Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2017) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

Information Technology and Quantitative Management (ITQM 2016)

Hybrid learning net: a novel architecture for fast learning

Ying Liu[a], Chao Xiang[a]

[a]*University of Chinese Academy of Sciences, Beijing 100049, China*

Abstract

Currently, neural networks have succeeded in object recognition tasks based on images, natural language translation, and voice recognition, to name a few. However, neural nets are customly built for different applications and vary a lot in achitectures and model hyperparameters like learning rate and parameter initialization, what's worse, these hyperparameter settings generally play a big role in performances of training and testing, and the best settings for specific applications are so far only available by manually repeatly trying different configurations which is really huge work. We, thereby, present a novel neural network achitecture, called Hybrid Learning Net(**HLN**), with Self Organizing Maps(SOMs) embedded in each layer to learn from samples in **both** unsupervised and supervised way, targeting to achieve a much **faster** net learning for general applications with good robustness to a few key hyperparameters such as the parameter initialization and the net strcuture variation. We've also experimented our architecture over the MNIST dataset, it has proved the impressive improvement on both training and testing phases of general applications, say compared to the traditional architecture, our method speed up the training process by up to **40** times, which only take **1** epoch to get an testing accuracy of over **87**.**5**%, and takes no more than **3** epoches to reach a profound accuracy of over **91**.**3**%. In addition, on big scale of input dimension and/or with deeper architecture, where the traditional architecture fails to learn at all, our method still have a fast learning, and can retrieve the same testing accuracy on MNIST. Moreover, we have discovered some interesting facts about neural network trainings, such as neuron activation sparsity is strongly correlated to the training loss within certain cases which may shed a little light on how such architecture really works.

Selection and/or peer-review under responsibility of ITQM2016.

Keywords: hybrid learning; neural network; general application; sparsity; SOM; MNIST; fast learning

1. Introduction

Since the first mathematical model of artificial neural network was proposed in 1943[1], the neural network has been designed into many architectures, such as Convolutional Neural Networks(CNNs) for image recognition [2], Recurrent Convolutional Neural Networks(R-CNNs) for object detection in videos[3], and Long Short Term Memorys(LSTMs) for speech recognition [4] and many, many more to make it a list. These specially designed neural network models are trained by lots of efficient methods with tons of carefully chosen little skills which we may call tricks. Such neural network architectures suit well for their specific applications but may have plain or worse performances on others, and their best performances rely heavily on hyperparameter configuration in

*Corresponding author. Tel.: +86-183-0114-2368;
E-mail address: xiangchao215@mails.ucas.ac.cn.

general. Thus, it is an in-demand job to propose a relatively universal architecutre that enables equal or similar performances among varied applications.

We notice that, although there're plenty of choices to train a neural network, literally all these methods can be reduced into 3 categories, supervised learning [5], unsupervised learning [6], and the semi-supervised [7]. In supervised learning, one can only train a model from labeled samples, however in real applications, labeling a large dataset is a tough and costly task, the unlabeled ones are therefore the primary data available. To make use of the majority unlabeled data, a few unsupervised training algorithms arose. These unsupervised learning methods, denoted as pretraining, attempt to produce an optimized parameter initialization [8]. Thus such unsupervised techniques can only be applied before the supervised training phase, once the supervised begins, the unsupversied learning will become unavailable for the model. While the semi-supervised learning allows the model to learn from both labeled and unlabeled samples at the same time, however they use a regularizer to embed the semi-supervised learner to original optimizing object, and generally a balance constraint is required to avoid the trival solution [9]. Such a enhanced learner brings more hyperparameters(say the balance constraint), making it even more difficult to search for best settings for current architectures. Additionally, this integration way makes it impossible to separate the two learners as the semi-supervised regularizer is built on the supervised learner, and therefore requires an early supervised training alone with profound labeled data before the semi-supervised regularizer can be applied. Thus we introduce a new architecture that learn in both supervised and unsupervised ways at the same time, and requires no extra techniques on training. Our architecture, Hybrid Learning Nets(HLNs), made of stacked layers, with each hidden layer embedded with a Self Organizing Map(SOM), training in the simplest way of backprop, demonstrate much faster learning capability and remain robust to parameter initialization and network configuration such as the net depth of layer.

The main contributions of our work are:

- We propose HLN, a SOM-embedded architecture to learn both unlabeled and labeled data at the same time.
- Our architecture HLN overcomes the problem brought by traditional semi-supervised learning methods that the semi-supervised learning requires an early standalone training for the supervised learner which contradicts with the key condition: no enough labeled data is provided.
- The HLN uses a SOM embedding coupling the first hidden layer near to the input layer, to learn a cluster mapping function from the cheap and massive unlabeled data, and this process can occur at any time, no matter the supervised learning begins or not, they're completely separated. For deeper hidden layers, each of them also has a SOM-embedding coupled, but can only learn from labeled data.
- The experimental result shows HLNs speed up the whole training a lot.
- Our architecture demonstrates a good robustness to some hyperparameters which may slightly relax the work of manually searching for best settings of hyperparameters by repeatly trying different configurations and run the whole training and testing over and over again.

HLN is implemented in Python and all our code and results of experiments in detail are available at `https://github.com/hiroki-kyoto/hybrid-learning-net`.

The rest of the article is as follows. In section 2 we describe existing semi-supervised algorithms for neural network models, recall the SOM and explore a different training method for SOM when applied in the neural network embedding. In section 3, we introduce our novel architecture HLN, show how to embed SOMs into deep architectures of neural nets. In section 4 we explain the exact training theory for HLNs. Section 5 gives experimental comparisons between nets with HLN architecture and without, and the last section concludes.

2. Related work and backgrounds

2.1. Semi-supervised learning for neural network

A key assumption in semi-supervised algorithms developed so far, is the structure assumption: two samples with similar distribution on the same mapping structure tend to have high probability of belonging to the same class. Based on this assumption, one can use large unlabeled data to uncover such structures. There're already a few algorithms dedicated to do this, such as cluster kernels [10], Low Density Separation(LDS) [11], label propagation [12], to name a few. In such algorithms, designing a regularizer to enable the model to learn the

representation or structure of raw data, in order to improve the supervised learning performance, becomes the key point.

Let's firstly focus on the general algorithm description of semi-supervised learning. Given a set of unlabeled samples, $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}(\mathbf{x} \in \mathbb{R}^d)$, and the similarity labels between any $\mathbf{x}_i$ and $\mathbf{x}_j$, $\mathbf{W} = \{W_{ij}|i, j = 1, \cdots, N\}$, we're to find the best embedding function, $f(\mathbf{x})$, for each sample $\mathbf{x}_i$, to minimize:

$$\Delta_f = \sum_{i=1}^{N} \sum_{j=1}^{N} L\Big(f(\mathbf{x}_i), f(\mathbf{x}_j), W_{ij}\Big) \tag{1}$$

To explain it,

- $L(\cdot)$ is the loss function of 3 variables: $\Big\langle f(\mathbf{x}_i), f(\mathbf{x}_j), W_{ij} \Big\rangle$, such as

$$L\Big(f(\mathbf{x}_i), f(\mathbf{x}_j), W_{ij}\Big) = \max\Big(0, \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\| - W_{ij}\Big)$$

- $f(\mathbf{x}) \in \mathbb{R}^n$ is the embedding function, it tries to produce a vector from $\mathbf{x}_i$, similar to that of $\mathbf{x}_j$ with $W_{ij} = 0$, and disimilar with $W_{ij} = 1$.
- $W_{ij} \in \mathbb{R}$ is the similarity label of the sample pair $\Big\langle \mathbf{x}_i, \mathbf{x}_j \Big\rangle$ from $\mathbf{X}$.

Label Propagation(LP) [12] is one of the most efficient algorithms using the semi-supervised learning scheme as described above in equation 1. It adds a Laplacian Eigenmap type regularization to a nearest neighbor classifier:

$$\min_f \sum_{i=1}^{L} \left\| \vec{f_i} - \hat{\mathbf{y}}_i \right\|^2 + \lambda \sum_{i=1}^{L+U} \sum_{j=1}^{L+U} W_{ij} \left\| \vec{f_i} - \vec{f_j} \right\|^2 \tag{2}$$

L is a labeled sample set of $\mathbf{X}$, and U is a unlabeled one, the right part of equation 2 is the semi-supervised regularizer, the left part is for supervised learning only. Parameter λ is the balance constraint. LP trains the classifier $f(\cdot)$ to give two examples with high similarity W_{ij} the same label, and the neighbors of neighbors tend to get the same label by transitivity. The two points make its name *label propagation*.

For neural network model, we replace $f(\cdot)$ with equation as follow,

$$y_i = \sum_j w_j^{M+1,i} y_j^M(\mathbf{x}) + b^{M+1,i}, \quad i = 1, \cdots, D \tag{3}$$

and such that

$$f(\mathbf{x}) = \vec{y} = (y_1, y_2, \cdots, y_D) \tag{4}$$

The equation 3 describe the simplest neural model with M hidden layers, D is the output dimension, $f_i(\mathbf{x})$ computes the output of i^{th} neuron in the $(M+1)^{th}$(index starts from 0) layer(the output layer of the net model) and y_j^M is the j^{th} hidden neuron on M^{th} layer, $w_j^{M+1,i}$ is the connection weight from j^{th} neuron in M^{th} layer to i^{th} neuron in output layer. $b^{M+1,i}$ is the bias for the i^{th} neuron in $(M+1)^{th}$ layer. To get y_j^M, just follow the equation as

$$y_i^k(\mathbf{x}) = \sigma\left(\sum_j w_j^{k,i} y_j^{k-1} + b^{k,i}\right), k > 1 \tag{5}$$

and when it comes to the first hidden layer,

$$y_i^1(\mathbf{x}) = \sigma\left(\sum_j w_j^{1,i} x_j + b^{1,i}\right) \tag{6}$$

σ can be any non-linear function, such as the sigmoid $(1 + e^{-x})^{-1}$, $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ and the latest ReLU $\sigma(x) = \max(0, x)$ and a few more. Notice that a nonlinear activation function is often required when the neural network model is used in a classification application.

2.2. Traditional way for semi-supervised embedding

As described in last subsection, the traditional way to embed semi-supervised learning ability into a neural net is through a weighted regularizer, which is practically adding a semi-supervised loss Δ_f onto the supervised learning loss [13], thus the learning problem turns to be minimizing:

$$\sum_{i=1}^{L} \ell\left(f(\mathbf{x}_i), \hat{\mathbf{y}}_i\right) + \lambda \sum_{i=1}^{L+U} \sum_{j=1}^{L+U} L\Big(f(\mathbf{x}_i), f(\mathbf{x}_j), W_{ij}\Big) \tag{7}$$

and in most efficient algorithms Euclidean metric is used for the loss.

2.3. Self Organizing Map

The Self Organizing Map(SOM) is an effective software tool for the visualization of high-dimensional data. It can also be used as an automatic clustering method. The SOM consists of a two-dimensional regular grid of nodes. The models are automatically organized into a meaningful two-dimensional order in which similar models are closer to each other in the grid than the more dissimilar ones[14]. It use such rules to update models:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{c(x),i}\left(\mathbf{x}(t) - \mathbf{m}_i(t)\right) \tag{8}$$

and the learning rate is dynamically determined as

$$h_{c(x),i} = \alpha(t) \exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_c\|^2}{2\sigma^2(t)}\right) \tag{9}$$

where $\mathbf{m}_i \in \mathbb{R}^n$ is the i^{th} model vector, $\mathbf{x}$ is an input pattern, $c(x)$ relates to the best match vector index in $\mathbf{m}$ for input pattern $\mathbf{x}$, and $\alpha(t)$ is a learning rate that decreases with training preceeding. $\mathbf{r}$ is the model vector location in the map, and $\sigma(t)$ corresponds to the width of the neighborhood function, which also decreases monotonically with the regression steps.

3. Our architecture: the Hybrid Learning Net

We propose the Hybrid Learning Net(HLN) as an architecture to enhance arbitrary nets on their training efficiency and robustness to some hyperparameters. Each layer in the HLN architecture embeds a SOM into its original layer, for the simplest situation where neurons connected in fully connected way, which we call the Fully Connected Neurons(FCN) architecture, we have such an embedding solution as described in figure 1.

In the HLN architecture from figure 1, $h(x)$ is an unifying function we proposed for SOMs to convert from pattern disimilarity $\|\mathbf{x}-\mathbf{m}_i\|$, into a semi-supervised learning factor for different hidden units. The semi-supervised learning factors as a whole act as a dynamic neuron activation sparsity mask for each hybrid learning layer. It works in a way like the Dropout technique[15], enabling the neural nets to improve the model robustness and prevent overfitting by learning their submodels for each batch. However, the HLN differs from techniques like the Dropout in which, the HLN does not generate sparsity with randomness, she uses the SOM to unsupervisedly learn the *static* policy of neuron-activation distribution for each layer, the net sparsity generator thereby will stablize with training steps, and the randomness in sparsity will disappear automatically. We propose $h(x)$ with the form as following,

$$\delta_{max} = \max_i \left(\|\mathbf{m}_i - \mathbf{x}\| + \epsilon\right) \tag{10}$$

$$\delta_{min} = \min_i \left(\|\mathbf{m}_i - \mathbf{x}\| + \epsilon\right) \tag{11}$$

$$\delta_{scale} = \frac{\mathbf{m}_{min}}{\mathbf{m}_{max} - \mathbf{m}_{min} + \epsilon} \tag{12}$$

$$h(x)|_{x=\|\mathbf{m}_i-\mathbf{x}\|} = \frac{\delta_{max} \cdot \delta_{scale}}{\|\mathbf{m}_i - \mathbf{x}\| + \epsilon} - \delta_{scale} \tag{13}$$

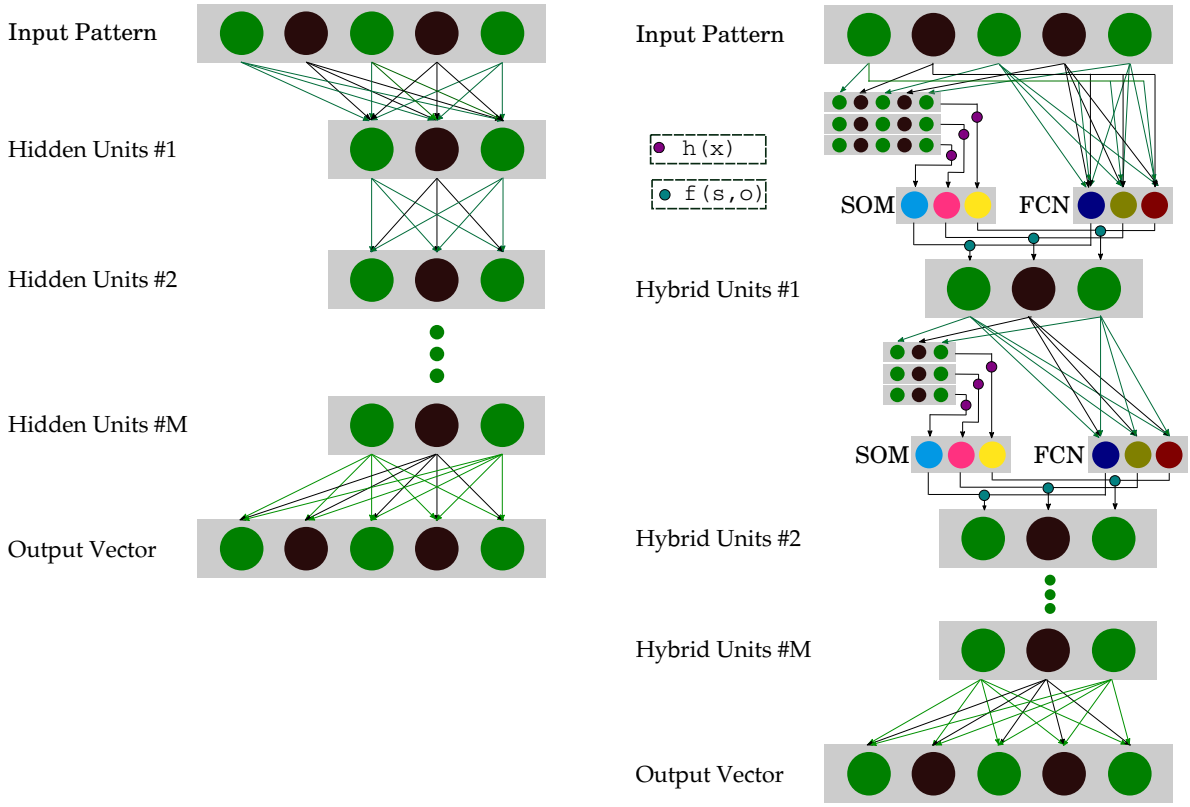

Fig. 1. (a) FCN architecture; (b) HLN architecture.

in which $\mathbf{m}_i$ and $\mathbf{x}$ is the same way defined in equation 8. ϵ is a constant of pretty small value like $\epsilon = 10^{-5}$ to avoid *division-by-zero* errors that may, though not very likely occur. Notice that if any other metrics be preferred, we can always replace the pattern disimilarity $\|\mathbf{m}_i - \mathbf{x}\|$ with its corresponding form. $f(s, o)$ is the function to combine the sparsity mask s generated by $h(\cdot)$ and the fully connected linear summation output o. In most cases, we choose the multiplication operator,

$$f(s, o) = s \cdot o = h\left(\left\|\mathbf{m}_i - \mathbf{y}^{\mathbf{k-1}}\right\|\right) \cdot \sum_j \left(w_j^{k,i} y_j^{k-1}\right) + b^{k,i}, k > 1 \tag{14}$$

where w, y, b, i, j, k is the same denotion as in equation 5. For the computing flow described in figure 1, $f(\cdot)$ is followed by an activation function, it can be any arbitrary nonlinear function that takes only one dimension inputs and outputs a single real.

4. The training theory for HLNs

Let's compare the embedding theories between HLNs and the ones mentioned in the section of *Related work and backgrounds*. The existing semi-supervised algorithms embed a regularizer into the supervised learner, making it impossible to train both of the supervised learner and the unsupervised separately. As analyzed before, performances of such regularizer solutions depends on the standalone supervised pretraining for which a profound labeled data is required. However, in our architecture HLN, we assign each of the learning methods a completely separate optimizing object, with no priority orders restricted.

For the supervised learning, we may have such form of optimizing object as

$$\arg\min_{g} \sum_{i=1}^{L} \ell\left(g(\mathbf{x}_i), \hat{\mathbf{y}}_i\right) \tag{15}$$

where $g(\mathbf{x})$ is the function describing the mapping from the input $\mathbf{x} \in \mathbb{R}$ to the output $\mathbf{y} \in \mathbb{R}^{\aleph}$, parameterized with $\mathbf{W}$ of the HLN. This equation 15 may become the following one when Enclidean metrics be applied for the loss,

$$\arg\min_{g} \frac{1}{2} \sum_{i=1}^{L} \|g(\mathbf{x}_i) - \hat{\mathbf{y}}_i\|^2 \tag{16}$$

5. Empirical Study

To test our architecture performance on small data, we use a synthetic dataset containing 3.5% noisy samples. A sampling from this dataset get the

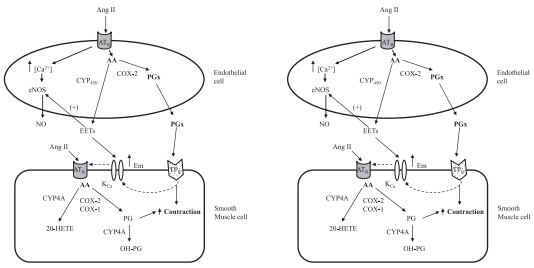

Fig. 2. (a) first picture; (b) second picture.

$$\begin{aligned} X_r &= \dot{Q}^{"}_{rad} \Big/ \left(\dot{Q}^{"}_{rad} + \dot{Q}^{"}_{conv}\right) \\ \rho &= \frac{\vec{E}}{J_c(T = \text{const.}) \cdot \left(P \cdot \left(\dfrac{\vec{E}}{E_c}\right)^{m} + (1 - P)\right)} \end{aligned} \tag{17}$$

They should also be separated from the surrounding text by one space.

6. Conclusions

All authors must Transfer the Online licence before the article can be published. This transfer agreement enables Elsevier to protect the copyrighted material for the authors, but does not relinquish the authors' proprietary rights. The copyright transfer covers the exclusive rights to reproduce and distribute the article, including reprints, photographic reproductions, microfilm or any other reproductions of similar nature and translations. Authors are responsible for obtaining from the copyright holder permission to reproduce any figures for which copyright exists.

Acknowledgements

These and the Reference headings are in bold but have no numbers. Text below continues as normal.

References

[1] W. S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bulletin of mathematical biology 5 (4) (1943) 115–133.

[2] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[3] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.

[4] A. Graves, N. Jaitly, A.-r. Mohamed, Hybrid speech recognition with deep bidirectional lstm, in: Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on, IEEE, 2013, pp. 273–278.

[5] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, L. D. Jackel, Handwritten digit recognition with a back-propagation network, in: Advances in neural information processing systems, 1990, pp. 396–404.

[6] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, Journal of Machine Learning Research 11 (Dec) (2010) 3371–3408.

[7] O. Chapelle, B. Scholkopf, A. Zien, Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews], IEEE Transactions on Neural Networks 20 (3) (2009) 542–542.

[8] Q. V. Le, Building high-level features using large scale unsupervised learning, in: Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, IEEE, 2013, pp. 8595–8598.

[9] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, C. D. Manning, Semi-supervised recursive autoencoders for predicting sentiment distributions, in: Proceedings of the conference on empirical methods in natural language processing, Association for Computational Linguistics, 2011, pp. 151–161.

[10] O. Chapelle, J. Weston, B. Schölkopf, Cluster kernels for semi-supervised learning, in: Advances in neural information processing systems, 2003, pp. 601–608.

[11] O. Chapelle, A. Zien, Semi-supervised classification by low density separation., in: AISTATS, 2005, pp. 57–64.

[12] X. Zhu, Z. Ghahramani, Learning from labeled and unlabeled data with label propagation.

[13] J. Weston, F. Ratle, H. Mobahi, R. Collobert, Deep learning via semi-supervised embedding, in: Neural Networks: Tricks of the Trade, Springer, 2012, pp. 639–655.

[14] T. Kohonen, The self-organizing map, Neurocomputing 21 (1) (1998) 1–6.

[15] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting., Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

Appendix A. An example appendix

Authors including an appendix section should do so after References section. Multiple appendices should all have headings in the style used above. They will automatically be ordered A, B, C etc.

Appendix A.1. Example of a sub-heading within an appendix

There is also the option to include a subheading within the Appendix if you wish.