ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8.
Available from http://www.i6doc.com/en/.

Deep Multi-Task Learning with evolving weights

Soufiane Belharbi[1], Romain Hérault[1], Clément Chatelain[1] and Sébastien Adam[2] *

1- INSA de Rouen - LITIS EA 4108
Saint Étienne du Rouvray 76800 - France

2- Université de Rouen, UFR des Sciences - LITIS EA 4108
Saint Étienne du Rouvray 76800 - France

Abstract. Pre-training of deep neural networks has been abandoned in the last few years. The main reason is the difficulty to control the over-fitting and tune the consequential raised number of hyper-parameters. In this paper we use a multi-task learning framework that gathers weighted supervised and unsupervised tasks. We propose to evolve the weights along the learning epochs in order to avoid the break in the sequential transfer learning used in the pre-training scheme. This framework allows the use of unlabeled data. Extensive experiments on MNIST showed interesting results.

1 Introduction

In many real life applications, acquiring unlabeled data is easy and cheap in the opposite of labeled data, where manual annotation costs money and time. Many approaches, such as semi-supervised learning, have been adopted to benefit from unlabeled data as an *inductive bias* to improve the *generalization* of the learned model [1, 2, 3, 4]. Most of these algorithms are based on: (i) a *sequential transfer learning* where unsupervised training is done *separately* using the unlabeled data followed by a supervised training. (ii) *shallow* architectures where there is only one or two transformations of the input to predict the output.

Deep neural networks (DNN) can also benefit from unlabeled data. Usually, only one task is performed at each hidden layer. One can think of involving multiple tasks at each layer [5, 6]. Sharing the hidden representations allows learning better features for the generalization. [7, 8, 9, 10] proposed an unsupervised auxiliary task based on a layer-wise training which leads to the concept of *pre-training* which is a sequential transfer learning consisting of an unsupervised task followed by a supervised task. Tow main drawbacks to this scheme are: (i) the difficulty to control the over-fitting that may happen in the unsupervised task which damages the parameters used for the supervised task. (ii) the large number of hyper-parameters one needs to tune. One cause is that both tasks are optimized *separately.* A natural way to solve this issue is to learn both tasks *simultaneously.*

The work presented in this paper is mainly inspired by [11], where the authors propose to code an auxiliary task in each layer based on similarity preservation and manifold assumption. It is a regulation scheme based on *parallel transfer* in Multi-Task Learning (MTL) in place of the traditional pre-training technique.

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

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

Here, a similar MTL framework learns an unsupervised and a supervised tasks simultaneously except that the auxiliary unsupervised task is achieved by a set of auto-encoder reconstruction functions. Auto-encoders follows the work of [12] who combines the idea of input corruption [13, 14] with layer-wise training which leads to the denoising.

Moreover, we propose to balance both tasks using evolving weights along the learning epochs. Thus, the traditional pre-training scheme is a special case of our framework when setting the weights in a particular setup.

Our approach allows easily a better generalization and fast training of DNN. It gives interesting results on MNIST dataset.

2 Proposed model

The approach is formulated as a multi-task learning (MTL) framework [6] that gathers a main and a secondary task. Let us consider a training set $\mathcal{D} = \{(x_1, y_1), \dots, (x_l, y_l), (x_{l+1}, —), \dots, (x_{l+u}, —)\}$ where the l first examples are labeled and the u last examples are unlabeled; $\mathcal{M}$ and $\mathcal{R}$ the prediction/regression function of the main and secondary task, respectively; $\mathcal{C}_s$ and $\mathcal{C}_r$ their respective costs.

The main task is a supervised task with parameters $\mathbf{w} = \{\mathbf{w}_{sh}, \mathbf{w}_s\}$ where $\mathbf{w}_s$ is a set of parameters proper to the main task; and with criterion $\mathcal{J}_s$,

$$\mathcal{J}_s(\mathcal{D}; \mathbf{w} = \{\mathbf{w}_{sh}, \mathbf{w}_s\}) = \sum_{i=1}^{l} \mathcal{C}_s(\mathcal{M}(x_i; \mathbf{w}), y_i) \ . \tag{1}$$

The secondary task is a *reconstruction* task with parameters $\mathbf{w}' = \{\mathbf{w}_{sh}, \mathbf{w}_r\}$ where $\mathbf{w}_r$ is a set of parameters proper to the reconstruction task; and with criterion $\mathcal{J}_r$,

$$\mathcal{J}_r(\mathcal{D}; \mathbf{w}' = \{\mathbf{w}_{sh}, \mathbf{w}_r\}) = \sum_{i=1}^{l+u} \mathcal{C}_r(\mathcal{R}(x_i; \mathbf{w}'), x_i) \ . \tag{2}$$

Both tasks *share* the set of parameters $\mathbf{w}_{sh}$. Our purpose in using both tasks in the same framework, is that we hope that the secondary task improves the main task. The importance of both tasks is balanced using the importance weights λ_s and λ_r for the main and secondary task, respectively.

The full objective of our model can be written as,

$$\mathcal{J}(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s, \mathbf{w}_r\}) = \lambda_s \cdot \mathcal{J}_s(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_s\}) + \lambda_r \cdot \mathcal{J}_r(\mathcal{D}; \{\mathbf{w}_{sh}, \mathbf{w}_r\}) \ . \tag{3}$$

We propose to evolve the importance weights λ_s and λ_r along the optimization iterations. The intuition is to give more importance to the secondary task in the first learning epochs, but keep the main task present to avoid large damage of $\mathbf{w}_{sh}$. The main task is the final target, thus, the importance of the main task is increased and the importance of the secondary task is decreased through the learning epochs (Fig.1).

The new criterion becomes,

$$\mathcal{J}(\mathcal{D};\{\mathbf{w}_{sh},\mathbf{w}_s,\mathbf{w}_r\}) = \lambda_s(t)\cdot\mathcal{J}_s(\mathcal{D};\{\mathbf{w}_{sh},\mathbf{w}_s\}) + \lambda_r(t)\cdot\mathcal{J}_r(\mathcal{D};\{\mathbf{w}_{sh},\mathbf{w}_r\}) \ , \quad (4)$$

where $t \geq 0$ indicates the learning epochs. For the sake of a fair comparison between different models in the experiments, weights are constrained such that $\forall t \geq 0$: $0 \leq \lambda_s(t) \leq 1$, $0 \leq \lambda_r(t) \leq 1$ and $\lambda_s(t) + \lambda_r(t) = 1$. These conditions are not *mandatory* in an MTL [6, 15]. By doing this, we make sure that the observed benefit in our approach is not due to any boost of the learning rate in the optimization of Eq.4, as the learning rate has a direct relation with the importance weight.

3 Implementation details

3.1 The reconstruction task

In practice, the main task is achieved by a neural network NN with K hidden layers. The secondary task is achieved by a set of K reconstruction functions, each one is represented by a denoising auto-encoder (DAE). We recall that a DAE is a 2-layers neural network (a coding layer followed by a decoding layer). The k^{th} DAE has a set of parameters $\mathbf{w}_{dae,k} = \{\mathbf{w}_{c,k}, \mathbf{w}_{d,k}\}$ where $\mathbf{w}_{c,k}$ and $\mathbf{w}_{d,k}$ are respectively the parameters of the coding and decoding layers, and a noise function f (a binomial for instance) which is used to corrupt the input. In this case, all the parameters $\mathbf{w}_{c,k},\ 1 \leq k \leq K$ are *shared* with the supervised task which allows a *parallel transfer learning*. We set $\mathbf{w}_r = \{\mathbf{w}_{dae,1}, \ldots, \mathbf{w}_{dae,K}\}$. Each DAE is provided with its own cost function $\mathcal{C}_{dae,k}$. The criterion of the secondary task can be written in this case as,

$$\mathcal{J}_r(\mathcal{D};\mathbf{w}_r) = \sum_{i=1}^{l+u}\sum_{k=1}^{K}\mathcal{C}_{dae,k}(\mathcal{R}(f(x_{i,k-1});\mathbf{w}_{dae,k}), x_{i,k-1}) \ , \quad (5)$$

where $x_{*,k}$ is the representation at the k^{th} layer of NN ($x_{*,0}$ is the original input). For the sake of simplicity, we consider that all the reconstruction functions have the same importance weight $\lambda_r(t)$. One may think to associate different importance weights to each one.

3.2 Evolution of the importance weights along learning

Four different ways to evolve the importance weights through learning epochs (Fig.1) are studied in this work: • ***Stairs schedule***: the traditional pre-training scheme referenced as $stairs_{t_1}$ where $\lambda_s(t) = 0$ and $\lambda_r(t) = 1$ before an iteration t_1 and $\lambda_s(t) = 1$ and $\lambda_r(t) = 0$ after. • ***Linear schedule***: the weights progress linearly from the epoch 0 to the last training epoch; This case is referenced as *lin*. • ***Abridged linear schedule***: The linear trend can be stopped at an iteration t_1 then $\lambda_s(t)$ and $\lambda_r(t)$ are respectively saturated to 1 and 0 after that. This case is referenced as lin_{t_1}. • ***Exponential schedule***: the weights evolve exponentially proportionally to $exp^{\frac{-t}{\sigma}}$, where t is the current number of epochs, σ is the slope. This case is referenced as exp_{σ}.

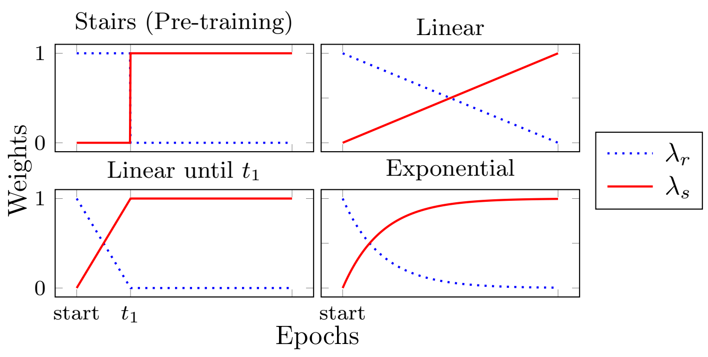

Fig. 1: Evolution of importance weights along training epochs

3.3 Optimization

Eq.4 is minimized by Stochastic Gradient Descent (SGD). Usually, in a case where one has multiple tasks in one single objective, alternating between task works well [6, 11, 16, 15]. The optimization technique is illustrated in Alg.1.

Algorithm 1 Training our model for one epoch

```
1: $\mathcal{D}$ is the *shuffled* training set. $B$ a mini-batch.
2: for $B$ in $\mathcal{D}$ do
3:     $B_s \Leftarrow$ labeled examples of $B$,
4:     Make a gradient step toward $\mathcal{J}_r$ using $B$ (update $\mathbf{w}'$)
5:     Make a gradient step toward $\mathcal{J}_s$ using $B_s$ (update $\mathbf{w}$)
6: end for
```

4 Experiments

We evaluate our approach on the MNIST dataset for the classification task using a similar protocol as in [11]. All the networks have the same input (28^2) and output (10). We refer to each network by NN_K where K is the number of hidden layers. We use the following networks: • NN_1: with size 50. • NN_2: with size: 60, 50. • NN_3: with size: 70, 60, 50. • NN_4: with size: 80, 70, 60, 50.

Each network NN_K is optimized once with only a supervised criterion $\mathcal{J}_s$. The resulting classification test error is denoted as the *baseline error*. Starting from the very same random neuron weights, each network NN_K is also optimized with multi-task criteria following different weight schedules: $stairs_{100}$ (traditional pre-training), lin_{100}, lin and exp_{40}. All the trainings end after 5000 epochs using a mini-batch size of 600 and a constant learning rate (0.01) which is decreased over the last 500 epochs. Only for the $stairs_{100}$ schedule, as in traditional pre-training setup, we use a learning rate which is optimized on a validation set to train the denoising auto-encoders. We used a custom version of Crino [17] for all the experiments. The error difference with the baseline error is displayed in Tab.1 for different sizes l and u of respectively labeled and unlabeled sets (negative means better than baseline). For deeper networks, we present only results using the schedule exp_{40}, due to the lack of space.

In the case of shallow network (Tab.1a), one can see that our approach improves the results using different schedules with a better global performance using the exp_{40} schedule. One can also notice that the more we add labeled data, the less we improve the result using our approach. In that case, we observe high values in the shared parameters $\mathbf{w}_{sh}$; that can be overcome throught an l_1, l_2 regularization which is considered for future work. We notice also that we do not improve the performance when using only labeled data ($u = 0$). One explanation to this is that the observed improvement of our approach is mainly due to the information contained in the extra unlabeled data.

In the case of deeper networks (Tab.1b), one can observe a global improvement of the performance using our approach. We observe the same pattern when using larger labeled data, but we obtain lower improvement.

Table 1: Classification error over MNIST test.
(Figures are in percentage and relative to their corresponding baseline error)
(a) *Shallow* neural network

Labeled	$l = 100$				$l = 10^3$				$l = 10^4$			
Architecture	$\boldsymbol{NN_1}$				$\boldsymbol{NN_1}$				$\boldsymbol{NN_1}$			
Baseline Error	31%				12.85%				9.08%			
Schedule / Unlabeled	$\boldsymbol{stairs_{100}}$	$\boldsymbol{lin_{100}}$	$\boldsymbol{lin}$	$\boldsymbol{exp_{40}}$	$\boldsymbol{stairs_{100}}$	$\boldsymbol{lin_{100}}$	$\boldsymbol{lin}$	$\boldsymbol{exp_{40}}$	$\boldsymbol{stairs_{100}}$	$\boldsymbol{lin_{100}}$	$\boldsymbol{lin}$	$\boldsymbol{exp_{40}}$
$u = 0$	+0.03	+0.03	+1.32	+0.03	+0.03	0.0	+1.5	+0.05	+0.03	+0.01	+0.95	-0.04
$u = 10^3$	-0.1	-0.87	-0.19	-0.85	+0.09	-0.71	+0.33	-0.73	+0.03	-0.08	+0.85	-0.08
$u = 2 * 10^3$	-0.17	-1.35	-1.03	-1.46	+0.009	-0.98	-0.23	-0.95	+0.03	-0.2	+0.74	-0.21
$u = 5 * 10^3$	-0.16	-1.69	-1.78	-1.91	+0.05	-1.43	-0.95	-1.44	+0.03	-0.26	+0.31	-0.3
$u = 10^4$	-0.28	-2.21	-2.03	-2.21	0.0	-1.6	-1.12	-1.44	+0.01	-0.5	-0.1	-0.51
$u = 2 * 10^4$	-0.28	-1.82	-2.26	-2.42	-0.02	-1.83	-1.46	-1.78	+0.01	-0.63	-0.44	-0.75
$u = 4 * 10^4$	-0.46	-2.39	-2.62	-2.44	-0.049	-1.63	-1.37	-1.7	+0.01	-0.63	-0.63	-0.56
$u = 5 * 10^4 - l$	-0.19	-1.89	-2.03	-2.44	-0.06	-1.68	-1.56	-1.75	+0.01	-0.63	-0.63	-0.56

(b) *Deeper* neural networks

Labeled	$l = 100$			$l = 10^3$			$l = 10^4$		
Architecture	$\boldsymbol{NN_2}$	$\boldsymbol{NN_3}$	$\boldsymbol{NN_4}$	$\boldsymbol{NN_2}$	$\boldsymbol{NN_3}$	$\boldsymbol{NN_4}$	$\boldsymbol{NN_2}$	$\boldsymbol{NN_3}$	$\boldsymbol{NN_4}$
Baseline Error	31.48%	33.58%	30.53%	12.32%	11.96%	12.54%	5.62%	5.69%	5.5%
Schedule / Unlabeled	$\boldsymbol{exp_{40}}$			$\boldsymbol{exp_{40}}$			$\boldsymbol{exp_{40}}$		
$u = 0$	+0.01	-0.01	+0.03	+0.02	-0.08	-0.03	-0.11	-0.03	+0.08
$u = 10^3$	-1.05	-0.83	-0.92	-0.55	-0.58	-0.8	-0.2	-0.07	+0.12
$u = 2 * 10^3$	-2.03	-1.35	-1.17	-0.78	-0.77	-1.18	-0.4	-0.01	+0.04
$u = 5 * 10^3$	-2.32	-1.71	-1.31	-1.15	-1.23	-1.56	-0.54	-0.18	-0.04
$u = 10^4$	-2.26	-1.29	-0.72	-1.14	-1.29	-1.76	-0.75	-0.22	-0.24
$u = 2 * 10^4$	-2.4	-0.78	-0.82	-1.25	-1.48	-1.88	-0.81	-0.41	-0.08
$u = 4 * 10^4$	-2.12	-1.17	-1.65	-1.08	-1.3	-1.92	-0.81	-0.42	-0.55
$u = 5 * 10^4 - l$	-1.99	-1.23	-0.88	-1.23	-1.3	-1.85	-0.81	-0.42	-0.55

5 Conclusion

We presented in this paper a new learning scheme for deep neural networks where we used a multitask learning framework that gathers a supervised and unsupervised task. We proposed to evolve the weights of the tasks along the learning epochs using different schedules. Using less hyper-parameters, we improved the performance of deep neural networks easily. As a future work, we consider using l_1, l_2 regularization of the shared parameters. In the aim to set an automatic framework, we consider using an early stopping on the reconstruction task based on the error on the train and validation set.

References

[1] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-supervised learning*. Adaptive computation and machine learning. MIT Press, 2006.

[2] O. Chapelle, J. Weston, and B. Schölkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing 15, NIPS 2002*, pages 585–592, 2002.

[3] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, CMU-CALD-02-107, Carnegie Mellon university, 2002.

[4] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning*, 7:2399–2434, 2006.

[5] S. C. Suddarth and Y. L. Kergosien. Rule-injection hints as a means of improving network performance and learning time. In *Neural Networks, EURASIP workshop 1990*, pages 120–129, 1990.

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

[7] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[8] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[9] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural information Processing Systems 19, NIPS 2006*, pages 153–160, 2006.

[10] M. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun. unsupervised learning of invariant feature hierarchies with applications to object recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2007*, 2007.

[11] J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In *Machine Learning, Proceedings of 25th International Conference, ICML 2008*, pages 1168–1175, 2008.

[12] P. Vincent, H. Larochelle, I. Lajoie, and Y. Bengio. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408, 2010.

[13] L. Holmstrom and P. Koistinen. Using additive noise in back-propagation training. *Neural Networks, IEEE Transaction*, 3(1):24–38, 1992.

[14] J. Sietsma and R. J. F. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 4(1):67–79, 1991.

[15] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *Computer Vision, ECCV 2014, 13th European Conference*, pages 94–108, 2014.

[16] R. Collobert and J. Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of he 25th International Conference, ICML 2008*, pages 160–167, 2008.

[17] Crino, a neural-network library based on Theano. `https://github.com/jlerouge/crino`, 2014.