architectures used in single-task learning. Given the chosen architecture, two parameters needed to be specified: the number n of eigenvectors used to define the affine family, and the value of λ. For a group containing N tasks, the number n of eigenvectors that can be considered is $n = 1, \ldots, N-2$. As for λ, five different values were considered: 0.001, 0.01, 0.1, 1, and 5. Validation set performance was used to choose a pair (n, λ).

5) Learning a Common Solution: When a group of related tasks $G = \{T_n\}$ is available, learning a common solution can be achieved according to the following model:

$$w^* = \operatorname{argmin}_w \frac{1}{N} \sum_{n=1}^{N} \frac{1}{K_n} \sum_{k=1}^{K_n} C\left(f\left(w, x_{nk} \right), y_{nk} \right) \quad (16)$$

In that case, the solution w_n^* for each task T_n is: $w_n^* = w^*$.

All 23 neural network architectures used in the single-task learning experiments were considered. The choice of the "optimal" architecture was based on the validation performance.

D. Results

A statistical analysis using a generalized estimating equations (GEE) model [27][1] was performed. The factors considered in this analysis were: 1) six learning algorithms, namely single-task leaning (STL), affine manifold learning (AML), learning an internal representation (LIR), family discovery with a random initialization of the parameters (FDR), family

[1]The GEE model is an extension of the generalized linear models [16].

TABLE I
PERCENTAGE OF THE INCREASE OR DECREASE OF THE ODDS OF MAKING A CLASSIFICATION ERROR WHEN A MULTITASK LEARNING ALGORITHM IS USED INSTEAD OF THE SINGLE-TASK LEARNING TECHNIQUE. ALL INCREASES OR DECREASES WHICH ARE DIFFERENT FROM ZERO IN A STATISTICALLY SIGNIFICANT WAY AT THE 5% LEVEL ARE "FRAMED"

		Learning Algorithm				
K_n	N	AML	LIR	FDD	FDR	LCS
50	3	[-5.67]	-0.18	-0.44	[-3.35]	[49.12]
	6	[-6.58]	[-7.29]	-4.48	[-8.30]	[61.03]
100	3	[-31.47]	[-32.19]	-6.24	[-12.23]	[79.17]
	6	[-55.59]	[-61.74]	[-13.58]	[-18.86]	[109.07]
300	3	[-96.18]	[-96.18]	-19.46	[-75.58]	[783.67]
	6	[-96.19]	[-96.19]	-35.85	[-96.01]	[1125.60]

discovery with predetermined initial weights (FDD) (i.e., family discovery with initialization using the previous optimal parameters), and learning a common solution (LCS); 2) the size K_n of the training sets; 3) the number N of tasks; 4) the choice D_n of the datasets; and 5) the choice G of the groups of tasks. In what follows, we will present the main results concerning the evaluation of the learning algorithms.

In order to perform a statistical analysis using a GEE model, we had to separate the experimental results into six categories corresponding to the different combinations of (K_n, N). For each possible combination, we considered the following GEE model:

$$E = 1 + A + G + D \tag{17}$$

where E is the number of classification errors, A is a factor representing the learning algorithms, G is a factor representing the groups of tasks and D is a factor representing the different training sets. No interaction between these three factors was considered because tests revealed that the interactions were not statistically significant. It should be noted that all three factors are categorical and that the model defined in (17) is treated as a generalized estimating equations model with a binomial response (the number of classification errors) and a logit link [16].

The results of the statistical analysis based on the model defined in (17) are presented in Table I. These results show the percentage of an increase or a decrease of the odds of making a classification error when a multitask learning algorithm is used instead of the single-task learning algorithm [16]. And the generalization classification errors of all six learning algorithms are illustrated in Fig. 6. The results presented in Table I and in Fig. 6 lead to the following conclusions:

1) Affine manifold learning (AML) and family discovery with random initialization (FDR) are the only multitask learning algorithms that significantly generalize better than single-task learning (STL) for all choices of (K_n, N). But the decrease in the classification error observed when the affine manifold learning algorithm is used is more important than the decrease associated with the family discovery method.
2) Apart from the case where $K_n = 50$ and $N = 3$, learning an internal representation (LIR) leads to generalization performances that are significantly better than single-task learning and that are comparable to the performances of affine manifold learning.
3) Family discovery with a random weight initialization (FDR) generalizes better than family discovery with an initialization based on predetermined initial weights (FDD). The second method is in general comparable to single-task learning.
4) Learning a common solution (LCS) is always worse than single-task learning (STL). All other multitask learning algorithms generalize better than single-task learning. Also, the generalization performance of learning a common solution worsens when the number N of tasks increases. For the other multitask learning algorithms, an increase in N leads to an improvement in the generalization performance.
5) The classification error of the different learning algorithms decreases when the number K_n of training examples increases.
6) For $K_n = 300$, affine manifold learning and internal representation learning generalize perfectly (i.e., they do not make any classification error).
7) When K_n is very small, increasing the number N of tasks does not help substantially. N should probably be very large to observe a difference in the generalization performance. This is a consequence of the lack of information in each training set. A lot of properties could be extracted from small training sets, some of them correct and others false or too specific to the training examples. Identifying correct environment-related properties cannot be achieved except in the presence of a large number of tasks to be able to distinguish the properties that are common to all tasks from the properties that are specific to each training set.
8) The performances of multitask learning improve when the number K_n of training examples increases. But the difference between multitask learning and single-task learning tends to level out when K_n is large because the number of training examples is large enough to allow single-task learning to perform well without having to use any kind of additional knowledge.

We conclude this section by discussing some observations concerning the affine manifold learning method. It was mentioned in Section III-B that the capacity of a learning model that is forced to select a hypothesis that belongs to the affine family, can be controlled by controlling the dimensionality d of the family. And it was suggested that when the number of training examples is small, smaller values of d might be preferable, and when the number of training examples increases, the necessity to control the capacity of the learning model can be relaxed. An analysis of the dimensionalities chosen (based on the validation performances) when performing the experiments described above confirm this observation. When the size of the

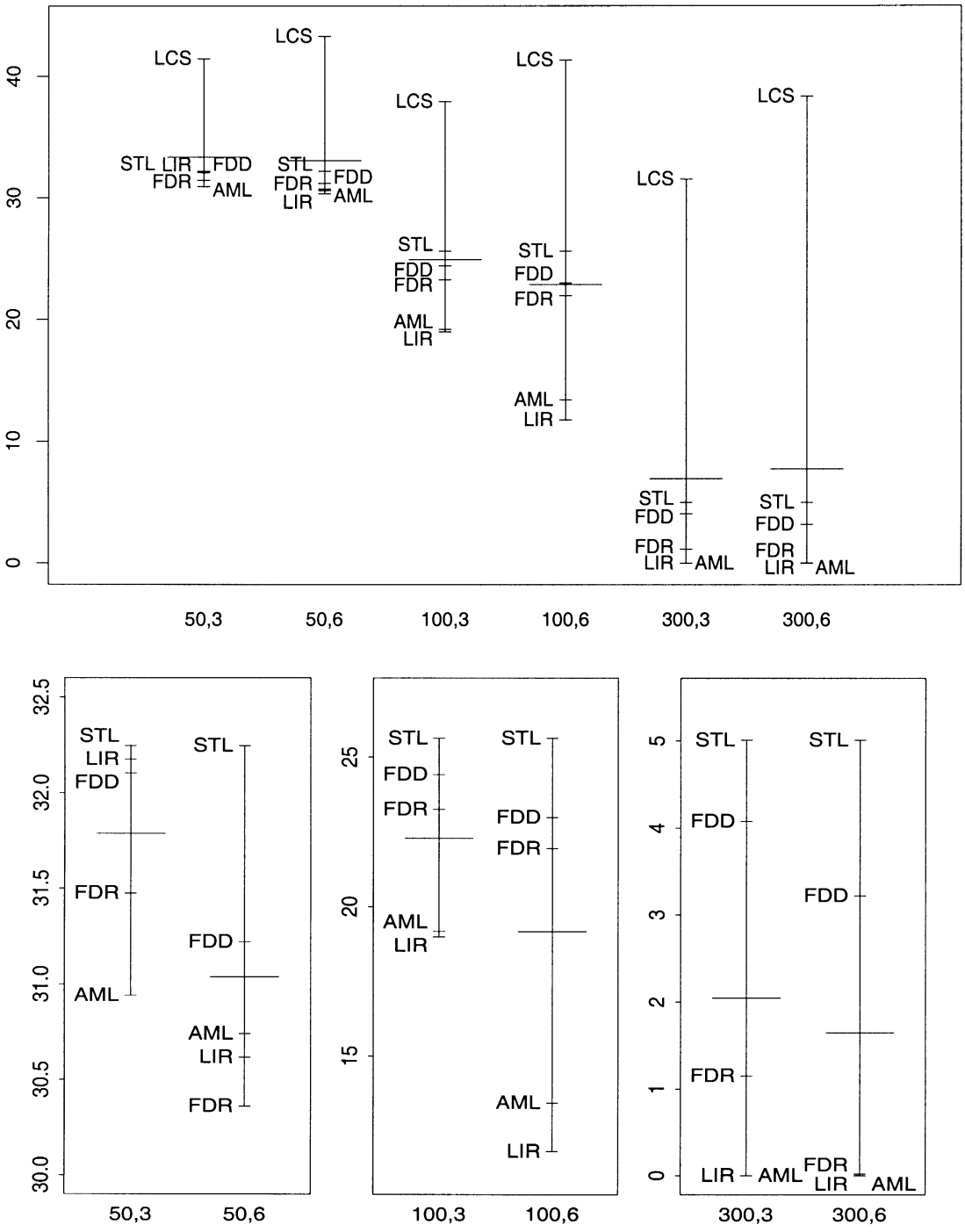


Fig. 6. Mean generalization classification errors (in %) of six learning algorithms: Affine manifold learning (AML), learning an internal representation (LIR), family discovery with random initial weights (FDR), family discovery with predetermined initial weights (FDD) (i.e., family discovery with initialization using the previous optimal parameters), learning a common solution (LCS) and single-task learning (STL). The values on the x-axis correspond to pairs: (K_n, N). For each pair and for each learning algorithm, the displayed mean generalization classification error is the mean of the errors computed over all combinations of D_n and G. The first figure plots the results for all pairs (K_n, N) while the remaining figures correspond to zooms of the first figure for different values of K_n (50,100,300).

training sets $K_n = 50$, there was a tendency to choose small values of d. For $K_n = 100$, this tendency started to level out. And for $K_n = 300$, it was possible to obtain zero classification errors even when d was large.

We also mentioned in Section III-B that the capacity of a learning model can be controlled by identifying a neighborhood on the affine family and forcing all the learning models trained on tasks evolving in the same environment to choose solutions in this neighborhood. This constraint led to generalization performances better than those observed when no neighborhood is defined. An interesting observation was made when we analyzed the relationship between the choice of the value of λ (the weight of the constraint) and the choice of d (the dimensionality of the affine family). For $K_n = 50$, small values of d were chosen and there was no particular tendency observed in the choice of λ (4 values of λ were considered: 0, 0.01, 0.1, and 1.0). This means

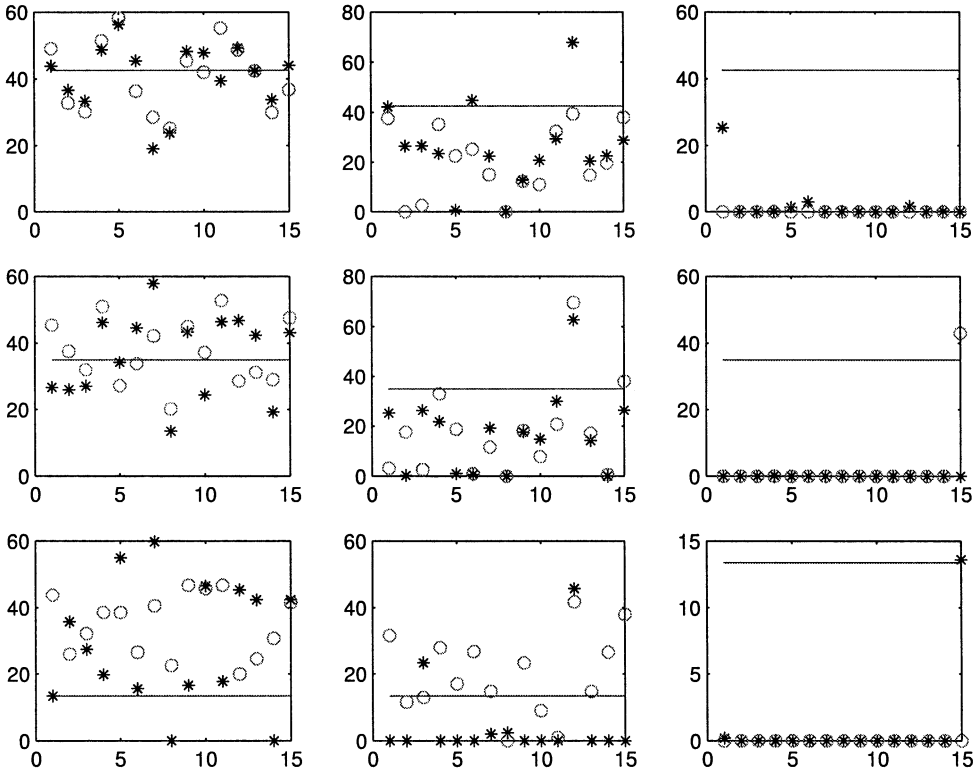


Fig. 7. Learning a new Boolean function using the families learned by the affine manifold learning method ($*$ points) and by the internal representation learning technique (o points). Each $*$ or o point in a plot corresponds to a family learned from a different group of tasks (ten groups of three tasks each and five groups of six tasks each). In each plot, the x-axis corresponds to the index of the group of tasks, the y-axis corresponds to the generalization classification error (in %). The horizontal bar in each plot represents the generalization error obtained in single-task learning. The plots are organized as follows: from top to bottom, each row corresponds to learning the new Boolean function with different numbers of training examples (50,100,300), and from left to right, each column corresponds to using families that were obtained with training sets of 50, 100, and 300 examples.

that the capacity of a learning model was controlled by controlling d. For $K_n = 100$, no particular tendency was observed for the choice of d. But a clear tendency toward large values of λ was apparent. In that case, the capacity was mostly controlled by controlling λ.

E. Learning to Learn

A final set of experiments was performed to test whether the family learned by a multitask learning method can be used to learn novel tasks. Both the affine manifold learning method and the internal representation learning method were tested. These methods were chosen because they had, in most cases considered in the previous section, comparable generalization performances and because they led to important improvements of the generalization performances when compared to single-task learning.

Five different invariant Boolean functions were selected from the set of (126–30) invariant Boolean functions (126 is the total number of invariant Boolean functions and 30 is the number of functions used in the previous section to generate several families). Each new Boolean function has five different datasets, as before.

The first experiments consisted in applying single-task learning. For each new function and each data set, 3×23 experiments were performed which correspond to using different values for the size of the training set (50,100,300) of the function and trying all 23 neural network architectures used in the original single-task learning experiments. The choice of the "optimal" architecture for each function, for each dataset, and for each size of the training set, was based on the validation set performance.

Then multitask learning was applied. For each learning method, each new function and each data set, $3 \times 3 \times 15$ different types of experiments were performed which correspond to all combinations of the size of the training set of the new function (50,100,300), the size of the training sets that were used when learning the family (50,100,300) and the number of groups of tasks used to learn the family (there are ten groups of three functions each, and five groups of six functions each). Learning a new task or function T_{N+1}, using a family already defined, can be undertaken according to the model defined in (9) for the internal representation learning method and according to the following model for the affine manifold learning method:

$$\alpha_{N+1}^{*} = \operatorname{argmin}_{\alpha_{N+1}} \cdot \left(\frac{1}{K_{N+1}} \sum_{k=1}^{K_{N+1}} C\left(f\left(\left(\theta^{*}\alpha_{N+1} + \beta^{*}\right), x_{N+1,k}\right), y_{N+1,k}\right) + \lambda \left\|\theta^{*}\alpha_{N+1}\right\|^{2} \right) \tag{18}$$

where N is the number of tasks used to generate the family, $\varphi^{*} = (\theta^{*}, \beta^{*})$ defines the location of the family as computed when learning the N tasks, and β^{*} is the position of a point

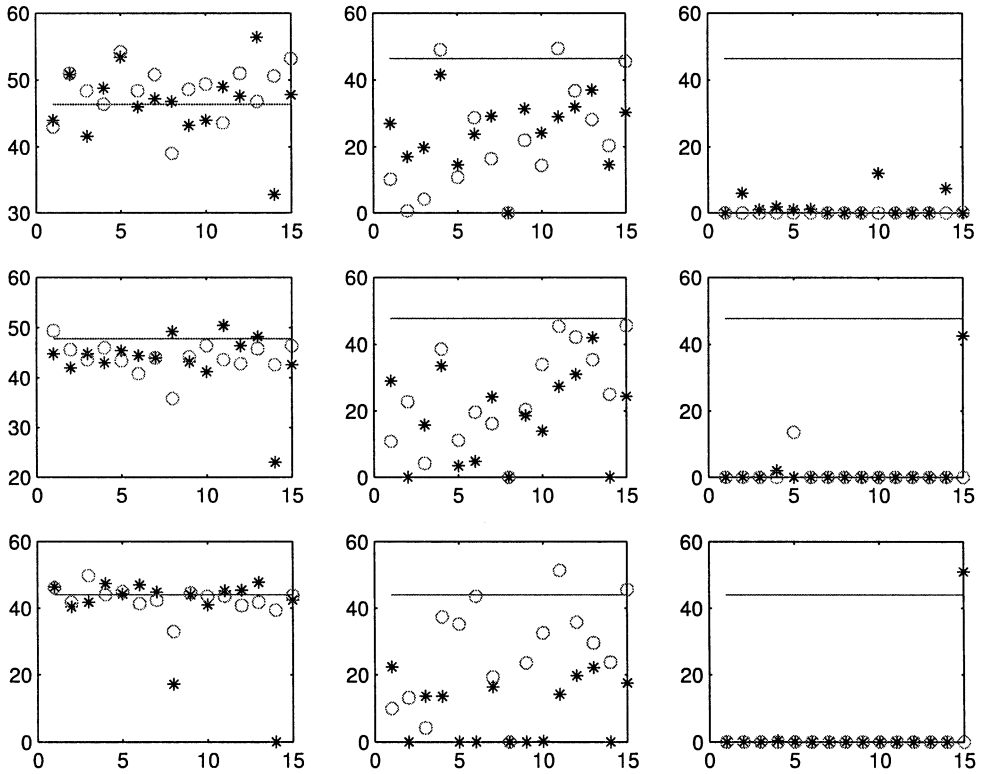


Fig. 8. Learning a new Boolean function (other than the one considered in Fig. 7) using the families learned by the affine manifold learning method (* points) and by the internal representation learning technique (o points). Each * or o point in a plot corresponds to a family learned from a different group of tasks (ten groups of three tasks each and five groups of six tasks each). In each plot, the x-axis corresponds to the index of the group of tasks, the y-axis corresponds to the generalization classification error (in %). The horizontal bar in each plot represents the generalization error obtained in single-task learning. The plots are organized as follows: from top to bottom, each row corresponds to learning the new Boolean function with different numbers of training examples (50,100,300), and from left to right, each column corresponds to using families that were obtained with training sets of 50, 100, and 300 examples.

defining a neighborhood on the family. This point was determined or chosen when learning the family and the N tasks.

The experiments presented in this section were, therefore, performed to test the *quality* of a family, i.e., to test if a family learned using a limited number of tasks evolving in an environment can be used to learn other tasks that evolve in the same environment.

Examples of the results that were obtained when learning two different new functions are illustrated in Figs. 7 and 8. A summary of the percentage of classification errors observed when learning the new tasks is presented in Fig. 9. These results lead to the following conclusions:

1) When K_n is small ($K_n = 50$), the family is learned using a limited number of examples. In that case, single-task learning outperforms both multitask learning methods except in two cases: when $N = 6$ and $K_{N+1} = 50$ and 100, affine manifold learning outperforms single-task learning.
2) When $K_n = 100$, both multitask learning methods outperform single-task learning except when $K_{N+1} = 300$ and $N = 3$, in which case single-task learning generalizes better than learning an internal representation but is outperformed by affine manifold learning. This difference between both multitask learning methods is in contrast with results presented in Table I. Indeed, in Table I, both multitask learning methods have comparable performances when learning a family but the families learned using the affine manifold learning method seem to be more representative of the environment than those learned using the internal representation learning method.
3) When $K_n = 300$, the difference between the generalization performances of multitask learning and single-task learning is very large. The generalization performances of multitask learning are nearly perfect (the number of classification errors is close to zero) even when K_{N+1} is very small ($K_{N+1} = 50$).
4) The generalization performances of both multitask learning methods improve when K_n, N and K_{N+1} increase.

V. Task Similarity

An important topic that needs to be addressed when dealing with multitask learning is the notion of task similarity or task relatedness. We mentioned that for multitask learning to be applied, one needs to consider tasks evolving in a common environment. While this condition is imposed by all multitask learning methods, some methods go a step further and assume special types or special classes of similarity. For example, in the internal representation learning method, tasks are considered similar if they have a common internal representation. Also, in a variation of this method [24], [25], a third constraint is imposed: tasks are considered similar if their solutions are geometrically close to each other in the solutions space (or in the hypothesis space). This proximity constraint is imposed in several multitask

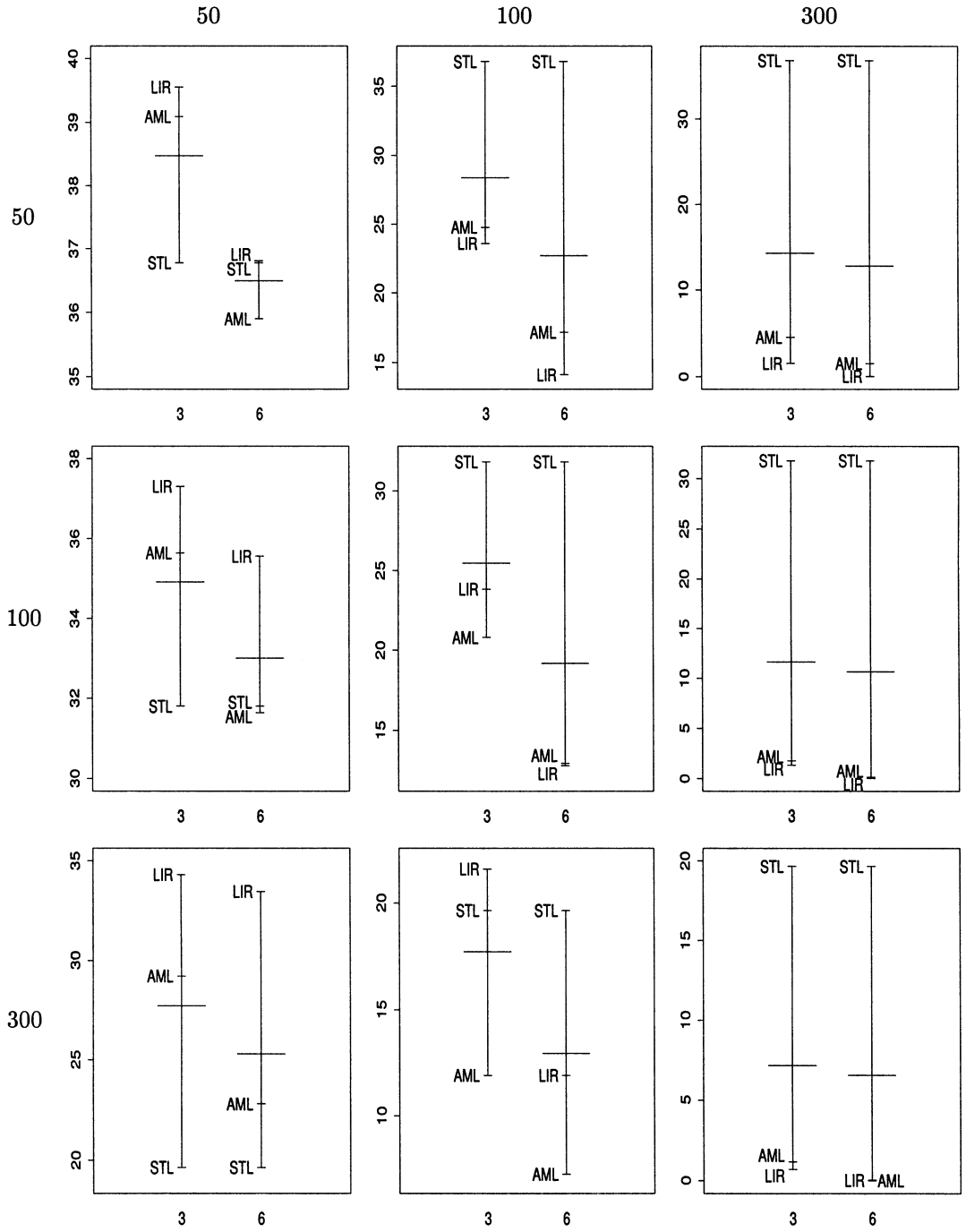


Fig. 9. The mean generalization classification errors (in %) when learning new tasks using families learned by the affine manifold learning algorithm (AML), the learning an internal representation (LIR) algorithm, and the single-task learning (STL) algorithm. In each figure, the x axis represents N (three or six). Each column corresponds to a different value of K_n (from left to right, $K_n = 50, 100,$ and 300) and each line corresponds to a different value of K_{N+1} (from top to bottom, $K_{N+1} = 50, 100,$ and 300).

learning methods. In the manifold learning method, we only expect the tasks to evolve in the same environment. No other constraints are imposed.

Considering that related tasks are those that evolve in a common environment leads to the following question: what is an environment and how can one be identified? We consider an environment to be a grouping of functions or tasks that share properties, that are affected by the same events, etc. Defining an appropriate environment needs to be done in close consultation and collaboration with an expert of the field being studied. For example, if we want to train learning models on medical data, we should consult doctors or other medical professionals. Their input is useful to define an environment.

Some have suggested that the similarity of tasks should be evaluated empirically, that is by comparing the performances of multitask learning with those of single-task learning [6]. When multitask learning outperforms single-task learning, it is concluded that the tasks are similar and when multitask learning does not outperform single-task learning, it is concluded that the tasks are not similar. We consider that such an approach

can lead to misleading conclusions and that it does not take into consideration the impact that several factors can have on the generalization performances of multitask and single-task learning. Let us consider a simple example: let (T_1, T_2, T_3, T_4, T_5, T_6) be six functions evolving in a common environment. If, when learning only the first three tasks (T_1, T_2, T_3), multitask learning leads to poor generalization performances compared to single-task learning, the empirical evaluation of similarity will force us to conclude that these three tasks are not related. Now, if all six tasks are learned simultaneously using multitask learning and if multitask learning outperforms single-task learning, we will conclude that all six tasks are similar. In the first case (i.e., learning only the first three tasks), the empirical evaluation of similarity leads us to the conclusion that these tasks are not similar, while in the second case (i.e., learning all six tasks), the empirical evaluation of similarity leads us to conclude that all six tasks are similar and therefore that the first three tasks are similar. The conclusion in the first case is, therefore, in contradiction with the conclusion of the second case. In fact, we can only conclude that there is task similarity when multitask learning outperforms single-task learning. We think that a discussion of task similarity should be divided into two different parts:

1) The process involved in the first part is to determine if tasks are similar. Task similarity should be decided based on the feedback of experts knowledgeable in the field being studied.
2) The process involved in the second part is to determine if multitask learning can discover and benefit from the similarity. As we saw in the previous sections, the answer to this topic depends on several factors like the size of the training sets used to learn a family, the number of tasks available, the choice of the multitask learning algorithm, etc. If multitask learning does not outperform single-task learning, one cannot conclude that the tasks are not similar. It might be that not all conditions necessary to obtain good generalization performances were met when applying multitask learning.

VI. Conclusion

A new multitask learning method was described that consists in learning a family of hypotheses embodying domain-related knowledge. The family is represented by a manifold or a mixture of manifolds defined in hypothesis space. The direction and location of the manifold or the mixture of manifolds are learned using training examples sampled from a group of related tasks derived from the same domain. This new approach constrains learning models trained on related tasks to select hypotheses that belong to the family.

Contrary to several multitask learning techniques that can only consider particular classes of families (i.e., particular forms of manifolds), the new approach can be used to learn a wide variety of families. No restraints are imposed on the form of manifolds that can be learned. Also, contrary to several multitask learning techniques that can only be applied to neural networks, the framework of the new approach is general enough to be applied to a wide range of learning models.

Experiments were performed to analyze the performances of the new method on a class of learning tasks. A statistical analysis showed that this method significantly outperforms single-task learning. When compared to other multitask learning methods, the new approach is either better or at least comparable. Results were also presented that illustrate the usefulness of the family learned by the new method when learning new tasks derived from the same domain. An analysis showed that the new method outperforms single-task learning in most learning contexts and that it is either better or at least comparable to another multitask learning method.

References

[1] Y. S. Abu-Mostafa, "Hints and the VC dimension," *Neural Comput.*, vol. 5, no. 2, pp. 278–288, 1993.

[2] J. Baxter, "Learning model bias," in *Advances in Neural Information Processing Systems*, M. Mozer, D. Touretzky, and M. Perrone, Eds. Cambridge, MA: MIT Press, 1996, vol. 8, pp. 169–175.

[3] ——, "A Bayesian/information theoretic model of learning to learn via multiple task sampling," *Machine Learning*, vol. 28, pp. 7–40, 1997.

[4] H. Bensusan, "Odd bites into bananas don't make you blind—Learning about simplicity and attribute addition," in *Proc. Eur. Conf. Machine Learning (ECML) Workshop Upgrading Learning Meta-Level*, 1998, pp. 30–42.

[5] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.

[6] ——, "Multitask Learning," Ph.D. dissertation, School Comput. Sci., Carnegie Mellon Univ., 1997.

[7] W. W. Cohen, "Compiling prior knowledge into an explicit bias," in *Proc. 9th Int. Machine Learning Workshop*, Aberdeen, U.K., 1992, pp. 102–110.

[8] P. Datta and D. Kibler, "Concept-sharing: A means to improve multi-concept learning," in *Proc. 10th Int. Conf. Machine Learning*, 1993, pp. 89–96.

[9] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, no. 1, pp. 1–58, Jan. 1992.

[10] J. Ghosn and Y. Bengio, "Multi-task learning for stock selection," in *Advances in Neural Information Processing Systems*, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, 1997, vol. 9, pp. 946–952.

[11] ——, "Bias leaning, knowledge sharing," in Proc. Int. Joint Conf. Neural Networks, 2000, pp. 9–14.

[12] T. Heskes, "Solving a huge number of similar tasks: A combination of multi-task learning and a hierarchical Bayesian approcah," presented at the Int. Conf. Machine Learning, 1998.

[13] N. Intrator and S. Edelman, "How to make a low-dimensional representation suitable for diverse tasks," Connection Sci. (Special Issue on Transfer in Neural Networks), 1996.

[14] K. J. Lang, A. H. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural Networks*, vol. 3, pp. 23–44, 1990.

[15] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*, D. Touretzky, Ed. Cambridge, MA: MIT Press, 1990, vol. 2, pp. 396–404.

[16] P. McCullagh and J. Nelder, *Generalized Linear Models*. London, U.K.: Chapman and Hall/CRC, 1999.

[17] T. Mitchell, R. Caruana, D. Freitag, J. M. Dermott, and D. Zabowski, "Experience with a learning personal assistant," in Commun. ACM, July 1994.

[18] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural Computation*, vol. 4, pp. 473–493, 1992.

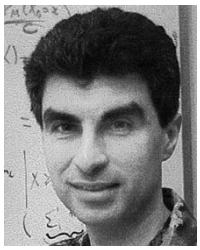
[19] S. M. Omohundro, "Family discovery," in *Advances in Neural Information Processing Systems*, M. Mozer, D. Touretzky, and M. Perrone, Eds. Cambridge, MA: MIT Press, 1996, vol. 8.

[20] D. Ourston and R. J. Mooney, "Improving shared rules in multiple category domain theories," in *Proc. 8th Int. Machine Learning Workshop*, Evanston, IL, June 1991, pp. 534–538.

[21] L. Y. Pratt and V. I. Gough, "Improving discriminability based transfer by modifying the IM metric to use sigmoidal activations," in *Cybernetics and Systems*, R. Trappl, Ed. Singapore: World Scientific, 1994, pp. 1719–1726.
[22] J. Schmidhuber, J. Zhao, and M. Wiering, "Shifting inductive bias with success-story algorithm, adaptive Levin seach, and incremental self-improvement," *Machine Learning*, vol. 28, pp. 105–130, 1997.
[23] J. Sill and Y. S. Abu-Mostafa, "Monotonicity hints," in *Advances in Neural Information Processing Systems*, vol. 9, 1997, pp. 634–640.
[24] D. L. Silver and R. E. Mercer, "The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness," *Connection Sci. (Special Issue on Transfer in Inductive Systems)*, vol. 8, no. 2, pp. 277–294, 1996.
[25] D. L. Silver, R. E. Mercer, and G. A. Hurwitz, "The functional transfer of knowledge for coronary artery disease diagnosis," Dept. Comput. Sci., Univ. Western Ontario, London, ON, Canada, Tech. Rep. 513, 1997.
[26] S. Thrun and J. O'Sullivan, "Learning to learn," in *Chapter Clustering Learning Tasks and the Selective Cross-Transfer of Knowledge*. Boston, MA: Kluwer, 1998.
[27] S. L. Zeger and K.-Y. Liang, "Longitudinal data analysis for discrete and continuous outcomes," *Biometrics*, vol. 42, pp. 121–130, 1986.

Joumana Ghosn is a Ph.D. student with the Université de Montréal, Montréal, QC, Canada.

She was a Research Assistant with the NEC Research Institute, Princeton, NJ, during the summers of 1995, 1996, and 1997. In 2000 and 2001, she collaborated with the vision group of Ecole Polytechnique de Montréal and Sainte Justine hospital, Montréal, QC, Canada. Her research interests include machine learning algorithms and pattern recognition problems.



Yoshua Bengio received the Ph.D. degree in computer science from McGill University, Montréal, QC, Canada, in 1991.

After two postdoctoral years, one at Massachusetts Institute of Technology, Cambridge, and one at AT&T Bell Laboratories, Middletown, NJ, he became Professor at the Department of Computer Science and Operations Research at Université de Montréal, Montréal, QC, Canada. He is the author of a book and more than 100 articles, the most cited being in the areas of recurrent neural networks, hidden Markov models/neural network hybrids, and convolutional neural networks. Since 2000, he has held a Canada Research Chair in Statistical Learning Algorithms.