Generalization to a zero-data task: an empirical study

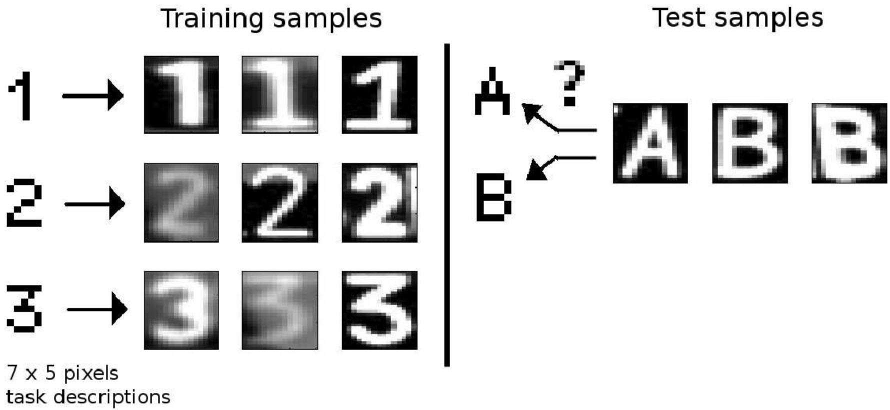
Hugo Larochelle, Dumitru Erhan and Yoshua Bengio
Université de Montréal

20/03/2007

Introduction

- **What is a zero-data task ?** task for which no training data are available
- **How can we generalize in such a case ?**
  - using a set of task representations or vectors $d(z)$ (given a priori)
  - using data from other related tasks
- **Why should we care ?**
  - occurs in certain applications : drug discovery, song/movie recommendation
  - gives worst case scenario for inductive transfer
  - corresponds to a more natural learning scenario for a human

Illustration of the problem in classification



Related Work

- Related to *life-long learning* (Thrun, 1996) and inductive transfer, where a learning agent has to sequentially learn some tasks f_i

  $$f_1 \longrightarrow f_2 \longrightarrow \cdots \longrightarrow f_T$$

  Task f_i is learned from training set $\mathcal{D}_i$ and previous training sets
- Here, we provide the possibility for tasks to be **learned from a description**
- Related to **one-shot** or **one-sample** learning (Miller, 2002), where a classification problem is learned
  - using only one training sample per class
  - using data from other related tasks
- If we let $d(z)$ be a sample from class z, zero-data classification learning is equivalent to one-shot learning

Two views of zero-data task learning

Here, we differentiate two views of the zero-data task learning problem :

- the **input space view**
- the **model indexation view**

We note :

- x_t as the input
- z as a task ID and $d(z)$ as its description vector
- y_t^z as the desired model output for input x_t and task z
- $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ as the training and test sets, with *data from different tasks*

Data notation : classification example

$z \rightarrow$ / $d(z) \rightarrow$	1 1	2 2	3 3	4 A	5 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 (columns 1–3)   test data (columns 4–5)

Data notation in general

$z \longrightarrow$	1	2	3	4	5
$d(z) \longrightarrow$					
x_t	y_t^1	y_t^2	y_t^3	y_t^4	y_t^5
	3	-	-1	0.5	-
	2.5	1	-	-	-
	-2	3	-	0.25	2
	-1	1	-	-2	-3
	1	-	1.5	3.5	4

training data

test data

Input space view

- The input space view is a reformulation of the problem into a standard learning problem.
- The corresponding training procedure is :
  - for each $(x_t, y_t^z) \in \mathcal{D}_{train}$, generate new sample
    $$\left([x_t, d(z)], y_t^z\right)$$
  - use the generated samples as the training set for any learning algorithm to obtain a model $f^*(\cdot)$
- The prediction for a new task z^{new} and an input x is simply

$$f^* \left([x, d(z^{new})]\right)$$

provided that $d(z^{new})$ is given

Model indexation view

- The model indexation view considers the model $f_z(x)$ for task z to be a **function** of its description $d(z)$:

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

- input space view is a special case of the model indexation view, which does not differentiate x from $d(z)$:

$$g_{d(z)}(x) = g([x, d(z)])$$

- Training corresponds to optimizing :

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

- The prediction for a new task z^{new} and an input x is simply

$$g_{d(z^{new})}(x)$$

provided that $d(z^{new})$ is given

Model indexation view models

- The first model we tried (**LinGen**) is a Gaussian model :

$$X|Y^z = 1 \sim \mathcal{N}(Ad(z), \Sigma)$$

  where Σ is diagonal
- Setting the prior on the classes to be constant, it can be shown that Bayes rule gives the following decision rule :

$$\operatorname*{argmax}_z g_{d(z)}(x) = \operatorname*{argmax}_z \left(d(z)'B\right) x - d(z)'BUB'd(z)$$

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

Model indexation view models

- The two other models used the same decision rule directly :

$$g_{d(z)}(x) = \left(d(z)'B\right) x - d(z)'BUB'd(z)$$

but with two different *discriminative* costs

- Model **LinDisc_1-vs-all** uses a multi-class cost :

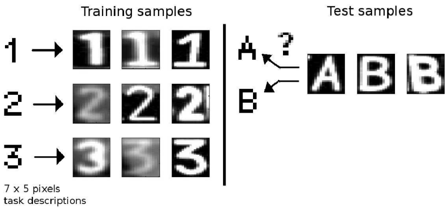
$$C(x_t, y_t^z) = -1_{\{y_t^z=1\}} \log(softmax(g_{d(z)}(x_t)))$$

where the softmax normalization is done over the **training classes only, not the test classes**

- Model **LinDisc_0-1** uses a binary cost :

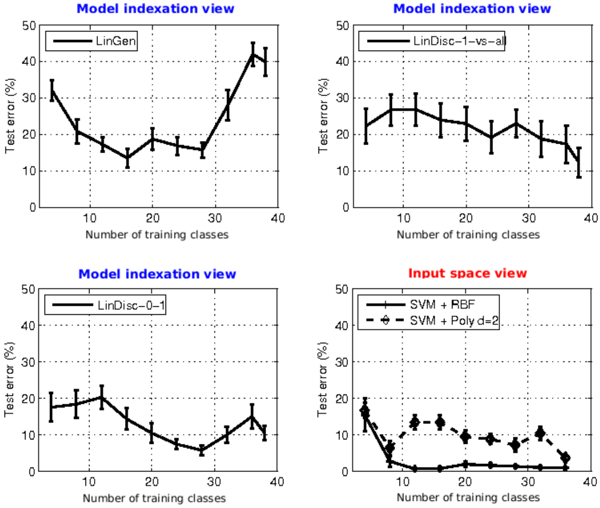
$$\begin{aligned} C(x_t, y_t^z) &= -y_t^z \log(sigm(g_{d(z)}(x_t))) \\ &\quad -(1 - y_t^z) \log(1 - sigm(g_{d(z)}(x_t))) \end{aligned}$$

First problem : classification



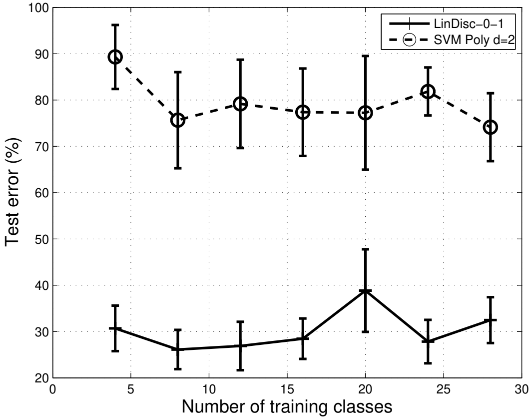
- Selected 6 classes randomly $\Rightarrow$ 15 pairs of *test classes* (character types)
- 200 samples per class
- Trained on samples from 4, 8, 12, ..., 36 and 38 *training classes* (mutually exclusives from the test classes)
- Measured generalization using classification error rate in the *test classes*

Results

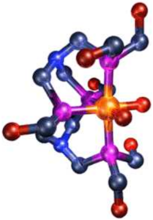

Model indexation view
LinGen
Test error (%)
Number of training classes
Model indexation view
LinDisc−1−vs−all
Test error (%)
Number of training classes
Model indexation view
LinDisc−0−1
Test error (%)
Number of training classes
Input space view
SVM + RBF
SVM + Poly d=2
Test error (%)
Number of training classes

Results

Classification error **when allowing to output a training class** at test time



Second problem : ranking



- Problem setting
  - x_t is a molecule (drug candidate)
  - z is a protein (disease) and $d(z)$ is its description
  - y_t^z is 1 if the molecule and the protein reach a certain level of activity, 0 otherwise
- We perform 7-fold cross-validation, by training on 6 proteins and testing on the held out protein
- The evaluation metric is the *area under the lift curve*
- A random ranking would obtain 1 under this measure, on average

Results

Area under the lift curve (AULC) results for the different proteins.

Protein	AULC (MOE)			AULC (FP)		
	LinDisc_0-1	SVM Gauss	Poly d=2	LinDisc_0-1	SVM Gauss	SVM Poly d=2
A	1.10	1.10	**2.02**	1.39	**2.79**	2.76
D	**1.34**	0.94	0.88	**1.46**	0.83	0.87
F	**1.25**	0.89	0.77	**0.85**	0.83	0.81
H	**1.36**	0.96	0.98	**1.59**	0.79	0.79
I	**1.45**	1.00	1.02	**1.45**	1.24	1.23
S	**0.96**	0.94	0.89	**1.61**	0.73	0.76
U	**0.93**	0.92	0.83	**0.96**	0.78	0.80

As a comparison, on the training tasks, the AULC is usually above 2.1.

Conclusion and future work

Conclusion

- We presented zero-data task learning and described different ways of approaching this problem
- We showed empirically that generalization is possible in this framework

Future work

- Do more experiments in other domains
  - recognition of more symbols
  - movie recommendation
- Develop a non-linear version of **LindDisc_0-1**

Conlusion and future work

THANK YOU !

Miller, E. (2002).
Learning from one example in machine vision by sharing probability densities.
PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science.

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