# Zero-data Learning of New Tasks

**Hugo Larochelle** and **Dumitru Erhan** and **Yoshua Bengio**
Université de Montréal
Montréal, Québec
{larocheh,erhandum,bengioy}@iro.umontreal.ca

## Abstract

We introduce the problem of zero-data learning, where a model must generalize to classes or tasks for which no training data are available and only a description of the classes or tasks are provided. Zero-data learning is useful for problems where the set of classes to distinguish or tasks to solve is very large and is not entirely covered by the training data. The main contributions of this work lie in the presentation of a general formalization of zero-data learning, in an experimental analysis of its properties and in empirical evidence showing that generalization is possible and significant in this context. The experimental work of this paper addresses two classification problems of character recognition and a multi-task ranking problem in the context of drug discovery. Finally, we conclude by discussing how this new framework could lead to a novel perspective on how to extend machine learning towards AI, where an agent can be given a specification for a learning problem before attempting to solve it (with very few or even zero examples).

## Introduction

Machine learning research has helped develop tools that can tackle ever more challenging problems. From the standard *supervised learning* setting, where all training examples are tagged with their desired target output, many solutions adapted to other settings have been developed and studied so far. These settings are usually characterized by limited labeled training data. Recognizing that labeled data is often scarce but unlabeled data usually is not, *semi-supervised learning* algorithms (Chapelle, Schölkopf, & Zien 2006) were proposed to make the most out of that situation. Also, *transfer learning* or *multi-task learning* (Caruana 1997) was proposed and developed so that labeled training sets from different but related tasks could be grouped together in order to improve performance for all these tasks, compared to solving them separately. *Life-long learning* (Thrun 1996) addresses a similar problem where a sequence of tasks must be learned one after the other. *One-shot learning* (Miller 2002) attempts to solve the problem of recognizing a particular object from only one labeled example, also by using labeled data from other categories of related objects. More recently, *self-taught learning* (Raina *et al.* 2007) algorithms were described for situations that combine the challenges of both semi-supervised and transfer learning, i.e. unlabeled data from a different but related classification problem must be leveraged. One motivation behind these different types of learning is that they move machine learning solutions closer to the sort of learning humans are capable of, making possibly small but significant steps towards artificial intelligence. Actually, these types of learning are often inspired by intuitions or observations about human learning. For instance, it seems reasonable to think that humans are able to transfer what they have learned between different but similar tasks (multi-task learning). There is also empirical evidence that humans are able to recognize objects (such as faces) with high accuracy (Moses, Ullman, & Edelman 1996) even after having been exposed to only one instance of that object (one-shot learning). Moreover, most of the stimuli humans receive are not labeled by an expert, which would mean that humans are able to leverage unlabeled stimuli, a fact that has been validated by experimental data (Zhu *et al.* 2007).

The work presented in this paper shares the same motivations. We introduce *zero-data learning*, which aims at solving classification problems where there is not enough labeled training data to cover all the classes to discriminate, or multi-task problems for which the available training data does not provide examples of desired outputs for some of the tasks to solve. However, it is assumed that descriptions of the classes or tasks are available, which will be used to compensate for the lack of labeled data. Though this situation can simply occur because there is little training data to begin with, it can also come up because the number of classes or tasks is too large. The latter case is particularly relevant in the context of hard problems associated with AI, such as vision problems. For example, the number of objects that a human is able to discriminate has been estimated to up to 30,000 (Bierderman 1987). Creating a database of labeled data with a sufficient amount of instances for all these categories would be a very laborious task. Zero-data learning is also appropriate when novel classes or tasks are introduced without training data and one still has to derive a solution for them, possibly until training data is collected (one can use this as a first step for *active learning*). This particular use of zero-data learning means that one can *communicate* a description of the task that the machine is expected to perform, by means other than a training set. For example,

given a set of image classes (e.g. handwritten English words) and labeled samples coming from these classes, it is easy to conceive a representation of each class (e.g. as a sequence of character symbols) that can also describe *new classes* of images to detect (e.g. out-of-vocabulary words) in order to build a *new classifier* for these classes, even before seeing training examples for these particular classes.

Zero-data learning attempts to solve a hard problem, but not an impossible one. When it comes to recognizing classes of patterns or objects, a general description of these classes in terms of smaller components shared by these classes can provide a lot of information, possibly worth more than examples of instances from these classes. For example, given a description of one of the many thousands Chinese characters in terms of the sequence of pen strokes that produces it, we can expect someone to be able to recognize handwritten versions of that character with reasonable accuracy. Similarly, a description decomposing a particular object in more abstract shapes (cylinders, spheres, cubes, etc.) can be sufficient for someone to identify a real world version of that object. This generalization without actual training data is possible because the knowledge about how to relate these descriptions to the decision making process that permits to solve these problems has been acquired.

In this paper, we formalize the problem of zero-data learning and show empirically that generalization is possible in that context.

## Zero-data Learning

Zero-data learning corresponds to learning a problem for which no training data are available for some classes or tasks and only descriptions of these classes/tasks are given. The description of each class or task is provided in some representation, the simplest being a vector of numeric or symbolic attributes. Though classification and multi-task problems are not exactly the same, we will use a common notation for simplicity. The notation is illustrated in Figures 1 and 2, which display how it is used in the contexts of classification and multi-task problems. We note $x_t \in \mathcal{X}^n$ for the $t^{th}$ input vector of a data set, $z$ for the identity of a class or task, with $d(z)$ its representation, and $y_t^z$ as the desired output (target) for class/task $z$ when the input is $x_t$.

For example, in a classification setting, the desired output $y_t^z$ would be 1 if sample $x_t$ belongs to class $z$ and 0 otherwise. Figure 1 shows an illustration of such a classification problem with images of characters. Each row corresponds to a sample, and column $j$ corresponds to a 1-vs-rest classification problem involving class $j$ against some other classes. The small 7 by 5 pixels images on each column correspond to the task description vectors $d(z)$. Hyphens correspond to missing values. In the multi-task setting (see Figure 2), the desired output is simply the value of the dependent variable associated with $x_t$ for task $z$.

When evaluating zero-data learning algorithms, we wish to measure the extent to which they can compensate for the lack of labeled data by using the description vectors. To accomplish this, we separate the training data $\mathcal{D}_{train}$ and the test data $\mathcal{D}_{test}$ in such a way that the training and test sets contain labeled data from different classes or tasks. In the multi-task setting of zero-data learning, the same input can appear in the data of a training task and of a test task, since the same input can be paired with more than one task.

## Related Work

We already mentioned the various types of learning settings which compensate for the lack of labeled data. Zero-data learning shares some similarities with one-shot learning which can be seen as a particular case of zero-data learning where the description of a class is a prototypical sample from that class in $\mathcal{X}^n$. It is also related to lifelong learning, for which a sequence of tasks have to be learned, each task being associated to its own training set. However, in zero-data learning, when learning a new task, the algorithm uses a description of that task instead of a training set. Moreover, class or task descriptions (features) have been investigated before (Bakker & Heskes 2003; Bonilla, Agakov, & Williams 2007; Tsochantaridis *et al.* 2005), but never in a setting where a task had no training data at all.

Finally, though zero-data learning has never been investigated as a general learning framework, it has been tackled in particular application settings before. In the context of recommender systems, it is known as the "cold start" problem (Schein *et al.* 2002) where a task corresponds to an object to recommend.

## Views of the Problem

There are at least two approaches to zero-data learning, which correspond to two different views of the problem. We refer to these approaches as the **input space view** and the **model space view**.

### Input Space View

The input space view considers the input of the learning problem to be the concatenation $[x_t, d(z)]$ of the input $x_t$ with the corresponding class/task description $d(z)$. In this framework, any supervised learning algorithm can then be used to train a model that will give predictions for the value of $y_t^z$. The procedure to follow is simple:

1. For each $(x_t, y_t^z) \in \mathcal{D}_{train}$, produce new sample $([x_t, d(z)], y_t^z)$
2. Use these samples as the training set for a chosen learning algorithm to obtain trained model $f^*(\cdot)$

The prediction or output for a new class or task $z^*$ and a sample $x^*$ is then simply $f^*([x, d(z^*)])$. Also, in a classification setting where we want to discriminate between the classes $z_1^*$ and $z_2^*$, one can ask for their outputs and choose the largest. This approach is really practical only if all $y^z$ are of the same type (e.g. scalars or vectors of the same length).

In our experiments with the input space view, we used SVM classifiers with Gaussian and polynomial kernels. The case of the Gaussian kernel actually corresponds to a particular case of the structured output framework using output features, presented by Tsochantaridis *et al.*.

| $z \rightarrow$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $d(z) \rightarrow$ | 1 | 2 | 3 | A | B |
| $x_t$ | $y_t^1$ | $y_t^2$ | $y_t^3$ | $y_t^4$ | $y_t^5$ |
| 1 | 1 | 0 | 0 | - | - |
| 2 | 0 | 1 | 0 | - | - |
| 3 | 0 | 0 | 1 | - | - |
| A | - | - | - | 1 | 0 |
| B | - | - | - | 0 | 1 |
| | | training data | | test data | |

Figure 1: Notation in 1-vs-rest classification, with test data corresponding to classes not seen at training time.

| $z \rightarrow$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $d(z) \rightarrow$ | 1 | 2 | 3 | A | B |
| $x_t$ | $y_t^1$ | $y_t^2$ | $y_t^3$ | $y_t^4$ | $y_t^5$ |
| 1 | 3 | - | -1 | 0.5 | - |
| 2 | 2.5 | 1 | - | - | - |
| 3 | -2 | 3 | - | 0.25 | 2 |
| A | -1 | 1 | - | -2 | -3 |
| B | 1 | - | 1.5 | 3.5 | 4 |
| | | training data | | test data | |

Figure 2: Notation for multi-task learning, with real-valued targets. The '-' entries are missing values.

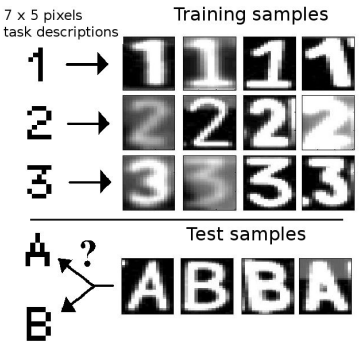


Figure 3: Illustration of zero-data classification for a character recognition problem.

## Model Space View

The model space view considers the model $f_z(\cdot)$ for a class or task $z$ to be a *function* of its description $d(z)$, i.e.

$$f_z(x) = g_{d(z)}(x).$$

For example, $g_{d(z)}(x)$ could be a linear classifier for which the weights are a function of $d(z)$. To train the "model of models" $g_{d(z)}(x)$, one can simply optimize its parameters to minimize the average cost over input/target pairs in the training set (where a different pair can exist for the same input and different classes/tasks):

$$\frac{1}{|\mathcal{D}_{train}|} \sum_{x_t, y_t^z} C(y_t^z, g_{d(z)}(x_t)).$$

This approach is very similar to the multi-task learning approach taken by Ghosn & Bengio, which propose to learn simultaneously a family of models (which would be $g_{d(z)}(\cdot)$ here) and the coordinates $d(z)$ on the manifold of models induced by the parametrized family of models. The main difference here is that we assume that the coordinates $d(z)$ for the different tasks are provided rather than learned, which is essential to be able to generalize to a zero-data task.

For a classification problem, one can easily come up with a model space view model by first defining a joint distribution $p(x, d(z))$ and then setting $g_{d(z)}(x) = p(x|d(z))$, where $x$ is the input and $d(z)$ is the representation of the class associated to $x$. However, this is not the only way to define $g_{d(z)}(x)$. One can simply choose a family of functions $h_\theta(x)$ parametrized by $\theta$ and arbitrarily define a function $q(d(z))$ whose output space is the same as the parameter space of $\theta$. Then, the model space view model is obtained by setting $g_{d(z)}(x) = h_{q(d(z))}(x)$.

This view is more general than the input space view and hence might give rise to more interesting models. Indeed, the input space view simply corresponds to setting $g_{d(z)}(x) = g([x, d(z)])$. The main difference between the input space view and the model space view is that the latter explicitly discerns the input from the class/task description, whereas the former ignores this information and asks that the training algorithm figures it out automatically. We believe that this is a crucial distinction and that it best explains the difference in the performance of the models from both views.

## Simple Models from the Model Space View

We developed and tested three linear models using the model space view. The first one is a *generative* model for classification, which considers that $X \sim \mathcal{N}(A\, d(z), \Sigma)$ when $X$ belongs to class $z$, where $\Sigma$ is diagonal. To lighten notation we omit the bias term for the Gaussian mean, equivalently represented by a constant component of value 1 in the representations $d(z)$. We call this model **LinGen**.

To learn this model, we can simply compute the value of $A$ and $\Sigma$ that maximizes the likelihood of the inputs $x_t$ given their corresponding classes. There is a simple analytic solution to this maximum likelihood optimization problem:

$$\hat{A} = \left( \sum_{t=1}^{T} x_t d(z_t)' \right) \left( \sum_{t=1}^{T} d(z_t) d(z_t)' \right)^{-1}$$

where $z_t$ is the class to which $x_t$ belongs. This solution can be easily verified by noticing that the optimization problem corresponds to $n$ regression problems from $d(z)$ to each of the components of $x_t$. From $\hat{A}$, the diagonal terms of $\Sigma$ can then be estimated. We denote by $\hat{\Sigma}$ the empirical estimate of $\Sigma$. This generative model can then be used in a Bayes classifier. By setting the prior on classes to be constant and using Bayes rule, we obtain the following classification rule:

$$\arg\max_z g_{d(z)}(x) = \arg\max_z \; (d(z)'B)\, x - d(z)'BCB'd(z)$$

where $B = 2\hat{A}\hat{\Sigma}^{-1}$ and $C = \frac{1}{4}\hat{\Sigma}$.

From this generative model, we derived two other *discriminative* models directly defined using the same decision function $g_{d(z)}(x) = (d(z)'B)\, x - d(z)'BCB'd(z)$, with $C$ constrained to be diagonal. We considered the following discriminative cost functions:

1. Model **LinDisc-1-vs-all** uses a multi-class cost:

$$C(x_t, y_t^z) = -1_{\{y_t^z = 1\}} \log \left( \frac{e^{g_{d(z)}(x_t)}}{\sum_{z'} e^{g_{d(z')}(x_t)}} \right)$$

where the normalization is done over the classes instantiated in the training set only.

2. Model **LinDisc-0-1** uses a binary cost:

$$C(x_t, y_t^z) = -y_t^z \log(sigm(g_{d(z)}(x_t))) - (1 - y_t^z) \log(1 - sigm(g_{d(z)}(x_t))).$$

where $sigm(\cdot)$ is the sigmoid function.

Similarly to the input space view, to discriminate between two new classes $z_1^*$ and $z_2^*$, we simply compare the values for $g_{d(z_1^*)}(x^*)$ and $g_{d(z_2^*)}(x^*)$.

Finally, we derived a non-linear version of LinDisc-0-1, noted **NNet-0-1**, where the input $x_t$ is replaced by a learned representation $h(x_t) = tanh(Wx_t)$ corresponding to the hidden layer of a feed-forward neural network. We chose to derive such a non-linear version only for LinDisc-0-1 because it had the best performances among other linear model space view models. Also LinDisc-0-1 is more general, as it can also be used for multi-task learning when the $y_t^z \in [0, 1]$.

LinDisc-1-vs-all, LinDisc-0-1 and NNet-0-1 can be trained using a gradient descent optimization algorithm on the average training cost. Learning in NNet-0-1 implies learning a good value for the matrix $W$ and for the parameters $B$ and $C$ simultaneously.

## Description of the Datasets

In order to evaluate and empirically analyze the concept of zero-data learning, we conducted experiments in two different settings: multi-class classification and multi-task learning. The classification problems were taken in the context of character recognition, and the multi-task problem in the context of drug discovery.

### Classification Problems: Character Recognition

We considered two character recognition problems. The first one is to identify characters in license plates. A program was used to generate images of European license plates and simulate different effects such as shadows, mud, dirt and geometric distortions. The set of characters we considered include digits from 0 to 9, letters from A to Z and three other accentuated characters, for a total of 40 classes. Each input sample corresponds to a 30 by 15 pixel image containing one centered character, for a total of 54413 samples. Because of the large number of experiments performed, we used a limited number of samples per class (200).

The second dataset corresponds to handwritten alphanumeric characters[1], where the input corresponds to a 20 by 16 pixel image in binary format. There are 36 classes, corresponding to the letters from A to Z and digits from 0 to 9, and there are 39 examples per class.

We also constructed small 7 by 5 pixels "canonical" representations of all characters found in both datasets, which were used as task description binary vectors. It was inspired by *sys* font characters of size 8. Some of these task descriptions are displayed in Figure 3.

[1]Available in Matlab format at `http://www.cs.toronto.edu/~roweis/data/binaryalphadigs.mat`

### Multi-task Problem: Multi-target Virtual High-throughput Screening

In this domain, one is interested in building a decision-making algorithm that can identify reliably whether a molecular compound $x_t$ is active ($y_t^z = 1$) in the presence of a potentially new biological agent $z$ (meaning that it might "cure the disease" associated with that agent) or not ($y_t^z = 0$). Then, a ranking is produced by a model for a biological agent by sorting the molecular compounds in order of the model predictions for that agent.

All the molecular compounds used in this study were provided by an anonymous pharmaceutical company. We experimented on the same 7 agents for which Erhan *et al.* reported results in a classical multi-task setting, in which each of these agents is treated as a task. Note that they did not report results when there was no training data for the agent at test time, as we did here.

We report results for two numerical description or features of the molecular compounds. In the first approach, 469 real-valued features ranging from atom frequencies to topological indices to 3D surface area descriptors were computed in the Molecular Operating Environment (MOE, version 2004.03, a package for high throughput discovery). The numerical values were normalized to zero mean and unit variance. In the second approach, 3640 binary features were generated using the pharmacophore triplet module in the Tripos software Sybyl6.9. The respective prediction power of the two types of features in single tasks are similar, but can vary slightly from task to task.

The pharmaceutical company also provided 25-dimensional task description vectors specific to each agent. The 25 features are based on the properties of the so-called binding pockets of the biological agents and are described in Erhan *et al.*. It should be noticed that not only these descriptors do not come from the same space as the input features, but in fact, they do not even "describe" the same kinds of objects.

## Experimental setup

For the classification problems, we used the following experimental setup. We tested the ability of the models to perform zero-data generalization by testing the discrimination ability between *two character classes not found in the training set*. We measured performance by looking at the outputs for the test classes, selecting the class for which the output was the largest and comparing it with the labeled class. This procedure is illustrated in Figure 3.

The experimental procedure we followed was designed to measure the relationship between the number of classes in the training set and the zero-data classification performance at test time. For each fold of a multi-fold experiment, two classes were selected for the test set and a subset of the remaining classes for the training set. The discrimination performance among the two test classes was measured as more classes were added to the training set. For instance, starting from the fold example of Figure 3, we would train all models on the examples of classes '1', '2' and '3' and test on the examples of classes 'A' and 'B'. Then, we would insert in

the training set the examples from classes other than 'A' and 'B' and repeat the procedure. This was done for $\binom{6}{2} = 15$ folds, each constructed by picking 2 test classes among the same 6 randomly selected classes. The inserted classes from one training set size to the other are the same for all folds.

For the multi-task ranking problem, the experimental setup was simple: each of the 7 biological agents (tasks) was used as a test task when training from the 6 remaining agents. Given that the dataset is unbalanced, we use a ROC-like measure called Area under the LIFT curve or AUL (see Quionero Candela *et al.* for more details). A random ranker will obtain 1 under this measure on average and better ranking yields higher scores.

For all models and all experiments, model selection was done to select the hyper-parameters of the models such as the weight decay factors, the number of hidden units in NNet-0-1 or the SVM kernel parameters. Hyper-parameters were chosen based on their associated performance (classification error or AUL) on a validation set containing data from the same classes/tasks as those in the training set. The LinDisc-1-vs-all and LinDisc-0-1 models were trained using conjugate gradient descent for the classification problem. For the multi-task ranking problem, because of the large number of training examples, we used stochastic gradient descent with early stopping on the validation set AUL. NNet-0-1 was also trained using stochastic gradient descent. We only considered the second degree polynomial kernel in order to compare its performance with the linear model space view models, which have similar parameterizations.

## Results

Figure 4 shows for the two classification problems the relationship between zero-data error and the number of different training classes, i.e. the number of classes instantiated in the training set. We can see that classification error tends to improve as the number of training classes increases. The Gaussian kernel SVM gives the best performance on the license plate character problem, reaching almost perfect generalization error, and the performance of NNet-0-1 is slightly worse. On the binary alphanumeric dataset, all models behave similarly. The progression of the classification error for the LinGen model is particularly surprising. Indeed, the error curve has a "U" shape, meaning that after a certain number of training classes, the classification error worsens. We hypothesize that this is due to the effect of atypical classes and the small capacity of the LinGen model. These "outlier" classes would particularly hurt a misspecified generative model such as LinGen.

We also measured the classification error when *allowing the models to produce as answer one of the training classes*. For character recognition, this setup might be more appropriate in certain contexts where new symbols are introduced and must be considered by the system (e.g. the Euro sign in 1996). This experiment is also informative of situations where the total number of possible outputs is too large to be covered by the training data. We did this by comparing the models' predictions among the union of the test classes and training classes. The results are displayed in Figure 5. As can be seen, the SVM classifiers, especially SVM$_{rbf}$, almost ignore the test classes in their prediction. On the other hand LinDisc-0-1 frequently selects the correct label. One could argue that this is a consequence of doing inadequate model selection, on a validation set that contains the same classes as those in the training set. So, we ran the same experiments but with the validation and training sets containing data from different classes. The general picture of 5 did not change and the overfitting was only slightly reduced.

This points to an unusual type of overfitting phenomenon. Given an inappropriate choice of model, it seems that a learning algorithm can heavily *overfit on the training classes*, that is it can be relatively overconfident in its output for the training classes with respect to the test classes, regardless of the input. This is exemplified by the fact that similar model structures can produce important differences in zero-data generalization performance. The drastic change in relative performances of the polynomial SVM and LinDisc-0-1 model is especially interesting, as both models have bilinear terms and can be written as $f_z(x) = d(z)'Wx + d(z)'Vd(z) + x'Ux$. The LinDisc-0-1 model has more constraints, with $W = B$, $V = BCB'$ and $U = 0$.

For the multi-task ranking experiment, we only considered the input space view SVM models, the LinDisc-0-1 model and the NNet-0-1 model, the other models being inappropriate for this problem. We also allowed SVM$_{rbf}$ to have two distinct bandwidth hyper-parameters for the task description and the input, since such model selection was computationally feasible for this experiment. However, we obtained only a slight performance improvement. This model is more similar to what is proposed by Tsochantaridis *et al.*. The results are displayed in Table 6.

As can be seen, the model space view models generalize better to the new biological agent than the input space SVM models in the majority of cases. They are also more likely to generalize to the new biological agent, i.e. to do better than a random ranking. Though better results are usually obtained when training data is available for an agent (AUL of more than 2 are easy to reach), a zero-data ranker can be a nice initial solution to determine which drugs to study first. In other words, the zero-data ranker can be used to choose the first samples to label in an active learning setting.

## Conclusion and Discussion

We have introduced the problem of zero-data generalization and have shown that it can be addressed. We have formalized and empirically investigated it, proposed two approaches to tackle this problem and have analyzed some of their properties. Clearly, the performance one can obtain is highly dependent on the quality of the task representations and their relative consistency. How we can quantitatively evaluate these important properties of the descriptions is not obvious and is a question worth exploring. One of the interesting experimental results is the observation of an overfitting phenomenon at the level of classes.

We also believe that zero-data learning should be considered not necessarily as an answer to very specific application-related problems, but also as an avenue to extend the way machine learning algorithms are used. Generally

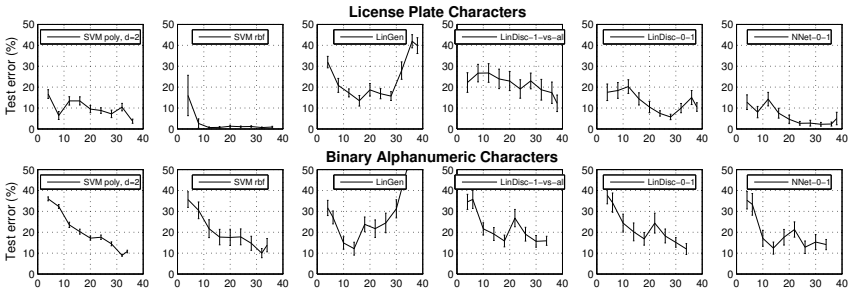


Figure 4: Zero-data task generalization error progression on classification problems as the number of training classes increases. The error bars correspond to the error standard deviation.

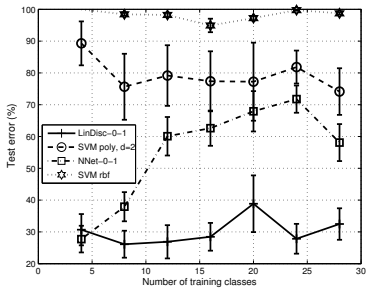


Figure 5: Zero-data task generalization error progression on license plate characters dataset when we allow to output a training class.

| Algorithm | MOE/FP input features results (AUL) | | | | | | |
|---|---|---|---|---|---|---|---|
| | G1A | G1D | G1F | G1H | G1I | G1S | G1U |
| $\mathbf{SVM}_{rbf}$ | 1.99/**3.02** | 0.94/0.88 | 1.07/0.83 | 0.99/1.03 | **1.50**/1.36 | 0.99/0.68 | 0.94/0.89 |
| $\mathbf{SVM}_{poly,d=2}$ | **2.42**/2.98 | 0.83/0.77 | 0.87/0.82 | 0.98/0.89 | 1.01/1.10 | 0.91/0.69 | 0.92/0.84 |
| **LinDisc-0-1** | 1.10/1.39 | 1.34/1.46 | **1.25**/0.85 | **1.36**/1.59 | 1.45/**1.45** | 0.96/**1.61** | 0.93/0.96 |
| **NNet-0-1** | 1.44/1.59 | **1.37/1.49** | 1.20/**1.14** | 1.20/**1.61** | 1.39/1.41 | 0.96/ 1.10 | 0.93/0.81 |

Figure 6: Zero-data task generalization performance for each biological agent, according to area under the LIFT curve (AUL) score, with best performance put in bold when bigger than 1 (random ranker baseline) for each type of input features.

speaking, zero-data learning gives the possibility to define some kind of user interface with the trained models via task descriptions. It can then be considered as falling between human engineered systems, which requires a full, manually-specified description of a new task's solution, and pure machine learning, which requires data for that task to come up with an appropriate model for the solution. Zero-data learning combines aspects of both approaches, taking advantage of human knowledge (task descriptions) and training data (from related tasks), in such a way that it needs less human labor and training data for the new task. Moreover, one would expect an artificial intelligence system to understand instructions specifying a task and perform the task without any additional supervised training, if it has already been trained on related tasks described in the same language.

## Acknowledgements

We thank Csaba Szepesvári for making the license plates dataset available to us.

## References

Bakker, B., and Heskes, T. 2003. Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research* 4:83–99.

Bierderman, I. 1987. Recognition-by-components: A theory of human image understanding. *Psychological Review* 94(2):115–147.

Bonilla, E. V.; Agakov, F. V.; and Williams, C. K. I. 2007. Kernel multi-task learning using task-specific features. In *Proceedings of AISTATS'2007*.

Caruana, R. 1997. Multitask learning. *Machine Learning* 28(1):41–75.

Chapelle, O.; Schölkopf, B.; and Zien, A. 2006. *Semi-Supervised Learning*. Cambridge, MA: MIT Press.

Erhan, D.; L'Heureux, P.-J.; Yue, S. Y.; and Bengio, Y. 2006. Collaborative filtering on a family of biological targets. *Journal of Chemical Information and Modeling* 46(2):626–635.

Ghosn, J., and Bengio, Y. 2003. Bias learning, knowledge sharing. *IEEE Transactions on Neural Networks* 14:748–765.

Miller, E. G. 2002. *Learning from one example in machine vision by sharing probability densities*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Moses, Y.; Ullman, S.; and Edelman, S. 1996. Generalization to novel images in upright and inverted faces. *Perception* 25(4):443–461.

Quionero Candela, J.; Rasmussen, C. E.; Sinz, F.; Bousquet, O.; and Schlkopf, B. 2006. *Evaluating Predictive Uncertainty Challenge*. Lecture Notes in Computer Science : 3944. Heidelberg, Germany: Springer. 1–27.

Raina, R.; Battle, A.; Lee, H.; Packer, B.; and Ng, A. Y. 2007. Self-taught learning: transfer learning from unlabeled data. In *ICML*, 759–766.

Schein, A. I.; Popescul, A.; Ungar, L. H.; and Pennock, D. M. 2002. Methods and metrics for cold-start recommendations. In *SIGIR '02*, 253–260. New York, NY, USA: ACM Press.

Thrun, S. 1996. Is learning the $n$-th thing any easier than learning the first? In Touretzky, D., and Mozer, M., eds., *Advances in Neural Information Processing Systems 8*, 640–646. Cambridge, MA: MIT Press.

Tsochantaridis, I.; Joachims, T.; Hofmann, T.; and Altun, Y. 2005. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.* 6:1453–1484.

Zhu, X.; Rogers, T. J.; Qian, R.; and Kalish, C. 2007. Humans perform semi-supervised classification too. In *AAAI*, 864–. AAAI Press.