Generalizing to a zero-data task: a linear model case study

Hugo Larochelle
with Yoshua Bengio and Aaron Courville

17/08/06

Plan

Introduction

- **What is a zero-data task ?** task for which no training data are available

Introduction

- **What is a zero-data task ?** task for which no training data are available
- **How can we generalize in such a case ?**

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 (given a priori)

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 (given a priori)
  - using data from other related tasks

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 (given a priori)
  - using data from other related tasks
- **Why do this ?**

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 (given a priori)
  - using data from other related tasks
- **Why do this ?**
  - this situation occurs in certain applications : drug discovery, song/movie recommandation, NLP tasks

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 (given a priori)
  - using data from other related tasks
- **Why do this ?**
  - this situation occurs in certain applications : drug discovery, song/movie recommandation, NLP tasks
  - gives us a worst case scenario in an inductive transfer model

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 (given a priori)
  - using data from other related tasks
- **Why do this ?**
  - this situation occurs in certain applications : drug discovery, song/movie recommandation, NLP tasks
  - gives us a worst case scenario in an inductive transfer model
  - hasn't been studied so much

In this talk

- I will present some experiments in the case of a linear model

In this talk

- I will present some experiments in the case of a linear model
- I will compare different training settings for the linear model in a classification task :
  - discriminative (with 2 different cost functions)
  - generative

In this talk

- I will present some experiments in the case of a linear model
- I will compare different training settings for the linear model in a classification task :
  - discriminative (with 2 different cost functions)
  - generative
- I will present results for two problems :
  - classifi cation problem
  - ranking problem

First problem : classifi cation

- **Tasks’ nature :** character recognition in the context of license plates

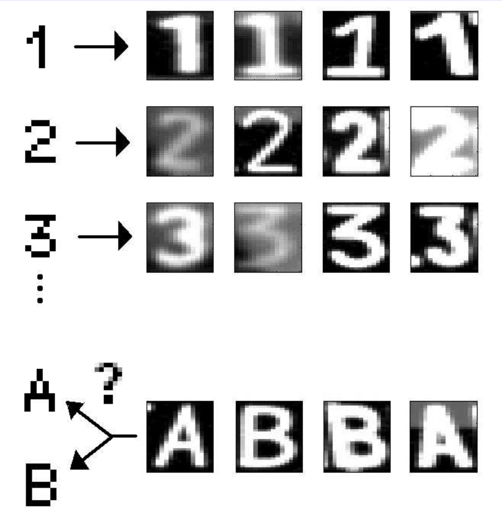
First problem : classifi cation

- **Tasks' nature :** character recognition in the context of license plates
- **Training tasks :** we are given a set of class descriptors (vectors) with corresponding instances from these classes

First problem : classifi cation

- **Tasks' nature :** character recognition in the context of license plates
- **Training tasks :** we are given a set of class descriptors (vectors) with corresponding instances from these classes
- **Test tasks :** we are given two class descriptors and we need to discriminate between these two classes for a given set of untagged instances

First problem : classifi cation



Model variations

- **Question :** how do we get a model (classifier) for a new task ?

Model variations

- **Question :** how do we get a model (classifier) for a new task ?
- **Answer :** by predicting it.

Model variations

- **Question :** how do we get a model (classifier) for a new task ?
- **Answer :** by predicting it.

- In general, we could model “models” like this :

$$f_y(x) = \left[g(d_y)\right](x)$$

where x is the input, y is a task (class), d_y is it’s descriptor and $g(d_y)$ predicts f_y

Model variations

- **Question :** how do we get a model (classifier) for a new task ?
- **Answer :** by predicting it.
- In general, we could model "models" like this :

$$f_y(x) = \left[g(d_y)\right](x)$$

where x is the input, y is a task (class), d_y is it's descriptor and $g(d_y)$ predicts f_y
- Here, we will have $g(d_y)$ be a linear transformation $A'd_y$, hence

$$f_y(x) = d_y A x$$

or, in the energy-based framework

$$E(x, y) = -d_y A x$$

Model variations

We will compare 2 training criteria

- discriminative (1 vs all)

$$\mathcal{L}(x, y) = -log(softmax(-E(x, y)))$$

with normalisation over the training classes.

Model variations

We will compare 2 training criteria

- discriminative (1 vs all)

$$\mathcal{L}(x, y) = -log(softmax(-E(x, y)))$$

  with normalisation over the training classes.
- generative (Gaussian model, maximum likelihood)

$$X|Y = y \sim \mathcal{N}(A'd_y, \Sigma)$$

$$D_y \sim \mathcal{U}(0, 1)^m$$

  where Σ is a diagonal matrix

Model variations

We will compare 2 training criteria

- discriminative (1 vs all)

$$\mathcal{L}(x, y) = -log(softmax(-E(x, y)))$$

  with normalisation over the training classes.

- generative (Gaussian model, maximum likelihood)

$$X|Y = y \sim \mathcal{N}(A'd_y, \Sigma)$$
$$D_y \sim \mathcal{U}(0, 1)^m$$

  where Σ is a diagonal matrix

- We use standard regularisation techniques, such as early stopping, weight decay, etc.

Experiments

- For each of the 4 folds, we select two classes to isolate from others

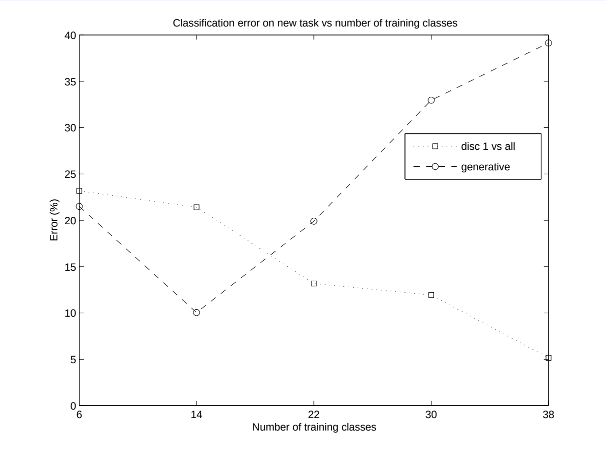
Experiments

- For each of the 4 folds, we select two classes to isolate from others
- Validation is performed on a set of instances from the training classes

Experiments

- For each of the 4 folds, we select two classes to isolate from others
- Validation is performed on a set of instances from the training classes
- We vary the number of classes present in the training set, but we keep the same test classes of the different folds

Results


Classification error on new task vs number of training classes
Error (%)
40
35
30
25
20
15
10
5
0
6
14
22
30
38
Number of training classes
disc 1 vs all
generative

Observations

- Generative model does better when few task examples are available

Observations

- Generative model does better when few task examples are available
- Discriminative model does better when many task examples are available

Observations

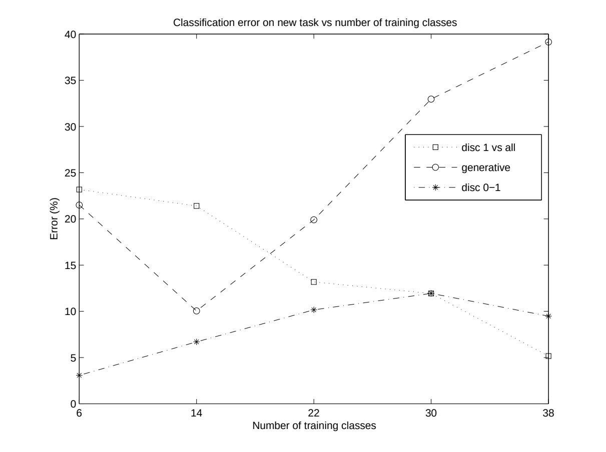
- Generative model does better when few task examples are available
- Discriminative model does better when many task examples are available
- More “examples” of tasks doesn’t imply better generalisation

Other model variation

- Discriminative 0-1 multi-task classification

$$\begin{aligned}\mathcal{L}(x,y) &= -log(sigmoid(-E(x,y))) \\ &\quad - \sum_{c \in \mathcal{Y}, c \neq y} log(1 - sigmoid(-E(x,c)))\end{aligned}$$

Results


Classification error on new task vs number of training classes
Error (%)
Number of training classes
0
5
10
15
20
25
30
35
40
6
14
22
30
38
disc 1 vs all
generative
disc 0−1

Second problem : ranking

- **Tasks' nature :** virtual screening for drug discovery

Second problem : ranking

- **Tasks' nature :** virtual screening for drug discovery
- **Training tasks :** a set of "descriptors" for different proteins (disease), for which we want to estimate the probability of activity in the presence of different compounds (drug candidates). Each compound has a vectorial representation, based on features of its molecular structure.

Second problem : ranking

- **Tasks' nature :** virtual screening for drug discovery
- **Training tasks :** a set of "descriptors" for different proteins (disease), for which we want to estimate the probability of activity in the presence of different compounds (drug candidates). Each compound has a vectorial representation, based on features of its molecular structure.
- **Test tasks :** a new protein descriptor, with which we can get an estimator of the conditional probability of activity for each compound for that protein.
  We can then rank compounds in decreasing probability of activity, and makes suggestions of compounds to test.

Second problem : ranking

- **Tasks' nature :** virtual screening for drug discovery
- **Training tasks :** a set of "descriptors" for different proteins (disease), for which we want to estimate the probability of activity in the presence of different compounds (drug candidates). Each compound has a vectorial representation, based on features of its molecular structure.
- **Test tasks :** a new protein descriptor, with which we can get an estimator of the conditional probability of activity for each compound for that protein.
  We can then rank compounds in decreasing probability of activity, and makes suggestions of compounds to test.
- **Nature of descriptors d_y :** TOP SECRET !

Experimental setup

- We have 7 proteins and a bank of compounds. All compounds has been previously tested for certain proteins. The target can be “1” for an “active” compound and “0” for an “inactive” compound.

Experimental setup

- We have 7 proteins and a bank of compounds. All compounds has been previously tested for certain proteins. The target can be “1” for an “active” compound and “0” for an “inactive” compound.
- We perform a 7-fold cross-validation, by training on 6 proteins and testing on the held out protein.

Experimental setup

- We have 7 proteins and a bank of compounds. All compounds has been previously tested for certain proteins. The target can be “1” for an “active” compound and “0” for an “inactive” compound.
- We perform a 7-fold cross-validation, by training on 6 proteins and testing on the held out protein.
- The same model is used as in the first experiment. We use the cost of the sum of the cross-entropy, for the available activity target.

Experimental setup

- We have 7 proteins and a bank of compounds. All compounds has been previously tested for certain proteins. The target can be “1” for an “active” compound and “0” for an “inactive” compound.
- We perform a 7-fold cross-validation, by training on 6 proteins and testing on the held out protein.
- The same model is used as in the first experiment. We use the cost of the sum of the cross-entropy, for the available activity target.
- The evaluation metric is the “lift” :

$$LIFT = 100 \cdot \frac{a_s}{a_{100}}$$

where a_s is the fraction of active compounds in the s first compounds, as ranked by the model. Here, we use $s = 30$, which is a standard size in computational chemistry.

Results

LIFT results for the different proteins.

Protein	LIFT
A	167.90
D	152.89
F	97.07
H	170.57
I	163.07
S	172.91
U	95.27

For the training tasks, LIFT is around 220.

Results

LIFT results for the different proteins.

Protein	LIFT
A	167.90 $$$
D	152.89 $$$
F	97.07
H	170.57 $$$
I	163.07 $$$
S	172.91 $$$
U	95.27

For the training tasks, LIFT is around 220.

Future work

- Extention to possibly get rid of task-overfitting, by having

$$E(x,y) = -d_y A x - w_y x$$

for training classes, but

$$E(x,y) = -d_y A x$$

for test classes.

Future work

- Extention to possibly get rid of task-overfitting, by having

$$E(x,y) = -d_y A x - w_y x$$

  for training classes, but

$$E(x,y) = -d_y A x$$

  for test classes.
- Do more experiments, on other datasets (any suggestion ?)

Future work

- Extention to possibly get rid of task-overfitting, by having
$$E(x, y) = -d_y A x - w_y x$$
for training classes, but
$$E(x, y) = -d_y A x$$
for test classes.
- Do more experiments, on other datasets (any suggestion ?)
- Use a more complex model (Deep Belief Network ?)