Two-Layer Contractive Encodings with Linear Transformation of Perceptrons for Semi-Supervised Learning

Hannes Schulz[1], Kyunghyun Cho[2], Tapani Raiko[2], and Sven Behnke[1]

[1] Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik VI, Friedrich-Ebert-Allee 144, 53113 Bonn
{schulz, behnke}@ais.uni-bonn.de
[2] Department of Information and Computer Science
Aalto University School of Science, Finland
{kyunghyun.cho, tapani.raiko}@aalto.fi

Abstract. It is difficult to train a multi-layer perceptron (MLP) when there are only a few labeled samples available. However, by pretraining an MLP with vast amount of unlabeled samples available, we may achieve better generalization performance. Schulz et al. (2012) showed that it is possible to pretrain an MLP in a less greedy way by utilizing the two-layer contractive encodings, however, with a cost of a more difficult optimization problem. On the other hand, Raiko et al. (2012) proposed a scheme for making the optimization problem much easier in deep networks. In this paper, we show that it is beneficial to combine these two approaches.

Keywords: Multi-Layer Perceptron, Two-Layer Contractive Encoding, Linear Transformation, Semi-Supervised Learning

1 Introduction

Multi-layer perceptrons (MLP) are provably powerful at performing nonlinear classification or regression tasks [8]. However, until recently it was believed that it is difficult to train an MLP well to have a good generalization performance (see, e.g., [1]). In [7, 12, 3], greedy layer-wise pretraining which initializes the parameters of an MLP in unsupervised manner was proposed to overcome this problem.

Often, only a few labeled training samples are available together with vast amount of *unlabeled* samples. Semi-supervised learning [5] aims to utilize not only the labeled samples but also the unlabeled samples to improve generalization performance of a classifier. Pretraining of an MLP naturally fits well in this framework of semi-supervised learning, as the pretraining procedure does not require any labeled sample. In short, an MLP can be initialized with pretraining using the vast amount of unlabeled samples and subsequently finetuned with (few) labeled samples.

In this paper, we empirically evaluate an advanced procedure for training an MLP in the task of semi-supervised classification. Unlike the existing approach of greedily pretraining an MLP with unlabeled samples, we investigate pretraining an MLP in a less greedy way by pretraining *two* hidden layers at a time, as proposed in [15]. This approach, which requires training a regularized autoencoder with two hidden layers is made practical by utilizing the recently proposed method of linearly transforming perceptrons [10]. We empirically show that we can achieve a better classification accuracy on unseen test samples by using the two-layer contractive encoding and the linear transformation of hidden neurons, when only a few labeled samples are available.

In Sec. 2, we discuss in more detail the motivation behind our work. A two-layer contractive encoding and linear transformation of perceptrons which constitute the main parts of our proposed framework are described briefly in Sec. 3. The empirical evaluation of the propose framework is provided in Sec. 4, followed by the conclusion and discussion in Sec. 5.

2 Motivations

2.1 Training an MLP with Few Labeled Samples

In the framework of multi-layer perceptrons (MLP), unsupervised pretraining is a natural way to incorporate the vast amount of unlabeled samples (see, e.g., [7]). As the parameters of an MLP are initialized in an *unsupervised* manner, all available samples including both labeled and unlabeled ones may be exploited during pretraining. Once the parameters are initialized, we can finetune the whole model using only the labeled samples. This approach was adopted earlier by, for instance, [11] which showed that pretraining helps improving generalization performance when only a few labeled samples together with a large amount of unlabeled samples were available.

One prominent method of pretraining an MLP is a greedy layer-wise pretraining originally proposed in [7, 3, 12]. The greedy layer-wise pretraining sequentially trains each consecutive pair of layers of an MLP in unsupervised manner.

One hypothesis by Bengio et al. (see, e.g., [1, 2]) on why the greedy layer-wise pretraining helps in semi-supervised learning is that stacking of unsupervised neural networks disentangles factors of variations and that this disentangled factors have better discriminative property than raw variables. Unlabeled samples are used during unsupervised pretraining to better disentangle factors of variations, and labeled samples are subsequently used to find mapping between those factors and labels. Then, one way to improve this procedure of unsupervised pretraining and subsequent finetuning is to replace the oft-used layer-wise pretraining with a more sophisticated pretraining method.

2.2 Less Greedy Pretraining

Although greedy layer-wise pretraining has been found to be a good strategy to initialize the parameters of a multi-layer perceptron (MLP) with vast amount of unlabeled samples, recently in [15] it was found that this greedy strategy may fail to help an MLP to discover features that are inherently nonlinear. In certain cases, it was shown that greedy layer-wise pretraining could actually hurt the overall performance of an MLP.

Hence, in [15] a two-layer contractive encoding was proposed as a way to pretrain an MLP in a less greedy way. The experiments in [15] revealed that a better classification performance can be achieved by using the two-layer contractive encoding as a building block than by using a neural network with a single hidden layer.

It is, however, often more difficult to train an unsupervised neural network such as a deep autoencoder using a stochastic gradient method (SGD) without greedily pretraining each layer. Raiko et al. [10] proposed to linearly transform each hidden neuron, while also introducing connections that skip the hidden layer. They showed that in this way, it is possible to train a deep neural network directly with good generalization performance. This approach effectively makes SGD closer to second-order optimization method [16] without sacrificing its computational advantage.

In this paper, hence, we focus on the approach of pretraining an MLP less greedily. We aim to provide empirical evidence that one must use both the two-layer contractive encoding and the linear transformation order to achieve good initialized parameters using less greedy pretraining. In other words, we claim that a less greedy pretraining approach requires both well-founded regularization and powerful learning algorithm.

3 Background

In this section, we discuss each one of those two methods, the linear transformations and the two-layer contractive encoding, in more detail.

3.1 Linear Transformations in Perceptrons

Let us study a single hidden layer within a possibly deep MLP network. The inputs to this layer are marked with column vectors $\mathbf{x}_t$ and its outputs as column vectors $\mathbf{y}_t$, where t is the sample index. We allow short-cut connections that by-pass one or more hidden layers.[3] The mapping from $\mathbf{x}_t$ to $\mathbf{y}_t$ is modelled as

$$\mathbf{y}_t = \mathbf{A}\mathbf{f}\left(\mathbf{B}\mathbf{x}_t\right) + \mathbf{C}\mathbf{x}_t, \tag{1}$$

where $\mathbf{f}$ is a nonlinearity (such as $\tanh$) applied to each component of the argument vector separately, $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are the weight matrices. In order to avoid separate bias vectors that complicate formulas, the input vectors $\mathbf{x}_t$ are assumed to have been supplemented with an additional component that is always one.

Let us supplement the $\tanh$ nonlinearity with auxiliary scalar variables α_i and β_i for each nonlinearity f_i. They are updated during training in order to help learning of the other parameters $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. We define

$$f_i(\mathbf{b}_i\mathbf{x}_t) = \tanh(\mathbf{b}_i\mathbf{x}_t) + \alpha_i\mathbf{b}_i\mathbf{x}_t + \beta_i, \tag{2}$$

where $\mathbf{b}_i$ is the ith row vector of matrix $\mathbf{B}$. We will ensure that

$$0 = \sum_{t=1}^{T} f_i(\mathbf{b}_i\mathbf{x}_t), \qquad 0 = \sum_{t=1}^{T} f_i'(\mathbf{b}_i\mathbf{x}_t) \tag{3}$$

by setting α_i and β_i to

$$\alpha_i = -\frac{1}{T}\sum_{t=1}^{T}\tanh'(\mathbf{b}_i\mathbf{x}_t), \qquad \beta_i = -\frac{1}{T}\sum_{t=1}^{T}\left[\tanh(\mathbf{b}_i\mathbf{x}_t) + \alpha_i\mathbf{b}_i\mathbf{x}_t\right].$$

[3] Both the inputs and outputs of this particular layer may actually be on several layers.

These seemingly arbitrary update rules are motivated below.

The effect of changing the transformation parameters α_i and β_i are compensated exactly by updating the shortcut mapping $\mathbf{C}$ by

$$\mathbf{C}_{\text{new}} = \mathbf{C}_{\text{old}} - \mathbf{A}(\boldsymbol{\alpha}_{\text{new}} - \boldsymbol{\alpha}_{\text{old}})\mathbf{B} - \mathbf{A}(\boldsymbol{\beta}_{\text{new}} - \boldsymbol{\beta}_{\text{old}}) \left[0 \;\; 0 \ldots 1\right], \tag{4}$$

where $\boldsymbol{\alpha}$ is a matrix with elements α_i on the diagonal and one empty row below for the bias term, and $\boldsymbol{\beta}$ is a column vector with components β_i and one zero below for the bias term. Thus, any change in α_i and β_i does not change the overall mapping from $\mathbf{x}_t$ to $\mathbf{y}_t$ at all, but they do change the optimization problem instead.

One way to motivate the transformations in Equations (3), is to study the expected output $\mathbf{y}_t$ and its dependency on the input $\mathbf{x}_t$:

$$\frac{1}{T}\sum_t \mathbf{y}_t = \mathbf{A}\left[\frac{1}{T}\sum_t \mathbf{f}(\mathbf{B}\mathbf{x}_t)\right] + \mathbf{C}\left[\frac{1}{T}\sum_t \mathbf{x}_t\right] \tag{5}$$

$$\frac{1}{T}\sum_t \frac{\partial \mathbf{y}_t}{\partial \mathbf{x}_t} = \mathbf{A}\left[\frac{1}{T}\sum_t \mathbf{f}'(\mathbf{B}\mathbf{x}_t)\right]\mathbf{B}^T + \mathbf{C} \tag{6}$$

We note that by making nonlinear activations $\mathbf{f}(\cdot)$ zero mean in Eq. (3) (left), we disallow the nonlinear mapping $\mathbf{Af}\left(\mathbf{B}\cdot\right)$ to affect the expected output $\mathbf{y}_t$, that is, to compete with the bias term. Similarly, by making the nonlinear activations $\mathbf{f}(\cdot)$ zero slope in Eq. (3) (right), we disallow the nonlinear mapping $\mathbf{Af}\left(\mathbf{B}\cdot\right)$ from affecting the expected dependency on the input, that is, to compete with the linear short-cut mapping $\mathbf{C}$. In traditional neural networks, the linear dependencies (expected $\partial\mathbf{y}_t/\partial\mathbf{x}_t$) are modeled by many competing paths from an input to an output (e.g. via each hidden unit), whereas this architecture gathers the linear dependencies to be modeled only by $\mathbf{C}$.

In [10] it was shown experimentally that less competition between parts of the model will speed up learning. In [16], more careful connections to second-order optimization methods were drawn.

3.2 Two-Layer Contractive Encoding

A common regularizer for MLPs is the L_2 penalty on the weight matrices. This regularizer is well-motivated for linear methods (e.g. ridge regression or logistic regression), where it penalizes strong dependence of y on few variables in $\mathbf{x}$, and thus ensures invariance of y to small changes in $\mathbf{x}$. For MLPs, which contain saturating non-linearities, this desirable property can be achieved with strongly positive or negative weights as well. Rifai et al. [14] show that the generalization of the L_2-norm penalty to the case where non-linearities are involved in the computation of y is a penalty on the Frobenius norm of the Jacobian $\|J_y(\mathbf{x})\|_F$ ("contractive" regularization). [14, 13] demonstrate that pretraining simple auto-encoders with the contractive penalty produces features which identify the data manifold and can aid finetuning. However, Schulz et al. [15] demonstrate that auto-encoders with one hidden layer can fail to identify stable features in the input when their variables in $\mathbf{x}$ are XOR-related. [15] generalize the contractive

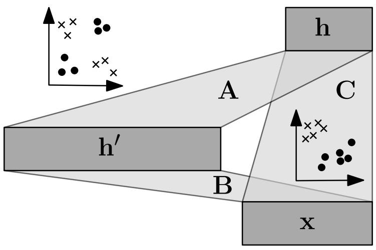

Fig. 1: Schematic visualization of our encoder. Features $\mathbf{h}$ of input $\mathbf{x}$ are determined both by a one-layer encoder via $\mathbf{C}$, and by a two-layer encoder via $\mathbf{B}$ and $\mathbf{A}$. Contractive regularization [14] and two-layer contractive regularization [15] are used to learn stable linear/non-linear representations in $\mathbf{h}$, respectively. Linear transformations in the two-layer part are moved to $\mathbf{C}$ using compensations [10] (not shown).

regularizer to the two-layer case,

$$\|J_{\mathbf{h}}(\mathbf{x})\|_F^2 = \sum_n^N \sum_m^M \left(1 - \mathbf{h}_n^2\right)^2 \left(\sum_k^K \mathbf{A}_{nk} \mathbf{B}_{km} (1 - \mathbf{h'}_k^2) \right)^2 , \tag{7}$$

where $\mathbf{x} \in \mathbb{R}^M$, $\mathbf{B} \in \mathbb{R}^{K \times M}$, $\mathbf{A} \in \mathbb{R}^{N \times K}$, and $\mathbf{h} = \tanh(\mathbf{A} \tanh(\mathbf{Bx})) = \tanh(\mathbf{Ah'})$.

In this paper, we argue that while two-layer encodings are harder to learn, we can combine their ability to detect highly non-linear features with the easy-to-learn one-layer encodings by introducing shortcuts. Shortcut weights $\mathbf{C}$ from the input to the second hidden layer can be regularized as in [14], while the two-layer encoder is regularized as in [15]. We employ linear transformations and compensations (Sec. 3.1) to ensure that simple features continue to be learned by the shortcut weights, while the two-layer part of the encoder can focus on the difficult features. For this purpose, we extend the two-layer contractive regularizer to account for the linear transformations in (1) and (2),

$$\|J_{\mathbf{h}}(\mathbf{x})\|_F^2 = \sum (1 - \mathbf{h}^2)^{2^T} \left(\mathbf{C} + \mathbf{A}(\mathbf{B}^\alpha + \mathbf{B}')\right)^2 , \tag{8}$$

where $\mathbf{B}^\alpha_{km} = \alpha_k \mathbf{B}_{km}$, $\mathbf{B}'_{km} = \mathbf{B}_{km}(1 - \tanh^2(\mathbf{B}_{k\cdot}\mathbf{x}))$. Fig. 1 visualizes the proposed encoder structure.

4 Experiments

In the experiment, we evaluate the proposed framework in a semi-supervised setting using a handwritten digit dataset (MNIST, [9]). We assume that only 1200 training samples have their labels available, while all the other training samples are unlabeled. The task is to use an MLP trained on the training samples to classify 10,000 test samples.

4.1 Model and Learning

Our base model is a multi-layer perceptron (MLP) with two hidden layers having $\tanh$ hidden neurons. The output of the MLP is

$$\mathbf{y} = \mathbf{W} \left(\tanh \left(\mathbf{A} \tanh \left(\mathbf{Bx} \right) \right) \right) , \tag{9}$$

Table 1: Classification accuracies depending on training strategy on MNIST using 1200 labeled examples, and the size of the second hidden layer fixed to 100. Standard deviations are over 10 trials with different draws of the training set.

Strategy	Test Error	Std. Dev.
S	13.27	1.47
U+S	8.835	0.33
C+U+S	8.989	0.25
T+U+S	9.132	0.19
2C	8.77	0.43
C+T+U+S	**8.695**	0.41

where $\mathbf{W}$, $\mathbf{A}$ and $\mathbf{B}$ are weight matrices. We have omitted biases for simplicity of notation. As baseline we trained this MLP both with and without pretraining. For the pretrained MLP we consider the bottom two layers as an autoencoder with two hidden layers and trained them using both labeled and unlabeled samples.

When the hidden neurons, or perceptrons, in the MLP were linearly transformed, we added shortcut connections from the input to the second hidden layer to maintain the equivalence after the transformation. In that case, the output of the MLP is

$$\mathbf{y} = \mathbf{W}\mathbf{h} = \mathbf{W}\tanh(\mathbf{A}\mathbf{h}' + \mathbf{C}\mathbf{x}), \tag{10}$$

where $\mathbf{h}' = \tanh(\mathbf{B}\mathbf{x}) + \mathbf{B}^{\alpha}\mathbf{x} + \boldsymbol{\beta}$, and $\mathbf{C}$ is the weight matrix of the shortcuts.[4].

As a comparison, we tried both using either one of the two-layer contractive encoding and the linear transformation and using both of them together. In this way, we can easily see the effectiveness of the proposed way of using both approaches together.

Specifically, we used six different training strategies:

1. **S**: MLP trained with labeled samples only
2. **U+S**: MLP pretrained with unlabeled samples and finetuned
3. **2C**: MLP pretrained with stacked contractive auto-encoders
4. **C+U+S**: MLP pretrained and finetuned with two-layer contractive encoding
5. **T+U+S**: MLP with shortcuts pretrained and finetuned with linear transformation
6. **C+T+U+S**: MLP with shortcuts pretrained and finetuned using both the two-layer contractive encoding and linear transformation

We estimated hyperparameters such as learning rates, weight decay constant, regularization strength and the size of the first hidden layer using hyperopt [4]. The number of training epochs is determined with early stopping. For pre-training, we minimized reconstruction error on a 10 000 sample validation set for every training strategy. We then determined 1200 labeled training samples randomly and employed five-fold cross validation and hyperopt to determine learning rates for finetuning on the cross-entropy loss function. To reduce overfitting induced by the small size of labeled samples, we fixed the number of hidden neurons in the second hidden layer to 100 (see, e.g., [3]). The weight matrices $\mathbf{A}$ and $\mathbf{B}$ were initialized randomly according to the normalized scale [6], while $\mathbf{C}$ was initialized with zeroes.

[4] When we pretrained the MLP as a two-layer contractive encoding, we tied the weights $\mathbf{A}$ and $\mathbf{B}$ between the encoder and decoder. However, we did not share $\mathbf{C}$, α_l's and β_l's.

Once the hyperparameters were found, we evaluated each strategy by training 10 MLPs with different sets of randomly sampled 1200 labeled training samples and classifying the held-out test samples.

4.2 Result and Analysis

In Table 1, the resulting classification accuracies for all six strategies are presented. As expected the best performing strategy was the one which pretrained the MLP as the two-layer contractive encoding using the linear transformation (**C+T+U+S**). This strategy was able to outperform the strategies **U+S**, **C+U+S** as well as **T+U+S**. On the other extreme, any approach which pretrains an MLP less greedily significantly outperforms the case where only labeled samples were used to finetune an MLP (**S**). Our proposed method also has a slight advantage over the stacked contractive auto-encoder (**2C**).

Interestingly, using either the two-layer contractive encoding or the linear transformation only turned out to be just as good as the naïve pretraining strategy (**U+S**). This suggests that it is not easy to train the two-layer contractive encoding well easily without a good training algorithm. Only when training became easier by linearly transforming perceptrons to have zero-mean and zero-slope on average, we were able to see the improvement (**C+T+U+S**), which confirms our claim.

5 Conclusions

In this paper we claimed that a less greedy approach of pretraining a multi-layer perceptron (MLP) can be improved by having both good regularization based on minimizing the Jacobian of hidden activations with respect to the input [15, 14] and powerful learning algorithm based on linearly transforming hidden neurons [10, 16]. We focused on validating this claim in a semi-supervised setting where only few labeled samples and vast amount of unlabeled samples are available

We empirically demonstrated the validity of our claim by considering a task of classifying handwritten digits using an MLP when only 1200 training samples out of 60,000 were assumed to have corresponding labels. It was clear from the experiment that a less greedy pretraining approach indeed helps significantly when there were only a handful of labeled samples. Furthermore, we were able to see that generalization performance could be significantly improved by pretraining an MLP with a two-layer contractive encoding using the linear transformation, confirming the validity of our claim.

The experiments in the paper are, however, limited in a couple of dimensions. Firstly, the structure of the MLP was limited to have only two hidden layers, and a small fixed-size second hidden layer, which makes it important for future to evaluate the proposed method with larger and deeper models. Secondly, it will be desirable to evaluate the proposed method with other datasets. These are left for future to be further investigated.

Acknowledgements This work was supported by the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN, 251170).

References

1. Bengio, Y.: Learning deep architectures for AI. Foundations and Trends in Machine Learning 2(1), 1–127 (2009)
2. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence Special Issue on Learning Deep Architectures (2013), early Access
3. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in Neural Information Processing Systems 19, pp. 153–160. MIT Press, Cambridge, MA (2007)
4. Bergstra, J., Bardenet, R., Bengio, Y., Kgl, B.: Algorithms for hyper-parameter optimization. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 24, pp. 2546–2554 (2011)
5. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-Supervised Learning. MIT Press, Cambridge, MA (2006)
6. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010). JMLR Workshop and Conference Proceedings, vol. 9, pp. 249–256. JMLR W&CP (May 2010)
7. Hinton, G., Salakhutdinov, R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (July 2006)
8. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2(5), 359–366 (Jan 1989)
9. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
10. Raiko, T., Valpola, H., LeCun, Y.: Deep learning made easier by linear transformations in perceptrons. In: Proceedings of the Fifteenth Internation Conference on Artificial Intelligence and Statistics (AISTATS 2012). JMLR Workshop and Conference Proceedings, vol. 22, pp. 924–932. JMLR W&CP (April 2012)
11. Ranzato, M., Huang, F.J., Boureau, Y.L., LeCun, Y.: Unsupervised learning of invariant feature hierarchies with applications to object recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2007). pp. 1–8 (June 2007)
12. Ranzato, M., Poultney, C., Chopra, S., LeCun, Y.: Efficient learning of sparse representations with an energy-based model. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in Neural Information Processing Systems 19, pp. 1137–1144. MIT Press, Cambridge, MA (2007)
13. Rifai, S., Mesnil, G., Vincent, P., Muller, X., Bengio, Y., Dauphin, Y., Glorot, X.: Higher order contractive auto-encoder. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science, vol. 6912, pp. 645–660. Springer Berlin Heidelberg (2011)
14. Rifai, S., Vincent, P., Muller, X., Glorot, X., Bengio, Y.: Contractive auto-encoders: Explicit invariance during feature extraction. In: Getoor, L., Scheffer, T. (eds.) Proceedings of the 28th International Conference on Machine Learning (ICML 2011). pp. 833–840. ACM, New York, NY, USA (June 2011)
15. Schulz, H., Behnke, S.: Learning two-layer contractive encodings. In: Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN 2012) (September 2012)
16. Vatanen, T., Raiko, T., Valpola, H., LeCun, Y.: Pushing stochastic gradient towards second-order methods – backpropagation learning with transformations in nonlinearities. arXiv:1301.3476 [cs.LG] (May 2013)