Learning Structured Output Dependencies Using Deep Neural Networks

Soufiane Belharbi[1] SURNAME.LASTNAME@LITISLAB.EU
Clément Chatelain[1]
Romain Hérault[1]
Sébastien Adam[2]
[1]LITIS EA 4108, INSA de Rouen, Saint Étienne du Rouvray 76800, France.
[2]LITIS EA 4108, Université de Rouen, Saint Étienne du Rouvray 76800, France

Abstract

1. Problem Statement

Many recent machine learning applications address challenging problems where the data outputs are in *high dimension* and *structural dependencies* lie between these outputs. These inter-dependencies constitute a structure such as sequences, strings, trees, graphs which should be either discovered if unknown, or integrated in the learning process. Applications can be found in statistical natural language processing, bio-informatics, speech processing, etc. Many approaches have been proposed to deal with this category of problems such as Kernel based methods (kernel dependency estimation (Weston et al., 2002) and Discriminative methods (structured support vector machine (Blaschko & Lampert, 2008))), but they suffer from the pre-image problem. Graphical models (Hidden Markov Models (HMM), Conditional Random Fields (CRF)) are also made for modeling structured data, but they only provide a single transformation layer between the input and the output variables.

Our proposed method is a generic formulation for regression/classification with structured output data based on Deep Neural Networks (DNN) where we incorporate the inputs dependencies learning, the outputs dependencies learning and the supervised task in the same framework.

2. Proposed Objective

We extend the objective function of (Weston et al., 2008) for embedding input dependencies to output dependencies. Following this framework, the total cost $\mathfrak{L}$ of learning the parameters $\boldsymbol{\theta} = \{\theta, \theta_{in}, \theta_{out}\}$ of a model over a labeled dataset $\boldsymbol{\mathcal{D}} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ can be written:

$$\begin{aligned}\mathfrak{L}(\boldsymbol{\theta}, \boldsymbol{\mathcal{D}}(\mathbf{x}, \mathbf{y})) = \frac{1}{n}\sum_{i=1}^{n} \quad & \Big[\boldsymbol{\mathcal{C}}(\mathcal{M}(x_i;\theta,\theta_{in},\theta_{out}), y_i) \\ + \quad & \boldsymbol{\ell}_{\mathbf{in}}(\boldsymbol{\mathcal{R}}_{\boldsymbol{in}}(x_i;\theta_{in}), x_i) \\ + \quad & \boldsymbol{\ell}_{\mathbf{out}}(\boldsymbol{\mathcal{R}}_{\boldsymbol{out}}(y_i;\theta_{out}), y_i)\Big] \quad (1)\end{aligned}$$

The function $\mathcal{M}(.;\theta,\theta_{in},\theta_{out})$ is the mapping from the input to the output space. Only this part will be used at decision time. $\boldsymbol{\mathcal{R}}_{\boldsymbol{in}}(.;\theta_{in})$, $\boldsymbol{\mathcal{R}}_{\boldsymbol{out}}(.;\theta_{out})$ are reconstruction functions of the input and the output, such as auto-encoders. $\boldsymbol{\mathcal{C}}(.), \boldsymbol{\ell}_{in}(.), \boldsymbol{\ell}_{out}(.)$ are defined costs.

Through the reconstruction tasks, parameters θ_{in} and θ_{out} embed input and output dependencies respectively. We assume that the regression task can benefit from this dependencies, by transfer learning, as these parameters are shared among reconstruction and regression.

3. IODA Framework

With Input Output Deep Architecture (IODA) (Labbe et al., 2009; Lerouge et al., 2015), we have proposed an extension of pre-training strategy of Deep Neural Network to the output layers.

The objective function (Eq. 1) is relaxed by splitting it into three sub-costs, each one is associated to one of the following tasks:

- $\boldsymbol{\mathcal{R}}_{\boldsymbol{in}}(.;\theta_{in})$, which corresponds to pre-trained input layers,
- $\boldsymbol{\mathcal{R}}_{\boldsymbol{out}}(.;\theta_{out})$, which corresponds to pre-trained output layers,
- $\mathcal{M}(.;\theta,\theta_{in},\theta_{out})$, a supervised regression.

Input-layer pre-training is addressed as in (Hinton et al., 2006; Bengio et al., 2007) by stacking encoding parts of auto-encoders. For output-layer pre-training, layers are pre-trained backward, and it is the decoding parts of auto-encoders that are kept to initialize the weights of the layers.

The full procedure is depicted in Alg. 1. We assume the existence of these two functions:

- $X' \leftarrow$MLPFORWARD($[\mathbf{W}_1, .., \mathbf{W}_K]$,$X$) that propagates X through layers $[\mathbf{W}_1, .., \mathbf{W}_K]$,
- $[\mathbf{W}'_1, .., \mathbf{W}'_K] \leftarrow$MLPTRAIN($[\mathbf{W}_1, .., \mathbf{W}_K]$,$X$,$Y$) that trains layers $[\mathbf{W}_1, .., \mathbf{W}_K]$ using back-propagation algorithm according to a labeled dataset (X,Y).

Algorithm 1 Simplified IODA training algorithm

Require: X, a training feature set of size $Nb_{\text{examples}} \times Nb_{\text{features}}$
Require: Y, a corresponding training label set of size $Nb_{\text{examples}} \times Nb_{\text{labels}}$
Require: N_{input}, the number of input layers to be pre-trained
Require: N_{output}, the number of output layers to be pre-trained
Require: N, the number of layers in the IODA, $N_{\text{input}} + N_{\text{output}} < N$
Ensure: $[\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_N]$, the parameters for all the layers

Randomly initialize $[\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_N]$

Input pre-training

$R \leftarrow X$
for $i \leftarrow 1..N_{\text{input}}$ **do**
  {*Training an AE on R and keeps its encoding parameters*}
  $[\mathbf{W}_i, \mathbf{W}_{\text{dummy}}] \leftarrow$ MLPTrain($[\mathbf{W}_i, \mathbf{W}_i^\intercal]$, R, R)
  Drop $\mathbf{W}_{\text{dummy}}$
  $R \leftarrow$ MLPForward$[\mathbf{W}_i]$, R
end for

Output pre-training

$R \leftarrow Y$
for $i \leftarrow N..N - N_{\text{output}} + 1$ **step** -1 **do**
  {*Training an AE on R and keeps its decoding parameters*}
  $[\mathbf{U}, \mathbf{W}_i] \leftarrow$ MLPTrain($[\mathbf{W}_i^\intercal, \mathbf{W}_i]$, R, R)
  $R \leftarrow$ MLPForward($[\mathbf{U}]$, R)
  Drop $\mathbf{U}$
end for

Final supervised learning

$[\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_N]$
  $\leftarrow$ MLPTrain($[\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_N]$, X, Y)

4. Application to Facial Landmark Detection

Facial landmark detection is an example of structured output problem which aims at predicting a geometric shape induced by an input face image. It plays an important role in face recognition and analysis. Therefore, it has been studied extensively in the recent years. However, this task remains a challenging problem due to the complex variations in the face appearance caused by the high variation in the poses, expressions, illuminations and by partial occlusions.

Facial landmarks are a set of key points on human face images. These points are defined by their real coordinates (x,y) on the image as shown in Fig. 1. The number of landmarks is dataset or application dependent. As the positions of the points in the face shape are dependent (spatial dependencies), facial landmark detection task naturally falls into the structured output regression problem.

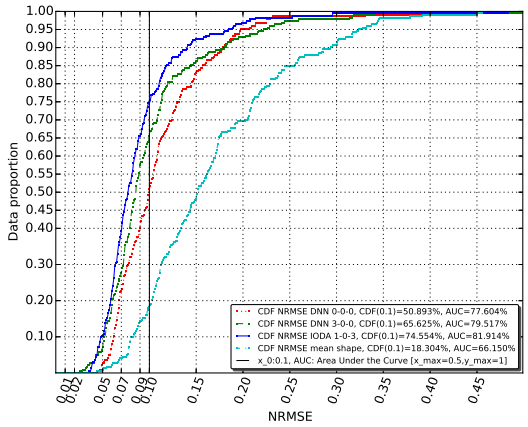
While most approaches consist in defining the face shape constraints explicitly (Cootes et al., 1995; Saragih et al., 2011; Zhu & Ramanan, 2012; Yu et al., 2013; Zhou et al., 2013), we propose to *learn* the structure of the face shape by discovering the hidden dependencies between the landmarks. For that purpose, we apply our IODA approach on this task (Belharbi et al., 2015) where we compare a non pre-trained, input pre-trained and input/output pre-trained (IODA) deep neural network on two challenging datasets: LFPW and HELEN.

Figure 1. Definition of 68 facial landmarks from LFPW (Belhumeur et al., 2011) training set.

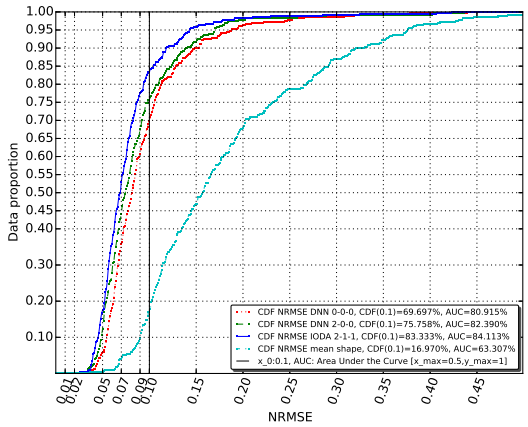
Experiments results are sum-up in Fig. 2. We use a deep neural network with three hidden layers the size of which has been set to 1024, 512, 64 through a validation procedure on the LFPW validation set. The input representation size is: $50 \times 50 = 2500$, and the output representation size is: $68 \times 2 = 136$. The notation DNN 0-0-0 stands for no pre-training at all; DNN 2-0-0 stands for 2 pre-trained input layers and DNN 2-1-1 stands for 2 pre-trained input layers and 1 pre-trained output layer (as exposed in IODA). To be fair, for all the pre-training strategies the initial weights are the same.

The CDF_{NRMSE} represents the percentage of images with error less or equal than the specified NRMSE value. For example a $CDF_{0.1} = 0.4$ over a test set means that 40% of the test set images have an error less or equal than 0.1. A CDF curve can be plotted according to these CDF_{NRMSE} values by varying the value of $NRMSE$.

Here, the IODA strategy, i.e. with input and output pre-training, achieves the best results.



(a) LFPW



(b) HELEN

Figure 2. CDF curves of best multiple configurations on: (a) LFPW, (b) HELEN.

Acknowledgement

This work has been partly supported by the ANR-11-JS02-010 project LeMon.

References

Belharbi, S., Chatelain, C., Herault, R., and Adam, S. Input/Output Deep Architecture for Structured Output Problems. *ArXiv 1504.07550*, April 2015. Submitted to ECML 2015.

Belhumeur, Peter N., Jacobs, David W., Kriegman, David J., and Kumar, Neeraj. Localizing parts of faces using a consensus of exemplars. In *CVPR*, pp. 545–552. IEEE, 2011.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy Layer-Wise Training of Deep Networks. *NIPS*, pp. 153–160, 2007.

Blaschko, MB. and Lampert, CH. Learning to Localize Objects with Structured Output Regression. In *ECCV 2008*, pp. 2–15, 2008.

Cootes, T.F., Taylor, C.J., Cooper, D.H., and Graham, J. Active shape models-their training and application. *CVIU*, 61(1):38–59, 1995.

Hinton, G. E., Osindero, S., and Teh, Y.-W. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554, July 2006.

Labbe, B., Herault, R., and Chatelain, C. Learning Deep Neural Networks for High Dimensional Output Problems. In *ICMLA, Miami, USA*, pp. 6p, 2009.

Lerouge, J., Herault, R., Chatelain, C., Jardin, F., and Modzelewski, R. IODA: An Input Output Deep Architecture for image labeling. *Pattern Recognition*, 2015.

Saragih, J., Lucey, S., and Cohn, J. Deformable Model Fitting by Regularized Landmark Mean-Shift. *IJCV*, 2011.

Weston, J., Chapelle, O., Vapnik, V., Elisseeff, A., and Schölkopf, B. Kernel dependency estimation. In *NIPS*, pp. 873–880, 2002.

Weston, J., Ratle, F., and Collobert, R. Deep learning via semi-supervised embedding. *ICML*, pp. 1168–1175, 2008.

Yu, X., Huang, J., Zhang, S., Yan, W., and Metaxas, D. Pose-Free Facial Landmark Fitting via Optimized Part Mixtures and Cascaded Deformable Shape Model. In *ICCV*, pp. 1944–1951, 2013.

Zhou, F., Brandt, J., and Lin, Z. Exemplar-Based Graph Matching for Robust Facial Landmark Localization. In *ICCV*, pp. 1025–1032, 2013.

Zhu, X. and Ramanan, D. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, pp. 2879–2886. IEEE, 2012. ISBN 978-1-4673-1226-4.