Generalization to a zero-data task: an empirical study

Hugo Larochelle, Dumitru Erhan and Yoshua Bengio
Dept. IRO, Université de Montréal, C.P. 6128, Montreal, Qc, H3C 3J7, Canada
{*larocheh,erhandum,bengioy*}*@iro.umontreal.ca*
http://www.iro.umontreal.ca/∼lisa

We study the problem of generalization to a *zero-data task*. A zero-data task corresponds to a learning problem for which no training data is available and only a description $d(y)$ of the task y is given. Though learning in such a scenario might seem impossible, such tasks can actually be found in the context of a number of fields: for instance, in drug discovery [1], where one needs to decide whether a candidate molecule x is a possible cure for a **new disease** y, or in the so-called "cold start" problem of recommender systems, where one needs to determine whether a **new item** y (movie, album, book, etc.) should be recommended to a user x or not. It is also related to the *one sample learning* problem [2], where only one training example per class is provided in a classification setting.

There are at least two approaches to learning zero-data tasks, which correspond to two different views of the problem. We refer to these approaches as the **input space view** and the **model indexation view**. The input space view considers the *input* of the learning problem to be the concatenation of a sample with the corresponding task description $[x, d(y)]$[1]. The desired *output* for the respective task y, noted $z(y)$, can be 1 if sample x is from class y and 0 otherwise for a classification setting, or a value of the dependent variable associated with x in task y, in a regression setting. Then, to obtain an answer for a new task y^* from a sample x^*, the input $[x^*, d(y^*)]$ is provided to the trained model.

The model indexation view considers the model $f_y(x)$ for a task y to be a **function** of its description $d(y)$, i.e. $f_y(x) = g_{d(y)}(x)$. For example, $g_{d(y)}(x)$ could be a Gaussian density where the mean is a function of $d(y)$, or it could be a linear classifier (perceptron) for which the weights are a function of $g_{d(y)}$. To train the "model of models" $g_{d(y)}(x)$, one can simply optimize its parameters to minimize the average cost for each pair of samples and tasks from the training set $\frac{1}{T}\sum_{x_t,y_t,z_t} C(z_t, g_{d(y_t)}(x_t))$. Though not all algorithms can be cast in this framework, it 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 $g_{d(y)}(x) = g\left([x, d\left(y\right)]\right)$.

We conducted experiments in a classification and a ranking setting, in order to evaluate these two approaches. We tested the four following models on these problems. We evaluated the input space view principle using an SVM model with a Gaussian and a polynomial kernel and we tested the model indexation view principle using the following models:

1. a Gaussian model for which the mean of the gaussian is a linear function of description, i.e. $X|Y = y \sim \mathcal{N}(A \cdot d(y), \Sigma)$ where Σ is a diagonal matrix with individual variances for each components: different values of y imply different gaussian centers $A \cdot d(y)$. This model can be trained to maximize the likelihood of the samples x_t given their corresponding classes and this problem can be solved analytically; using flat priors over the classes and Bayes' rule we can recover the posteriors. We will refer to this linear generative model as **LinGen**.

   It can be shown that the decision rule for this model is as follows:
   $$\mathrm{argmax_y g_{d(y)}(x)} \quad = \quad \mathrm{argmax_y}\ \mathrm{(d(y)'B)\,x - d(y)'BUB'd(y)} \tag{1}$$
   where $B = 2A\Sigma^{-1}$ and $U = \frac{1}{4}\Sigma$. This decision rule is the same as the one of a perceptron with weights $(d(y)'B)$ and bias $-d(y)'BUB'd(y)$ for class y.

2. a linear model that uses the decision rule of equation 1 but that is trained on the following discriminative criteria:
   $$-\sum_{x_t,y_t} \log(softmax(d(y_t)'Bx_t - d(y_t)'BUB'd(y_t)))$$
   where the softmax normalization constant is computed over all the **training classes only, not the test classes**. We will refer to this linear discriminative model as **LinDisc_1-vs-all**.

3. a linear model that also uses the decision rule of equation 1 but that is trained on the following discriminative criteria:
   $$-\sum_{x_t,y_t}\sum_{y^*} \left(z_t(y^*)\log(sigmoid(g_{d(y^*)}(x))) + (1 - z_t(y^*))\log(1 - sigmoid(g_{d(y^*)}(x)))\right)$$
   where again $g_{d(y^*)}(x) = d(y^*)'Bx_t - d(y^*)'BUB'd(y^*)$. The sum over y^* is also done only over the training tasks (classes). We will refer to this other linear discriminative model as **LinDisc_0-1**.

The first experiment comes from a license plate character recognition problem. The training data corresponds to samples from different types of characters and a 7×5 pixels description of these characters. Testing corresponds to classifying samples from **two types of characters not found in the training set** (see figure 1 for an illustration of the problem). We selected randomly 6 of the 40 possible characters (0 through 9, A through Z and 4 special characters) found in the dataset, from which we constructed all possible binary classification problems (a total of 15) on which the different models were evaluated. We varied the size of the training set by varying the number of types of characters that it contained.

[1] The square brackets correspond to concatenation

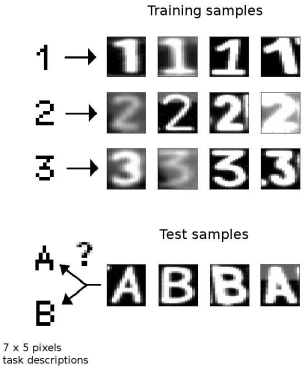

Figure 1: Illustration of classification experiment

Agent	LIFT LinDisc_0-1	LIFT SVM
A	168	105
D	153	108
F	97	150
H	171	161
I	163	130
S	173	106
U	95	105

Figure 2: Results on the drug discovery experiment

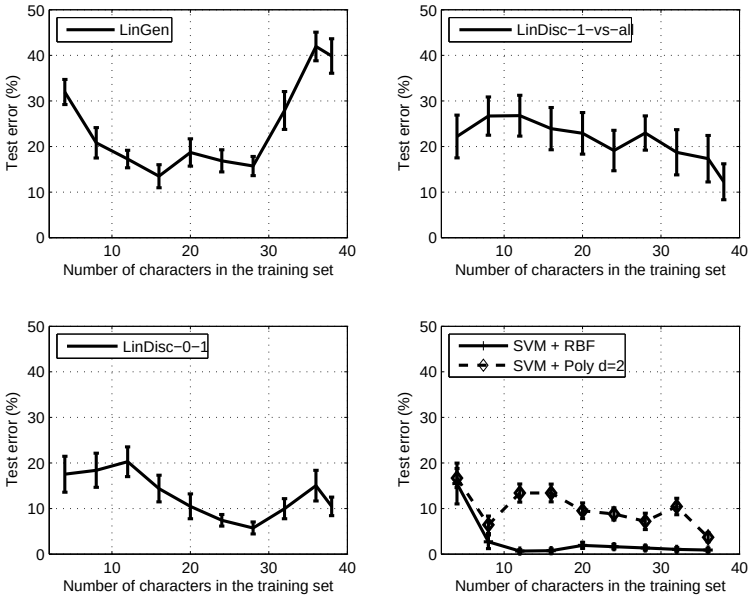

Figure 3: Estimation of the generalization performance on a zero-data task for LinGen, LinDisc_1-vs-all, LinDisc_0-1 and SVM

We also report a set of experiments performed in the context of virtual screening for drug discovery, which is a ranking problem. In this domain, one is interested in building a decision-making algorithm that would identify reliably molecular compounds x that are active in the presence of a biological agent y (i.e., they "could cure the disease"). The dataset that we use consists of 7 biological agents, along with features for each of them. The experimental setup was simple: the algorithm is presented with the training data from 6 agents and is tested on the data from the remaining agent. Given that the dataset is unbalanced, we use a ROC-like measure called LIFT[3]. It corresponds roughly to a point on the ROC curve and the higher the value, the better, with 100 corresponding to a random binary decision.

The hyper-parameters of all the models were tuned on a validation set which contained samples from the same classes as those of the training set. Figure 3 displays the results for the character classification problem. The classification error on a zero-data task is estimated by computing the average of the classification errors on the 15 binary problems. The best model in term of estimated generalization error is the SVM model with a Gaussian kernel, which can achieve an error of 2%. The best model indexation view model is LinDisc_0-1. We can also observe that the progression of the error with respect to the number of training classes has a U shape for models LinGen and LinDisc_0-1. This is rather counter-intuitive, as we might expect that more training classes will result in better generalization to new classes. This phenomenon might be explained by the fact that the models have a hard time fitting the training data as there are more training classes, given that these models have a fixed capacity.

In the case of the drug discovery experiment, only the SVM model and LinDisc_0-1 could be used, the other models being only adequate for classification problems. The results are reported in table 2. Both the SVM and the LinDisc_0-1 models manage to generalize successfully to many of the new tasks. LinDisc_0-1 is clearly better and taken together the algorithms manage to generalize relatively well to 6 out of 7 tasks.

The performance of the SVM on the character problem is impressive, but it is most likely due to the greater capacity of the Gaussian kernel; the other models are bilinear. Therefore, the fairer comparison is between these models and an SVM with a polynomial kernel of degree 2, whose performance we also plot. Both LinDisc_0-1 and the polynomial SVM reach about 5% error. In the future, we plan to introduce non-linearity in the models that we presented above so that we could compare with the RBF kernel. We have investigated many other options for the drug discovery problem, including neural network architectures and a kernel algorithm—the results presented in table 2 are quite encouraging as the problem is quite challenging (the other learners did not perform nearly as well).

References

[1] Dumitru Erhan, Pierre-Jean L'Heureux, Shi Yi Yue, and Yoshua Bengio. Collaborative fi ltering on a family of biological targets. *Journal of Chemical Information and Modeling*, 46(2):626–635, 2006.

[2] Erik Miller. *Learning from one example in machine vision by sharing probability densities*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2002.

[3] Gregory Piatetsky-Shapiro and Sam Steingold. Measuring lift quality in database marketing. *SIGKDD Explor. Newsl.*, 2(2):76–80, 2000.