Learning One More Thing

Sebastian Thrun
Universität Bonn
Institut für Informatik III
Römerstr. 164, D–53117 Bonn, Germany

Tom M. Mitchell
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA

Abstract

Most research on machine learning has focused on scenarios in which a learner faces a single, isolated learning task. The lifelong learning framework assumes that the learner encounters a multitude of related learning tasks over its lifetime, providing the opportunity for the transfer of knowledge among these. This paper studies lifelong learning in the context of binary classification. It presents the invariance approach, in which knowledge is transferred via a learned model of the invariances of the domain. Results on learning to recognize objects from color images demonstrate superior generalization capabilities if invariances are learned and used to bias subsequent learning.

1 Introduction

Supervised learning is concerned with learning an unknown target function from a finite collection of input-output examples of that function. Formally, the framework of supervised learning can be characterized as follows. Let \( f \) denote the set of all target functions. For example, in a robot arm domain, \( F \) might be the set of all kinematic functions for robots with three joints. Every function \( f \in F \) is defined over the same input space, denoted by \( I \), and the same output space, denoted by \( O \). The learner has a set of hypotheses that it can consider, denoted by \( H \), which might or might not be different from \( F \). For example, the set \( H \) could be the set of all artificial neural networks with 20 hidden units, or, alternatively, the set of all decision trees with depth less than 10. Throughout this paper, we make the simplifying assumption that all functions in \( F \) are binary classifiers, \( i.e. \), \( O = \{0, 1\} \). We will refer to instances that fall into class 1 as positive instances, and to those that fall into class 0 as negative instances.

To learn an unknown target function \( f^* \in F \), the learner is given a finite collection of input-output examples (training examples)

\[
X = \{ (i, f^*(i)) \}, \tag{1}
\]

which are possibly distorted by noise. The goal of the learner is to generate a hypothesis \( h \in H \), such that the deviation (error)

\[
E = \sum_{i \in \text{train}} \text{Prob}(i) \| f^*(i) - h(i) \| \tag{2}
\]

between the target function \( f^* \) and \( h \) on future examples will be as small as possible. Here \( \text{Prob} \) is the probability distribution according to which the training examples are generated. \( \text{Prob} \) is generally unknown to the learner, as is \( f^* \).

Standard supervised learning focuses on learning a single target function \( f^* \), and training data is assumed to be available only for this one function. However, if functions in \( F \) are appropriately related, it can be helpful to have access to training examples of other functions \( f \in F \) as well. For example, consider a robot whose task is to find and fetch various objects, using its camera for object recognition. Let \( F \) be the set of recognition \( (i.e., \) classification) functions for all objects, one for each potential target object, and let the target function \( f^* \in F \) correspond to an object the robot must learn to recognize. \( X \), the training set, will consist of positive and negative examples of this object. The task of the learner is to find an \( h \) which minimizes \( E \). In particular, the robot should learn to recognize the target object invariant of rotation, translation, scaling in size, change of lighting and so on. Intuitively speaking, the more profound the learner’s initial understanding of these invariances, the fewer training examples it will require for reliable learning. Because these invariances are common to all functions in \( F \), images showing other objects can provide additional information and hence support learning \( f^* \).

This example illustrates the idea of lifelong learning. In lifelong learning, a collection of related learning problems is encountered over the lifetime of the learner. When learning the \( n \)-th task, the learner may employ knowledge gathered in the previous \( n - 1 \) tasks to improve its performance [Thrun and Mitchell, to appear].

This paper considers a particular form of lifelong learning in which the learning tasks correspond to learning boolean classifications \( (\text{concepts}) \), and in which the previous experience consists of training examples of other classification functions from the same family \( F \). More formally, in addition to the set of training examples \( X \) for the target function \( f^* \), the learner is also provided \( n - 1 \) sets of examples

\[
X_k = \{ (i, f_k(i)) \} \quad (k \in \{k_1, k_2, \ldots, k_{n-1}\})
\]

with \( k_j \in \{1, 2, \ldots, |F|\} \)

\[
\forall j \in \{1, 2, \ldots, n - 1\} \tag{3}
\]

of other functions \( \{f_{k_1}, f_{k_2}, \ldots, f_{k_{n-1}}\} \subseteq F \) taken from the same function family \( F \). Since this additional data can support learning \( f^* \), we shall call each \( X_k \) a support set for \( X \). The set
Given:
- a space of hypotheses \( H : I \rightarrow O \)
- a set of training examples \( X = \{(i, f^*(i))\} \) of some unknown target function \( f^* \in F \), drawn with probability distribution \( \text{Prob} \).
- in lifelong supervised learning: a collection of support sets \( Y = \{X_k\} \), which characterize other functions \( f_k \in F \). Here \( X_k = \{(i, f_k(i))\} \).

determine:
- a hypothesis \( h \in H \) that minimizes
\[
\sum_{i \in I} \text{Prob}(i) \left| f^*(i) - h(i) \right|
\]

Table 1: Standard and lifelong supervised learning.

of available support sets for \( X \), \( \{X_k|k = k_1, k_2, \ldots, k_{n-1}\} \), will be denoted by \( Y \). Notice that the input-output examples in the support sets \( X_k \) may have been drawn from different probability distributions.

Support sets can be useful in a variety of real-world scenarios. For example, Sinha [Sinha, 1994] and Lando/Edelman [Lando and Edelman, 1995] have proposed approaches that improve the recognition rate of human faces based on knowledge learned by analyzing different views of other, related faces. In speaker-dependent approaches to speech recognition, learning to recognize personal speech is often done by speaker adaptation methods. Speaker adaptation simplifies the learning task by using knowledge learned from other, similar speakers (e.g., see [Hild and Waibel, 1993]). Other approaches that use related functions to change the bias of an inductive learner can be found in [Utgoff, 1986], [Rendell et al., 1987], [Suddarth and Kergosien, 1990], [Moore et al., 1992], [Sutton, 1992], [Caruana, 1993], and [Pratt, 1993].

Table 1 summarizes the problem definitions of standard supervised learning and the lifelong supervised learning problem. In lifelong supervised learning, the learner is given a collection \( Y \) of support sets, in addition to the training set \( X \) and the hypothesis space \( H \). This raises two fundamental questions:

1. How can a learner use support sets to generalize more accurately?
2. Under what conditions will a learner benefit from support sets?

This paper does not provide general answers to these questions. Instead, it proposes one particular approach, namely learning invariance functions, which relies on certain assumptions regarding the function set \( F \). It also presents empirical evidence that this approach to using support sets can significantly improve generalization accuracy when learning to recognize objects based on visual data.

2 The Invariance Approach

The invariance approach first learns an invariance function \( \sigma \) from the support sets in \( Y \). This function is then used to bias the learner as it selects a hypothesis to fit the training examples \( X \) of the target function \( f^* \).

2.1 Invariance Functions

Let \( Y = \{X_k\} \) be a collection of support sets for learning \( f^* \). Recall our assumption that all functions in \( F \) have binary output values. Hence, each example in a support set is either positive (i.e., output 1) or negative (i.e., output 0). Consider a target function, \( f_k \in F \) with \( k \in \{1, \ldots, |F|\} \), and a pair of examples, say \( i \in I \) and \( j \in I \). A local invariance operator \( \tau_k : I \times I \rightarrow \{0, 1\} \) is a mapping from a pair of input vectors defined as follows:

\[
\tau_k(i, j) = \begin{cases} 
1 & \text{if } f_k(i) = f_k(j) = 1 \\
0 & \text{if } f_k(i) \neq f_k(j) \\
\text{not defined} & \text{if } f_k(i) = f_k(j) = 0
\end{cases}
\]

The local invariance operator indicates whether both instances are members of class 1 (positive examples) relative to \( f_k \). If \( \tau_k(i, j) = 1 \), then \( f_k \) is invariant with respect to the difference between \( i \) and \( j \). Notice that positive and negative instances of \( f_k \) are not treated symmetrically in the definition of \( \tau \).

The local invariance operators \( \tau_k (k = 1, \ldots, |F|) \) define a (global) invariance function for \( F \), denoted by \( \sigma : I \times I \rightarrow \{0, 1\} \). For two examples, \( i \) and \( j \), \( \sigma(i, j) \) is 1 if there exists a \( k \) for which \( \tau_k(i, j) = 1 \). Likewise, \( \sigma(i, j) \) is 0 if there exists a \( k \) for which \( \tau_k(i, j) = 0 \):

\[
\sigma(i, j) = \begin{cases} 
1, & \text{if } \exists k \in \{1, \ldots, |F|\} \text{ with } \tau_k(i, j) = 1 \\
0, & \text{if } \exists k \in \{1, \ldots, |F|\} \text{ with } \tau_k(i, j) = 0 \\
\text{not defined}, & \text{otherwise}
\end{cases}
\]

The invariance function \( \sigma \) behaves like an invariance operator, but it does not depend on \( k \). It is important to notice that the invariance function can be ill-defined. This is the case if there exist two examples which both belong to class 1 under one target function, but which belong to different classes under a second target function:

\[
\exists i, j \in I, k, k' \in \{1, \ldots, |F|\} : \tau_k(i, j) = 1 \land \tau_{k'}(i, j) = 0
\]

In such cases the invariance mapping is ambiguous and is not even a mathematical function. A class of functions \( F \) is said to obey the invariance property if its invariance function is non-ambiguous\(^1\). The invariance property is a central assumption for the invariance approach to lifelong classification learning.

The concept of invariance functions is quite powerful. Suppose \( F \) holds the invariance property. If \( \sigma \) is known, every training instance \( i \) for an arbitrary function \( f_k \in F \) can be correctly classified, given there is at least one positive instance of \( f_k \) available. To see, assume \( i_{pos} \in I \) is known to be a positive instance for \( f_k \). Then for any instance \( i \in I \), \( \sigma(i, i_{pos}) \) will be 1 if and only if \( f_k(i) = 1 \). Although the invariance property imposes a restriction on the function family \( F \), it holds true for quite a few real-world problems, such as those typically studied in character recognition, speech understanding, and various other domains.

For example, a function family obeys the invariance property if all positive classes (of all functions \( f_k \) are disjoint. One such function family is the family of object recognition functions defined over distinct objects.

2.2 Learning the Invariants

In the lifelong learning regime studied in this paper, \( \sigma \) is not given. However, an approximation to \( \sigma \), denoted by \( \sigma \), can be...
be learned. Since $\sigma$ does not depend upon the specific target function $f^*$, every support set $X_k \in Y$ can be used to train $\hat{\sigma}$, as long as there is at least one positive instance available in $X_k$. For all $k \in \{1, \ldots, |Y|\}$, training examples for $\sigma$ are constructed from examples $i, j \in X_k$:

$$((i, j), \tau_k(i, j))$$

Here $\tau_k$ must be defined, i.e., at least one of the examples $i$ and $j$ must be positive under $f_k$. In the experiments described below, $\sigma$ is approximated by training an artificial neural network using the Backpropagation algorithm.

The invariance network, once learned, can be used in conjunction with a training set $X$ to infer values for $f^*$. Let $X_{\text{pos}} \subset X$ be the set of positive training examples in $X$. Then for any $i_{\text{pos}} \in X_{\text{pos}}, \hat{\sigma}(i, i_{\text{pos}})$ estimates $f^*(i)$ for $i \in I$. If this estimate is interpreted as a probability (for the event that $i$ is positive under $f^*$), Bayes’ rule can be applied

$$\text{Prob}(f^*(i)=1) = 1 - \left(1 + \prod_{i_{\text{pos}} \in X_{\text{pos}}} \frac{\hat{\sigma}(i, i_{\text{pos}})}{1-\hat{\sigma}(i, i_{\text{pos}})}\right)^{-1}$$

(4)

Notice that in this approach, $\hat{\sigma}$ is similar to a distance metric that is obtained from the support sets [Moore et al., 1992; Baxter, 1995]. The invariance network $\hat{\sigma}$ generalizes the notion of a distance metric, because the triangle inequality need not hold, and because an instance $i_{\text{pos}}$ can provide evidence that $i$ is member of the opposite class (if $\hat{\sigma}(i, i_{\text{pos}}) < 0.5$).

In general $\hat{\sigma}$ might not be accurate enough to describe $f^*$ correctly. This may be because of modeling limitations, noise, or lack of training data. We will therefore describe an alternative approach to the lifelong learning problem that employs the invariance network, which has been found empirically to generalize more accurately.

### 2.3 Extracting Slopes to Guide Generalization

The remainder of this section describes a hybrid neural network learning algorithm for learning $f^*$. This algorithm is a special case of both the Tangent-Prop algorithm [Simard et al., 1992] and the explanation-based neural network learning (EBNN) algorithm [Mitchell and Thrun, 1993]. Here we will refer to it as EBNN.

Suppose we are given a training set $X$, and an invariance network $\hat{\sigma}$ that has been trained using a collection of support sets $Y$. We are now interested in learning $f^*$. One could, of course, ignore the invariance network and the support sets altogether and train a neural network purely based on the training data $X$. The training set $X$ imposes a collection of constraints on the output values for the hypothesis $h$. If $h$

1. Let $X_{\text{pos}} \subset X$ be the set of positive training examples in $X$.
2. Let $X' = \emptyset$
3. For each training example $(i, f^*(i)) \in X_{\text{pos}}$ do:

   a. Compute $\nabla_i \hat{\sigma}(i) = \frac{1}{|X_{\text{pos}}|} \sum_{i_{\text{pos}} \in X_{\text{pos}}} \frac{\partial \hat{\sigma}(i)(i_{\text{pos}})}{\partial i}$ using the invariance network $\hat{\sigma}$.

   b. Let $X' = X' \cup (i, f^*(i), \nabla_i \hat{\sigma}(i))$
4. Fit $X'$.

Table 2: Application of EBNN to learning with invariance networks.

is represented by an artificial neural network, as is the case in the experiments reported below, the Backpropagation (BP) algorithm can be used to fit $X$.

EBNN does this, but it also derives additional constraints using the invariance network. More precisely, in addition to the value constraints in $X$, EBNN derives constraints on the slopes (tangents) for the hypothesis $h$. To see how this is done, consider a training example $i$, taken from the training set $X$. Let $i_{\text{pos}}$ be an arbitrary positive example in $X$. Then, $\hat{\sigma}(i, i_{\text{pos}})$ determines whether $i$ and $i_{\text{pos}}$ belong to the same class—information that is readily available, since we are given the classes of $i$ and $i_{\text{pos}}$. However, predicting the class using the invariance network also allows us to determine the output-input slopes of the invariance network. These slopes measure the sensitivity of class membership with respect to the input features in $i$. This is done by computing the partial derivative of $\hat{\sigma}$ with respect to $i$ at $(i, i_{\text{pos}})$:

$$\nabla_i \hat{\sigma}(i) := \frac{\partial \hat{\sigma}(i, i_{\text{pos}})}{\partial i}$$

$\nabla_i \hat{\sigma}(i)$ measures how infinitesimal changes in $i$ will affect the classification of $i$. Since $\hat{\sigma}(i, i_{\text{pos}})$ is an approximation to $f^*$, $\nabla_i \hat{\sigma}(i)$ approximates the slope $\nabla_i f^*(i)$. Consequently, instead of fitting training examples of the type $(i, f^*(i))$, EBNN fits training examples of the type

$$(i, f^*(i), \nabla_i f^*(i))$$

Gradient descent can be used to fit training examples of this type, as explained in [Simard et al., 1992]. Fig. 1 illustrates the utility of this additional slope information in function fitting.

Notice if multiple positive instances are available in $X$, slopes can be derived from each one. In this case, averaged slopes are used to constrain the target function:

$$\nabla_i \sigma(i) := \frac{1}{|X_{\text{pos}}|} \sum_{i_{\text{pos}} \in X_{\text{pos}}} \frac{\partial \sigma(i, i_{\text{pos}})}{\partial i}$$

(5)

Here $X_{\text{pos}} \subset X$ denotes the set of positive examples in $X$. The application of the EBNN algorithm to learning with invariance networks is summarized in Table 2.

Generally speaking, slope information extracted from the invariance network is a linear approximation to the variances and invariances of $F$ at a specific point in $I$. Along the invariant directions slopes will be approximately zero, while along others they will be large. For example, in the aforementioned
find-and-fetch tasks, it might happen that color is an important feature for classification while brightness is not. This is typically the case in situations with changing illumination. In this case, the invariance network could learn to ignore brightness, and hence the slopes of its classification with respect to brightness would be approximately zero. The slopes for color, however, would be large, given that slight color changes imply that the object would belong to a different class.

When training the classification network, slopes provide additional information about the sensitivity of the target function with respect to its input features. Hence, the invariance network can be said to bias the learning of the classification network. However, since EBNN trains on both slopes and values simultaneously, errors in this bias (incorrect slopes due to approximations in the learned invariance network) can be overturned by the observed training example values in $X$. The robustness of EBNN to errors in estimated slopes has been verified empirically in a robot navigation domain [Mitchell and Thrun, 1993].

3 Example

3.1 The Domain: Object Recognition

To illustrate the transfer of knowledge via the invariance network, we collected a database of 700 color camera images of seven different objects (100 images per object), as depicted in Fig. 2 (left columns).

<table>
<thead>
<tr>
<th>Object</th>
<th>color</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>bottle</td>
<td>green</td>
<td>medium</td>
</tr>
<tr>
<td>hat</td>
<td>blue and white</td>
<td>large</td>
</tr>
<tr>
<td>hammer</td>
<td>brown and black</td>
<td>medium</td>
</tr>
<tr>
<td>can</td>
<td>red</td>
<td>medium</td>
</tr>
<tr>
<td>book</td>
<td>yellow</td>
<td>depending on perspective</td>
</tr>
<tr>
<td>shoe</td>
<td>brown</td>
<td>medium</td>
</tr>
<tr>
<td>glasses</td>
<td>black</td>
<td>small</td>
</tr>
</tbody>
</table>

The objects were chosen so as to provide color and size cues helpful to their discrimination. The background of all images consisted of plain, white cardboard. Different images of the same object varied by the relative location and orientation of the object within the image. In 50% of all snapshots, the location of the light source was also changed, producing bright reflections at random locations in various cases. In some of the images the objects were back-lit, in which case they appeared to be black. Fig. 3 shows examples of two of the objects, the shoe and the glasses.

Images were encoded by a 300-dimensional vector, providing color, brightness and saturation information for a down-scaled image of size 10 by 10. Examples for the down-scaled images are shown in Figures 2 (right columns) and 3. Although each object appears to be easy to recognize from the original image, in many cases we found it difficult to visually classify objects from the subsampled images. However, subsampling was necessary to keep the networks to a reasonable size.

The set of target functions, $F$, was the set of functions that recognize objects, one for each object. For example, the indicator function for the bottle, $f_{\text{bottle}}$, was 1, if the image showed a bottle, and 0 otherwise. Since we only presented distinct objects, all sets of positive instances were disjoint. Consequently, $F$ obeyed the invariance property. The set of hypotheses $H$ was the set of all artificial neural networks with 300 input units, 6 hidden units, and 1 output unit, as such a network was employed to represent the target function.

The objective was to learn to recognize shoes, i.e., $f^* = f_{\text{shoe}}$. Five other objects, namely the bottle, the hat, the hammer, the can and the book, were used to construct the support sets $Y$. To avoid any overlap in the training set $X$ and the support sets in $Y$, we exclusively used pictures of a seventh object, glasses, as counterexamples for $f_{\text{shoe}}$. Each of the five support sets in $Y$, $X_{\text{bottle}}$, $X_{\text{hat}}$, $X_{\text{hammer}}$, $X_{\text{can}}$ and $X_{\text{book}}$, contained 100 images of the corresponding object (positive examples) and 100 randomly selected images of other objects (negative examples). When constructing training examples for the invariance network, we randomly selected a subset of 1,000 pairs of images, 800 of which were used for training and 200 for cross-validation. 50% of the final training and cross-validation examples were positive examples for the in-