Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks

Dong-Hyun Lee SAYIT78@GMAIL.COM
Nangman Computing, 117D Garden five Tools, Munjeong-dong Songpa-gu, Seoul, Korea

Abstract

We propose the simple and efficient method of semi-supervised learning for deep neural networks. Basically, the proposed network is trained in a supervised fashion with labeled and unlabeled data simultaneously. For unlabeled data, *Pseudo-Labels*, just picking up the class which has the maximum predicted probability, are used as if they were true labels. This is in effect equivalent to *Entropy Regularization*. It favors a low-density separation between classes, a commonly assumed prior for semi-supervised learning. With Denoising Auto-Encoder and Dropout, this simple method outperforms conventional methods for semi-supervised learning with very small labeled data on the MNIST handwritten digit dataset.

1. Introduction

Recently, deep neural networks have achieved great success in hard AI tasks (Hinton et al., 2006; Bengio et al., 2012). All of the successful methods for training deep neural networks have something in common : they rely on an unsupervised learning algorithm (Erhan et al., 2010). Most work in two main phases. In a first phase, *unsupervised pre-training*, the weights of all layers are initialized by this layer-wise unsupervised training. In a second phase, *fine-tuning*, the weights are trained globally with labels using backpropagation algorithm in a supervised fashion. All of these methods also work in a semi-supervised fashion. We have only to use extra unlabeled data for unsupervised pre-training.

Several authors have recently proposed semi-supervised learning methods of training supervised and unsupervised tasks using same neural network simultaneously. In (Ranzato et al., 2008), the weights of each layer are trained by minimizing the combined loss function of an autoencoder and a classifier. In (Larochelle et al., 2008), *Discriminative Restricted Boltzmann Machines* model the joint distribution of an input vector and the target class. In (Weston et al., 2008), the weights of all layers are trained by minimizing the combined loss function of a global supervised task and a *Semi-Supervised Embedding* as a regularizer.

In this article we propose the simpler way of training neural network in a semi-supervised fashion. Basically, the proposed network is trained in a supervised fashion with labeled and unlabeled data simultaneously. For unlabeled data, *Pseudo-Labels*, just picking up the class which has the maximum predicted probability every weights update, are used as if they were true labels. In principle, this method can combine almost all neural network models and training methods. In our experiments, Denoising Auto-Encoder (Vincent et al., 2008) and Dropout (Hinton et al., 2012) boost up the performance.

This method is in effect equivalent to *Entropy Regularization* (Grandvalet et al., 2006). The conditional entropy of the class probabilities can be used for a measure of class overlap. By minimizing the entropy for unlabeled data, the overlap of class probability distribution can be reduced. It favors a low-density separation between classes, a commonly assumed prior for semi-supervised learning (Chapelle et al., 2005).

Several experiments on the well-known MNIST dataset prove that the proposed method shows the state-of-the-art performance. This method (without unsupervised pre-training) earned second prize in *ICML 2013 Workshop in Challenges in Representation Learning: The Black Box Learning Challenge*.

ICML 2013 Workshop : Challenges in Representation Learning (WREPL), Atlanta, Georgia, USA, 2013.

2. Pseudo-Label Method for Deep Neural Networks

2.1. Deep Neural Networks

Pseudo-Label is the method for training deep neural networks in a semi-supervised fashion. In this article we will consider *multi-layer neural networks* with M layers of hidden units :

$$h_i^k = s^k \left(\sum_{j=1}^{d^k} W_{ij}^k h_j^{k-1} + b_i^k \right), \quad k = 1, ..., M+1 \quad (1)$$

where s^k is a non-linear activation function of the kth layer such as sigmoid, $f_i = h_i^{M+1}$ are output units used for predicting target class and $x_j = h_j^0$ are input values.

Sigmoid Unit is one of the most popular unit for neural network. This activation value usually stands for binary probability.

$$s(x) = \frac{1}{1 + e^{-x}} \quad (2)$$

When sigmoid units are used for output instead of softmax, we assume that the probability of each label is independent from each other. Though the labels of data set in our experiments are mutually exclusive, we use sigmoid output unit in order to make the best use of saturation region of sigmoid.

Rectified Linear Unit is receiving a great deal of attention recently (Glorot et al., 2011). This unit uses rectifier activation function :

$$s(x) = \max(0, x) \quad (3)$$

This is biologically plausible more than sigmoid and hyperbolic tangent. Because rectifier network gives rise to real zeros of hidden activations and thus truly sparse representations, this unit can boost up the network performance.

The whole network can be trained by minimizing supervised loss function

$$\sum_{i=1}^{C} L(y_i, f_i(x)), \quad (4)$$

where C is the number of labels, y_i's is the 1-of-K code of the label, f_i is the network output for i'th label, x is input vector. If we use sigmoid output unit, we can choose *Cross Entropy* as a loss function:

$$L(y_i, f_i) = -y_i \log f_i - (1 - y_i) \log(1 - f_i) \quad (5)$$

2.2. Denoising Auto-Encoder

Denoising Auto-Encoder is unsupervised learning algorithm based on the idea of making the learned representations robust to partial corruption of the input pattern (Vincent et al., 2008). This approach can be used to train autoencoders, and these DAE can be stacked to initialize deep neural networks.

$$h_i = s \left(\sum_{j=1}^{d_v} W_{ij} \widetilde{x}_j + b_i \right) \quad (6)$$

$$\widehat{x}_j = s \left(\sum_{i=1}^{d_h} W_{ij} h_i + a_j \right) \quad (7)$$

where $\widetilde{x}_j$ is corrupted version of the jth input value, $\widehat{x}_j$ is the reconstruction of the jth input value. Autoencoder training consists in minimizing The reconstruction error between x_j and $\widehat{x}_j$. For binary input value, common choice of the reconstruction error is *Cross Entropy* :

$$L(x, \widehat{x}) = \sum_{j=1}^{d_v} -x_j \log \widehat{x}_j - (1 - x_j) \log(1 - \widehat{x}_j) \quad (8)$$

We use DAE in a *unsupervised pre-training* phase. Masking noise with a probability 0.5 is used for corruption. Unlike original DAE, hidden unit is also masked with a probability 0.5 in our experiments. An exponentially decaying learning rate and linearly increasing momentum is also used. This scheme is inspired by Dropout.

2.3. Dropout

Dropout is a technique that can be applied to supervised learning of deep neural networks (Hinton et al., 2012). On the network activations of each example, hidden unit is randomly omitted with a probability of 0.5. Sometimes 20% dropout of visible units is also helpful.

$$h_i^k = drop \left(s^k \left(\sum_{j=1}^{d^k} W_{ij}^k h_j^{k-1} + b_i^k \right) \right), \quad k = 1, ..., M \quad (9)$$

where $drop(x) = 0$ with a probability of 0.5, otherwise $drop(x) = x$. Overfitting can be reduced by this technique to prevent complex co-adaptations on hidden representations of training data. Because in each weights update we train a different sub-model by omitting a half of hidden units, this training procedure is similar to bagging (Breiman, 1996), where many different networks are trained on different subsets of the

data. But dropout is different from bagging in that all of the sub-models share same weights.

For successful SGD training with dropout, An exponentially decaying learning rate is used that starts at a high value. And momentum is used to speed up training.

$$\Delta W(t+1) = p(t)\Delta W(t) - (1-p(t))\,\epsilon(t) < \nabla_W L > \tag{10}$$

$$W(t+1) = W(t) + \Delta W(t) \tag{11}$$

where,

$$\epsilon(t+1) = k\,\epsilon(t) \tag{12}$$

$$p(t) = \begin{cases} \frac{t}{T} p_f + \left(1 - \frac{t}{T}\right) p_i & t < T \\ p_f & t \geq T \end{cases} \tag{13}$$

with $k = 0.998$, $p_i = 0.5$, $p_f = 0.99$, $T = 500$, t is the current epoch, $< \nabla_W L >$ is the gradient of loss function, $\epsilon(0)$ is the initial learning rate. We use these parameters from original dropout paper (Hinton et al., 2012), but don't use weight regularization.

2.4. Pseudo-Label

Pseudo-Label are target classes for unlabeled data as if they were true labels. We just pick up the class which has maximum predicted probability for each unlabeled sample.

$$y'_i = \begin{cases} 1 & \text{if } i = \text{argmax}_{i'}\, f_{i'}(x) \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

We use Pseudo-Label in a *fine-tuning* phase with Dropout. The pre-trained network is trained in a supervised fashion with labeled and unlabeled data *simultaneously*. For unlabeled data, *Pseudo-Label*s recalculated every weights update are used for the same loss function of supervised learning task.

Because the total number of labeled data and unlabeled data is quite different and the training balance between them is quite important for the network performance, the overall loss function is

$$L = \frac{1}{n}\sum_{m=1}^{n}\sum_{i=1}^{C} L(y_i^m, f_i^m) + \alpha(t)\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} L(y_i'^m, f_i'^m), \tag{15}$$

where n is the number of mini-batch in labeled data for SGD, n' for unlabeled data, f_i^m is the output units of m's sample in labeled data, y_i^m is the label of that, $f_i'^m$ for unlabeled data, $y_i'^m$ is the pseudo-label of that for unlabeled data, $\alpha(t)$ is a coefficient balancing them.

The proper scheduling of $\alpha(t)$ is very important for the network performance. If $\alpha(t)$ is too high, it disturbs training even for labeled data. Whereas if $\alpha(t)$ is too small, we cannot use benefit from unlabeled data. Furthermore, the deterministic annealing process, by which $\alpha(t)$ is slowly increased, is expected to help the optimization process to avoid poor local minima (Grandvalet et al., 2006) so that the pseudo-labels of unlabeled data are similar to true labels as much as possible.

$$\alpha(t) = \begin{cases} 0 & t < T_1 \\ \frac{t-T_1}{T_2-T_1}\alpha_f & T_1 \leq t < T_2 \\ \alpha_f & T_2 \leq t \end{cases} \tag{16}$$

with $\alpha_f = 3$, $T_1 = 100$, $T_2 = 600$ without pre-training, $T_1 = 200$, $T_2 = 800$ with DAE.

3. Why could Pseudo-Label work?

3.1. Low-Density Separation between Classes

The goal of semi-supervised learning is to improve generalization performance using unlabeled data. The *cluster assumption* states that the decision boundary should lie in low-density regions to improve generalization performance (Chapelle et al., 2005).

Recently proposed methods of training neural networks using manifold learning such as *Semi-Supervised Embedding* and *Manifold Tangent Classifier* utilize this assumption. *Semi-Supervised Embedding* (Weston et al., 2008) uses embedding-based regularizer to improve the generalization performance of deep neural networks. Because neighbors of a data sample have similar activations with the sample by embedding-based penalty term, it's more likely that data samples in a high-density region have the same label. *Manifold Tangent Classifier* (Rifai et al., 2011b) encourages the network output to be insensitive to variations in the directions of low-dimensional manifold. So the same purpose is achieved.

3.2. Entropy Regularization

Entropy Regularization (Grandvalet et al., 2006) is a means to benefit from unlabeled data in the framework of maximum a posteriori estimation. This scheme favors low density separation between classes without any modeling of the density by minimizing the conditional entropy of class probabilities for unlabeled data.

$$H(y|x') = -\frac{1}{n'}\sum_{m=1}^{n'}\sum_{i=1}^{C} P(y_i^m = 1|x'^m)\log P(y_i^m = 1|x'^m) \tag{17}$$

where n' is the number of unlabeled data, C is the number of classes, y_i^m is the unknown label of the mth unlabeled sample, x'^m is the input vector of mth unla-

Table 1. The Conditional Entropy (17) of the network output of labeled(train) data, unlabeled data and test data on MNIST. dropNN is the neural network trained with only labeled data (corresponding to *Figure 1*(a)) , +PL is the network trained additionally with unlabeled data and Pseudo-Label (corresponding to *Figure 1*(b)).

	TRAIN	UNLABELED	TEST
DROPNN	2.63×10^{-9}	0.0349	0.0317
+PL	6.10×10^{-9}	0.0067	0.0114

beled sample. The entropy is a measure of class overlap. As class overlap decreases, the density of data points get lower at the decision boundary.

The MAP estimate is defined as the maximizer of the posterior distribution :

$$C(\theta, \lambda) = \sum_{m=1}^{n} \log P(y^m | x^m; \theta) - \lambda H(y | x'; \theta) \quad (18)$$

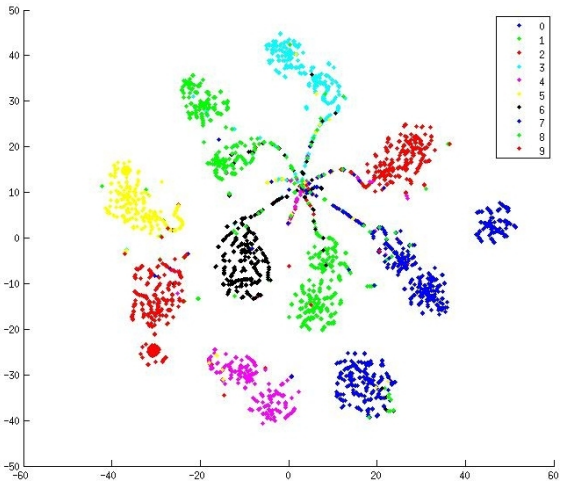
where n is the number of labeled data, x^m is the mth labeled sample, λ is a coefficient balancing two terms. By maximizing of the conditional log-likelihood of labeled data (the first term) with minimizing the entropy of unlabeled data (the second term), we can get the better generalization performance using unlabeled data.

3.3. Training with Pseudo-Label as Entropy Regularization

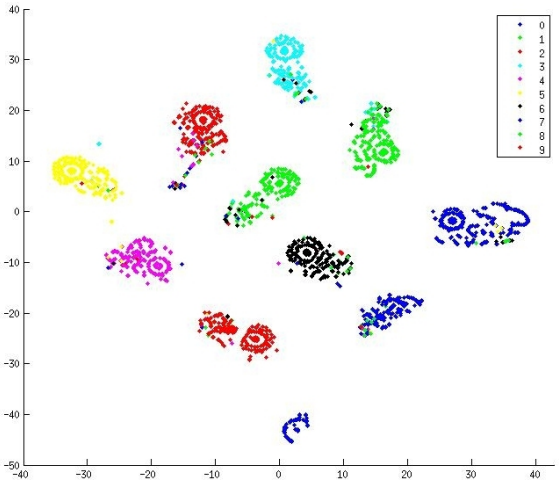
Our method encourages the predicted class probabilities to be near 1-of-K code via training with unlabeled data and Pseudo-Labels, so the entropy of (17) is minimized. Thus our method is equivalent to *Entropy Regularization*. The first term of (18) corresponds to the first term of (15), The second term of (18) corresponds to the second term of (15), α corresponds to λ.

Figure 1 shows t-SNE (Van der Maaten et al., 2008) 2-D embedding results of the network output of MNIST test data (not included in unlabeled data). The neural network was trained with 600 labeled data and with or without 60000 unlabeled data and Pseudo-Labels. Though the train error is zero in the two cases, the network outputs of test data is more condensed near 1-of-K code by training with unlabeled data and Pseudo-Labels, in other words, the entropy of (17) is minimized.

Table 2 shows the estimated entropy of (17). Though the entropy of labeled data is near zero in the two cases, the entropy of unlabeled data get lower by Pseudo-Label training, in addition, the entropy of test data get lower along with that. This makes the classification problem easier even for test data and makes the density of data points lower at the decision boundary. According to cluster assumption, we can get the better generalization performance.



(a) without unlabeled data (dropNN)



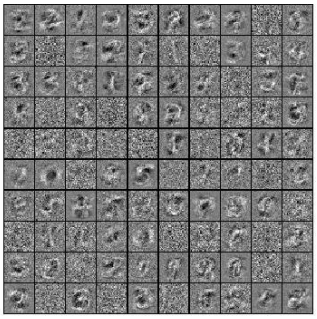
(b) with unlabeled data and Pseudo-Label (+PL)

Figure 1. t-SNE 2-D embedding of the network output of MNIST test data.

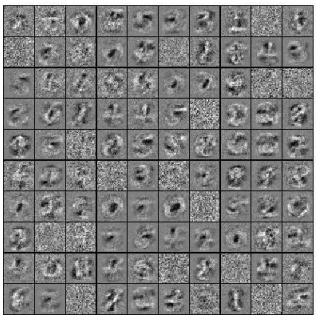
4. Experiments

4.1. Handwritten Digit Recognition (MNIST)

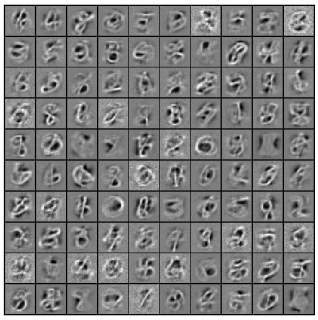
MNIST is one of the most famous dataset in deep learning literature. For comparison, we used the same semi-supervised setting with (Weston et al., 2008; Rifai et al., 2011b). We reduced the size of the labeled training set to 100, 600, 1000 and 3000. The training set has the same number of samples on each label. For validation set, we picked up 1000 labeled exam-



(a) without unlabeled data (dropNN)



(b) with unlabeled data and Pseudo-Label (+PL)



(c) using unsupervised pre-training with DAE (+PL+DAE)

Figure 2. Some filters on layer 1 of the network trained with or without unlabeled data on MNIST. 600 labeled data is used for supervised training.

ples separately. We used validation set for determining some hyper-parameters. The remaining data were used for unlabeled data. Because we could not get the same split of data set, 10 experiments on random split were done using the identical network and parameters. In the case of 100 labeled data, the results heavily depended on data split so that 30 experiments were done. 95% confidence interval is about $\pm 1 \sim 1.5\%$ for 100 labeled data, about $\pm 0.1 \sim 0.15\%$ for 600 labeled data, less than $\pm 0.1\%$ for 1000 and 3000 labeled data.

We used the neural network with 1 hidden layer. Rectified Linear Unit is used for hidden unit, Sigmoid Unit is used for output unit. The number of hidden units is 5000. For optimization, We used mini-batch Stochastic Gradient Descent with Dropout.[1] The initial learning rate is 1.5 and the number of mini-batch is 32 for labeled data, 256 for unlabeled data. These parameters were determined using validation set.

Table 2. Classification error on the MNIST test set with 600, 1000 and 3000 labeled training samples. We compare our method with results from (Weston et al., 2008; Rifai et al., 2011b). dropNN is our network model trained without unlabeled data, +PL with unlabeled data and Pseudo-Label, +PL+DAE using unsupervised pre-training with DAE in addition.

METHOD	100	600	1000	3000
NN	25.81	11.44	10.7	6.04
SVM	23.44	8.85	7.77	4.21
CNN	22.98	7.68	6.45	3.35
TSVM	16.81	6.16	5.38	3.45
DBN-RNCA	-	8.7	-	3.3
EMBEDNN	16.86	5.97	5.73	3.59
CAE	13.47	6.3	4.77	3.22
MTC	12.03	5.13	3.64	**2.57**
DROPNN	21.89	8.57	6.59	3.72
+PL	16.15	5.03	4.30	2.80
+PL+DAE	**10.49**	**4.01**	**3.46**	2.69

Table 2 compares our method with results from (Weston et al., 2008; Rifai et al., 2011b). Our method outperforms the conventional methods for small labeled data in spite of simplicity. The training scheme is less complex than *Manifold Tangent Classifier* and doesn't use computationally expensive similarity matrix between samples used in *Semi-Supervised Embedding*.

5. Conclusion

In this work, we have shown the simple and efficient way of semi-supervised learning for neural networks. Without complex training scheme and computationally expensive similarity matrix, the proposed method shows the state-of-the-art performance.

References

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *arXiv preprint arXiv:1206.5538*, 2012.

Breiman, L. Bagging predictors. *Machine learning*. 1996, 24.2: 123-140.

Chapelle, O., and Zien, A. Semi-supervised classica-

[1] We didn't use any weight regularization because the performance was the best without it.

tion by low density separation. *AISTATS*, 2005, (pp. 5764).

Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., and Bengio, S. Why does unsupervised pre-training help deep learning?. *The Journal of Machine Learning Research*, 2010, 11: 625-660.

Glorot, X., Bordes, A., and Bengio, Y. Deep Sparse Rectifier Networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume*. 2011. p. 315-323.

Yves Grandvalet and Yoshua Bengio, Entropy Regularization, In: *Semi-Supervised Learning*, pages 151–168, MIT Press, 2006.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, Vol. 313. no. 5786, pp. 504 - 507, 28 July 2006.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Larochelle, H., and Bengio, Y. Classification using discriminative restricted Boltzmann machines. In *Proceedings of the 25th international conference on Machine learning*. ACM, 2008. p. 536-543.

Ranzato, M., and Szummer, M. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th international conference on Machine learning*. ACM, 2008. p. 792-799.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011. p. 833-840.

Rifai, S., Dauphin, Y., Vincent, P., Bengio, Y., and Muller, X. The manifold tangent classifier. *Advances in Neural Information Processing Systems*, 2011, 24: 2294-2302.

Van der Maaten, L., and Hinton, G. Visualizing data using t-SNE. *The Journal of Machine Learning Research*, 2008, 9(2579-2605), 85.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P. A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*. ACM, 2008. pp. 1096-1103.

Weston, J., Ratle, F., and Collobert, R. Deep learning via semi-supervised embedding. In *Proceedings of the 25th international conference on Machine learning*. ACM, 2008. p. 1168-1175.