Regularization for Unsupervised Deep Neural Nets

Baiyang Wang, Diego Klabjan
Department of Industrial Engineering and Management Sciences,
Northwestern University, 2145 Sheridan Road, C210
Evanston, Illinois 60208

Abstract

Unsupervised neural networks, such as restricted Boltzmann machines (RBMs) and deep belief networks (DBNs), are powerful tools for feature selection and pattern recognition tasks. We demonstrate that overfitting occurs in such models just as in deep feedforward neural networks, and discuss possible regularization methods to reduce overfitting. We also propose a "partial" approach to improve the efficiency of Dropout/DropConnect in this scenario, and discuss the theoretical justification of these methods from model convergence and likelihood bounds. Finally, we compare the performance of these methods based on their likelihood and classification error rates for various pattern recognition data sets.

1 Introduction

Unsupervised neural networks assume unlabeled data to be generated from a neural network structure, and have been applied extensively to pattern analysis and recognition. The most basic one is the restricted Boltzmann machine (RBM) (Salakhutdinov, Mnih, and Hinton 2007), an energy-based model with a layer of hidden nodes and a layer of visible nodes. With such a basic structure, we can stack multiple layers of RBMs to create an unsupervised deep neural network structure, such as the deep belief network (DBN) and the deep Boltzmann machine (DBM) (Hinton, Osindero, and Teh 2006; Salakhutdinov and Hinton 2009a). These models can be calibrated with a combination of the stochastic gradient descent and the contrastive divergence (CD) algorithm or the PCD algorithm (Salakhutdinov, Mnih, and Hinton 2007; Tieleman 2008). Once we learn the parameters of a model, we can retrieve the values of the hidden nodes from the visible nodes, thus applying unsupervised neural networks for feature selection. Alternatively, we may consider applying the parameters obtained from an unsupervised deep neural network to initialize a deep feedforward neural network (FFNN), thus improving supervised learning.

One essential question for such models is to adjust for the high-dimensionality of their parameters and avoid overfitting. In FFNNs, the simplest regularization is arguably the early stopping method, which stops the gradient descent algorithm before the validation error rate goes up. The weight decay method, or L^s regularization, is also commonly used (Witten, Frank, and Hall 2011). Recently Dropout is proposed, which optimizes the parameters over an average of exponentially many models with a subset of all nodes (Srivastava et al. 2014). It has been shown to outperform weight decay regularization in many situations.

For regularizing unsupervised neural networks, sparse-RBM-type models encourage a smaller proportion of 1-valued hidden nodes (Cho, Ilin, and Taiko 2012; Lee, Ekanadham, and Ng 2007). DBNs are regularized in Goh et al. (2013) with outcome labels. While these works tend to be goal-specific, we consider regularization for unsupervised neural networks in a more general setting. Our work and contributions are as follows: (1) we extend common regularization methods to unsupervised deep neural networks, and explain their underlying mechanisms; (2) we propose partial Dropout/DropConnect which can improve the performance of Dropout/DropConnect; (3) we compare the performance of different regularization methods on real data sets, thus providing suggestions on regularizing unsupervised neural networks. We note that this is the very first study illustrating the mechanisms of various regularization methods for unsupervised neural nets with model convergence and likelihood bounds, including the effective newly proposed partial Dropout/DropConnect.

Section 2 reviews recent works for regularizing neural networks, and Section 3 exhibits RBM regularization as a basis for regularizing deeper networks. Section 4 discusses the model convergence of each regularization method. Section 5 extends regularization to unsupervised deep neural nets. Section 6 presents a numerical comparison of different regularization methods on RBM, DBN, DBM, RSM (Salakhutdinov and Hinton 2009b) and Gaussian RBM (Salakhutdinov, Mnih, and Hinton 2007). Section 7 discusses potential future research and concludes the paper.

2 Related Works

To begin with, we consider a simple FFNN with a single layer of input $\imath = (\imath_1, \ldots, \imath_I)^T$ and a single layer of output $o = (o_1, \ldots, o_J)^T \in \{0, 1\}^J$. The weight matrix W is of size $J \times I$. We assume the relation

$$E(o) = a(W \cdot \imath), \tag{1}$$

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

where $a(\cdot)$ is the activation function, such as the sigmoid function $\sigma(x) = 1/(1+e^{-x})$ applied element-wise. Equation (1) has the modified form in Srivastava et al. (2014),

$$\begin{cases} E(o|m) = a(m \star (W \cdot \imath)), \\ m = (m_1, \dots, m_J)^T \overset{iid}{\sim} Ber(p), \end{cases} \tag{2}$$

where $\star$ denotes element-wise multiplication, and $Ber(\cdot)$ denotes the Bernoulli distribution, thereby achieving the Dropout (DO) regularization for neural networks. In Dropout, we minimize the objective function

$$-l_{DO}(W) = -\sum_{n=1}^{N} E_m[\log p(o^{(n)}|\imath^{(n)}, W, m)], \tag{3}$$

which can be achieved by a stochastic gradient descent algorithm, sampling a different mask m per data example $(o^{(n)}, \imath^{(n)})$ and per iteration. We observe that this can be readily extended to deep FFNNs. Dropout regularizes neural networks because it incorporates prediction based on any subset of all the nodes, therefore penalizing the likelihood. A theoretical explanation is provided in Wager, Wang, and Liang (2013) for Dropout, noting that it can be viewed as feature noising for GLMs, and we have the relation

$$-l_{DO}(W) \doteq -\sum_{n=1}^{N} \log p(o^{(n)}|\imath^{(n)}, W) + R^q(W). \tag{4}$$

Here $J = 1$ for simplicity, and $R^q(W) = \frac{1}{2}\frac{p}{1-p}\sum_{n=1}^{N} \sum_{i=1}^{I} A''(W\imath^{(n)})(\imath_i^{(n)})^2 W_i^2$, where $A(\cdot)$ is the log-partition function of a GLM. Therefore, Dropout can be viewed approximately as the adaptive L^2 regularization (Baldi and Sadowski 2013; Wager, Wang, and Liang 2013). A recursive approximation of Dropout is provided in Baldi and Sadowski (2013) using normalized weighted geometric means to study its averaging properties.

An intuitive extension of Dropout is DropConnect (DC) (Wan et al. 2013), which has the form below

$$\begin{cases} E(o|m) = a((m \star W) \cdot \imath), \\ m = (m_{ij})_{J \times I} \overset{iid}{\sim} Ber(p), \end{cases} \tag{5}$$

and thus masks the weights rather than the nodes. The objective $l_{DC}(W)$ has the same form as in (3). There are a number of related model averaging regularization methods, each of which averages over subsets of the original model. For instance, Standout varies Dropout probabilities for different nodes which constitute a binary belief network (Ba and Frey 2013). Shakeout adds additional noise to Dropout so that it approximates elastic-net regularization (Kang, Li, and Tao 2016). Fast Dropout accelerates Dropout with Gaussian approximation (Wang and Manning 2013). Variational Dropout applies variational Bayes to infer the Dropout function (Kingma, Salimans, and Welling 2015).

We note that while Dropout has been discussed for RBMs (Srivastava et al. 2014), to the best of our knowledge, there is no literature extending common regularization methods to RBMs and unsupervised deep neural networks; for instance, adaptive L^s regularization and DropConnect as mentioned. Therefore, below we discuss their implementations and examine their empirical performance. In addition to studying model convergence and likelihood bounds, we propose partial Dropout/DropConnect which iteratively drops a subset of nodes or edges based on a given calibrated model, therefore improving robustness in many situations.

3 RBM Regularization

For a Restricted Boltzmann machine, we assume that $v = (v_1, \cdots, v_J)^T \in \{0,1\}^J$ denotes the visible vector, and $h = (h_1, \cdots, h_I)^T \in \{0,1\}^I$ denotes the hidden vector. Each v_j, $j = 1, \dots, J$ is a visible node and each h_i, $i = 1, \dots, I$ is a hidden node. The joint probability is

$$\begin{cases} P(v,h) = e^{-E(v,h)} / \sum_{\nu,\eta} e^{-E(\nu,\eta)}, \\ E(v,h) = -b^T v - c^T h - h^T W v. \end{cases} \tag{6}$$

We let the parameters $\vartheta = (b, c, W) \in \Theta$, which is a vector containing all components of b, c, and W. To calibrate the model is to find $\hat{\theta} = \arg\max_{\vartheta \in \Theta} \sum_{n=1}^{N} \log P(v^{(n)}|\vartheta)$.

An RBM is a neural network because we have the following conditional probabilities

$$\begin{cases} P(h_i = 1|v) = \sigma(c_i + W_{i\cdot} v), \\ P(v_j = 1|h) = \sigma(b_j + W_{\cdot j}^T h), \end{cases} \tag{7}$$

where $W_{i\cdot}$ and $W_{\cdot j}$ represent, respectively, the i-th row and j-th column of W. The gradient descent algorithm is applied to calibration. The gradient of the log-likelihood can be expressed in the following form

$$-\frac{\partial \log P(v^{(n)})}{\partial \vartheta} = \frac{\partial \mathcal{F}(v^{(n)})}{\partial \vartheta} - \sum_{v \in \{0,1\}^J} P(v) \frac{\partial \mathcal{F}(v)}{\partial \vartheta}, \tag{8}$$

where $\mathcal{F}(v) = -b^T v - \sum_{i=1}^{I} \log(1 + e^{c_i + W_{i\cdot} v})$ is the free energy. The right-hand side of (8) is approximated by contrastive divergence with k steps of Gibbs sampling (CD-k) (Salakhutdinov, Mnih, and Hinton 2007).

3.1 Weight Decay Regularization

Weight decay, or L^s regularization, adds the term $\lambda \|W\|_s^s$ to the negative log-likelihood of an RBM. The most commonly used is L^2 (ridge regression), or L^1 (LASSO). In all situations, we do not regularize biases for simplicity.

Here we consider a more general form. Suppose we have a trained set of weights W from CD with no regularization. Instead of adding the term $\lambda \|W\|_s^s$, we add the term $\frac{\mu}{IJ} \sum_{i,j} |W_{ij}|^s / |\hat{W}_{ij}|^s$ to the negative log-likelihood. Apparently this adjusts for the different scales of the components of W. We refer to this approach as adaptive L^s. We note that adaptive L^1 is the adaptive LASSO (Zou 2006), and adaptive L^2 plus L^1 is the elastic-net (Zou and Hastie 2005). We consider the performance of L^2 regularization plus adaptive L^1 regularization ($L^2 + AL^1$) below.

3.2 Model Averaging Regularization

As discussed in Srivastava et al. (2014), to characterize a Dropout (DO) RBM, we simply need to apply the following conditional distributions

$$\begin{cases} P_{DO}(h_i = 1|v,m) = m_i \cdot \sigma(c_i + W_{i\cdot}v), \\ P_{DO}(v_j = 1|h,m) = \sigma(b_j + W_{\cdot j}^T h). \end{cases} \tag{9}$$

Therefore, given a fixed mask $m \in \{0,1\}^I$, we actually obtain an RBM with all visible nodes v and hidden nodes $\{h_i : m_i = 1\}$. Hidden nodes $\{h_i : m_i = 0\}$ are fixed to zero so they have no influence on the conditional RBM. Apart from replacing (7) with (9), the only other change needed is to replace $\mathcal{F}(v)$ with $\mathcal{F}_{DO}(v|m) = -b^T v - \sum_{i=1}^{I} m_i \log(1 + e^{c_i + W_{i\cdot} v})$. In terms of training, we suggest sampling a different mask per data example $v^{(n)}$ and per iteration as in Srivastava et al. (2014).

A DropConnect (DC) RBM is closely related; given a mask $m = \{0,1\}^{IJ}$ on weights W, W in a plain RBM is replaced by $m * W$ everywhere. We suggest sampling a different mask m per mini-batch since it is usually much larger than a mask in a Dropout RBM.

3.3 Network Pruning Regularization

There are typically many nodes or weights which are of little importance in a neural network. In network pruning, such unimportant nodes or weights are discarded, and the neural network is retrained. This process can be conducted iteratively (Reed 1993). Now we consider two variants of network pruning for RBMs. For an trained set of weights $\hat{W}$ with no regularization, we consider implementing a fixed mask $m = (m_{ij})_{I\times J}$ where

$$m_{ij} = 1_{|\hat{W}_{ij}| \geq Q},\ Q = \mathcal{Q}_{100(1-p)\%}(|\hat{W}|), \tag{10}$$

i.e. Q is the $100(1-p)\%$-th left percentile of all $|\hat{W}_{ij}|$, and $p \in (0,1)$ is some fixed proportion of retained weights. We then recalibrate the weights and biases fixing mask m, leading to a simple network pruning (SNP) procedure which deletes $100(1-p)\%$ of all weights. We may also consider deleting $100(1-p)/r\%$ of all weights at a time, and conduct the above process r times, leading to an iterative network pruning (INP) procedure.

3.4 Hybrid Regularization

We may consider combining some of the above approaches. For instance, Srivastava et al. (2014) considered a combination of L^s and Dropout. We introduce two new hybrid approaches, namely partial DropConnect (PDC) presented in Algorithm 1 and partial Dropout (PDO), which generalizes DropConnect and Dropout, and borrows from network pruning. The rationale comes from some of the model convergence results exhibited later.

As before, suppose we have a trained set of weights $\hat{W}$ with no regularization. Instead of implementing a fixed mask m, we perform DropConnect regularization with different retaining probabilities p_{ij} for each weight W_{ij}. We let the quantile $Q = \mathcal{Q}_{100(1-q)\%}(|\hat{W}|)$, and

$$p_{ij} = 1_{|\hat{W}_{ij}| \geq Q} + p_0 \cdot 1_{|\hat{W}_{ij}| < Q}. \tag{11}$$

Therefore, we sample a different $m = (m_{ij})_{I\times J} \overset{ind}{\sim} Ber(p_{ij})$ per mini-batch, which means that we always keep $100q\%$ of all the weights, and randomly drop the remaining weights with probability $100(1-p_0)\%$. The mask m can be resampled iteratively. Intuitively, we are trying to maximize the following

$$\max_{\vartheta\in\Theta, p_{ij}\in\{p_0,1\}} E_m[\log P(data|\vartheta, m)]. \tag{12}$$

such that $m_{ij} \overset{ind}{\sim} Ber(p_{ij})$, and $\sum 1_{p_{ij}=1} = qIJ$.

```
Algorithm 1. (Partial DropConnect)
1. Initialize $\hat{\theta}_p = \hat{\theta}$, the unregularized trained parameters for an RBM.
2. Find retaining rates $p = (p_{ij})_{I\times J}$ from (11).
3. Retrain weights $\vartheta$ with DropConnect for a given number of iterations, and then update $\hat{\theta}_p$.
4. If maximum number of iterations reached, stop and obtain $\hat{\theta}_p$; otherwise, go back to Step 2.
```

This technique is proposed because we hypothesize that some weights could be more important than others *a posteriori*, so dropping them could cause much variation among the models being averaged. From (11), in partial Dropout, we tend to drop weights which have smaller magnitude, since setting larger weights to zero may substantially alter the structure of a neural network. Experiments on real data show that this technique can effectively improve the performance of plain DropConnect.

We denote $l_p(\vartheta) = E_{m\sim Ber(p)}[\log P(data|\vartheta, m)]$, and $l(\vartheta) = \log P(data|\vartheta)$. From first-order Taylor's expansion,

$$\begin{aligned} &|l_p(\hat{\theta}_p) - l(\hat{\theta})| \leq |l_p(\hat{\theta}_p) - l(\hat{\theta}_p)| + |l(\hat{\theta}_p) - l(\hat{\theta})| \\ &= \left| \sum_{i,j} \frac{\partial}{\partial W_{ij}} l(\tilde{\theta})(1-p_{ij})\hat{W}_{ij,p} \right| + |l(\hat{\theta}_p) - l(\hat{\theta})| \\ &\leq K \sum (1-p_{ij})|\hat{W}_{ij,p}| + |l(\hat{\theta}_p) - l(\hat{\theta})|. \end{aligned} \tag{13}$$

Here $\tilde{\theta}$ lies between $\hat{\theta}_p$ and $m * \hat{\theta}_p$ from Taylor's expansion, and $K = \sup_{\vartheta\in\Theta} \|\frac{\partial}{\partial\vartheta} l(\vartheta)\|_\infty$ is a Lipschitz constant.

Note that given p and $\hat{\theta}_p$, Step 2 in Algorithm 1 lowers the term $\sum(1-p_{ij})|\hat{W}_{ij,p}|$ by assigning $(1-p_0)$ to weights of smaller magnitude, reducing an upper bound of $|l_p(\hat{\theta}_p) - l(\hat{\theta})|$. Step 3 further increases $l_p(\hat{\theta}_p)$ and reduces the gap $|l_p(\hat{\theta}_p) - l(\hat{\theta})|$. Therefore, each iteration of Algorithm 1 tends to increase $l_p(\hat{\theta}_p)$, and hence Algorithm 1 provides an intuitive solution to problem (12).

We also consider a partial Dropout approach which is analogous to partial DropConnect and keeps some important nodes rather than weights. We set a mask for nodes $m = (m_1, \ldots, m_I)$, $m_i \overset{ind}{\sim} Ber(p_i)$, where

$$\begin{cases} p_i = 1_{\|\hat{W}_{i\cdot}\| \geq Q} + p_0 \cdot 1_{\|\hat{W}_{i\cdot}\| < Q}, \\ Q = \mathcal{Q}_{100(1-q)\%}(\|\hat{W}_{i\cdot}\|). \end{cases} \tag{14}$$

This algorithm protects more important hidden nodes from being dropped in order to reduce variation. We also evaluate its empirical performance later.

4 More Theoretical Considerations

Here we discuss the model convergence properties of different regularization methods when the number of data examples $N \to \infty$. We mark all regularization coefficients and parameter estimates with ${}^{(N)}$ when there are N data examples. We assume $\vartheta = (b, c, W) \in \Theta$, which is compact, $\dim(\Theta) = D$, $P(v|\vartheta)$ is unique for each $\vartheta \in \Theta$, and $v^{(1)}, \dots, v^{(N)}$ are i.i.d. generated from an RBM with a "true" set of parameters θ. We denote each regularized calibrated set of parameters as $\tilde{\theta}^{(N)}$.

Let $A = \{d : \theta_d \neq 0\}$ and $\theta_A = \{\theta_d : d \in A\}$. (Zou 2006) showed that AL^1 guarantees asymptotic normality and identification of set A for linear regression. We demonstrate that similar results hold for $L^2 + AL^1$ for RBMs. We let $\lambda^{(N)} = (\lambda_1^{(N)}, \dots, \lambda_D^{(N)})$ and $\mu^{(N)} = (\mu_1^{(N)}, \dots, \mu_D^{(N)})$ be the L^2 and L^1 regularization coefficients for each component. The proofs of all propositions and corollaries below are in the supplementary material (Wang and Klabjan 2016).

Proposition 1. (a) If $\lambda^{(N)}/N \to 0$, $\mu^{(N)}/N \to 0$ as $N \to \infty$, then the estimate $\tilde{\theta}^{(N)} \xrightarrow{P} \theta$; (b) if also, $\mu_d^{(N)}/\sqrt{N} \to 0 \cdot 1_{\theta_d \neq 0} + \infty \cdot 1_{\theta_d = 0}$, $\lambda^{(N)}/\sqrt{N} \to 0$, then $\sqrt{n}(\tilde{\theta}_A^{(N)} - \theta_A) \xrightarrow{d} N(0, I^{-1}(\theta_A))$, where I is the Fisher information matrix; $P(\hat{A}^{(N)} = A) \to 1$, where $\hat{A}^{(N)} = \{d : \tilde{\theta}_d^{(N)} \neq 0\}$. □

For Dropout and DropConnect RBMs, we also assume that the data is generated from a plain RBM structure. We assume $p^{(N)}$ is of size $I \times J$ as in (11) for DropConnect and of length I as in (14) for Dropout, therefore covering the cases of both original and partial Dropout/DropConnect with a fixed set of dropping rates. With a decreasing dropping rate $1 - p^{(N)} \to 0$ with $N \to \infty$, we obtain the following convergence result.

Proposition 2. If $p^{(N)} \to 1$ as $N \to \infty$, then $\tilde{\theta}^{(N)} \xrightarrow{P} \theta$. □

For network pruning, we show that as the number of data examples increase, if the retained proportion of parameters $p^{(N)} = p$ can cover all nonzero components of θ, we will not miss any important component.

Proposition 3. Assume $p > p_0 := |A|/D$. Then for simple network pruning, as $N \to \infty$, (a) $\tilde{\theta}^{(N)} \xrightarrow{P} \theta$; (b) for sufficiently large N, there exists $\rho > 0$ such that $P(A \in \hat{A}^{(N)}) \geq 1 - e^{-\rho N}$. □

Corollary 1. The above results also hold for iterative network pruning. □

We note that for all regularization methods, under the above conditions, the calibrated weights converge to the "true" set of parameters θ, which indicates consistency. Also, adding L^1 regularization guarantees that we can identify components of zero value with infinitely many examples. The major benefits of Dropout come from the facts that it makes L^2 regularization adaptive, and also encourages more confident prediction of the outcomes (Wager, Wang, and Liang 2013). We propose partial DropConnect also based on Proposition 3, i.e. we do not drop the more important components of θ, therefore possibly reducing variation caused by dropping influential weights. Partial Dropout follows from the same reasoning.

5 Extension to Other Networks

5.1 Deep Belief Networks

We consider the multilayer network below,

$$P(v, h^1, \dots, h^L) \; P(v|h^1) \prod_{l=1}^{L-1} P(h^l|h^{l+1}) P(h^L), \quad (15)$$

where each probability on the right-hand side is from an RBM. To train the weights of RBM(v, h_1), ..., RBM(h^{L-1}, h^L), we only need to carry out a greedy layer-wise training approach, i.e. we first train the weights of RBM(v, h_1), and then use $E(h^1|v)$ to train RBM(h_1, h_2), etc. The weights of the RBMs are used to initialize a deep FFNN which is finetuned with gradient descent. RBM regularization is applicable to each layer of a DBN.

Here we show that adding layers to a Dropout/DropConnect DBN improves the likelihood given symmetry of the weights of two adjacent layers. Similar results for plain DBN are in Hinton, Osindero, and Teh (2006) and Bengio (2007). We demonstrate this by using likelihood bounds.

We let DBN_L denote an L-layer DBN and DBN_{L+1} denote an $(L+1)$-layer DBN with the first L layers being the same as in DBN_L. For a data example of a visible vector v, the log-likelihood is bounded as follows,

$$\begin{aligned} & E_m[\log P_{DBN_{L+1}}(v|m, m^*)] \\ \geq & E_m[H_{P_{DBN_L}(h^L|v,m)}] + \sum_{h^L} E_{m,m^*}\{P_{DBN_L}(h^L|v, m) \\ & \cdot [\log P_{RBM_{L+1}}(h^L|m^*) + \log P_{DBN_L}(v|h^L, m)]\}. \end{aligned} \quad (16)$$

Here, H is the entropy function, and the derivation is analogous to Section 11 in Bengio (2007). Mask m is for DBN_L, and mask m^* is for the new $(L+1)$-th layer. Note that after we have trained the first L layers, and initialized the $(L+1)$-th layer symmetric to the L-th layer, assuming a constant dropping probability, we have

$$E_{m^*}[\log P_{RBM_{L+1}}(h^L|m^*)] = E_m[\log P_{DBN_L}(h^L|m)], \quad (17)$$

so DBN_{L+1} has the same log-likelihood bound as DBN_L. Training RBM_{L+1}, $E_{m^*}[\log P_{RBM_{L+1}}(h^L|m^*)]$ is guaranteed to increase, and therefore the likelihood of DBN_{L+1} is expected to improve. As a result, for regularized unsupervised deep neural nets, adding layers also tend to elevate the explanatory power of the network. Adding nodes has the same effect, providing a rationale for deep and large-scale networks. We present the following proposition.

Proposition 4. Adding nodes or layers (preserving weight symmetry) to a Dropout/DropConnect DBN continually improves the likelihood; also, adding layers of size $J \leq H^1 \leq H^2 \leq \cdots$ continually improves the likelihood. □

5.2 Other RBM Variants

More descriptions of DBMs, RSMs, and Gaussian RBMs are in the supplementary material (Wang and Klabjan 2016). RBM regularization can be extended to all these situations.

6 Data Studies

In this section, we compare the empirical performance of the aforementioned regularization methods on the following data sets: MNIST, NORB (image recognition); 20 Newsgroups, Reuters21578 (text classification); ISOLET (speech recognition). All results are obtained using GeForce GTX TITAN X in Theano.

6.1 Experiment Settings

We consider the following unsupervised neural network structures: DBN/DBM for MNIST; DBN for NORB; RSM plus logistic regression for 20 Newsgroups and Reuters21578; GRBM for ISOLET. CD-1 is performed for the rest of the paper. The following regularization methods are considered: None (no regularization); DO; DC; L^2; $L^2 + AL^1$; SNP; INP($r = 3$); PDO; PDC. The number of pretraining epochs is 100 per layer and the number of finetuning epochs is 300, with a finetuning learning rate of 0.1. For $L^2 + AL^1$, SNP, and INP which need re-calibration, we cut the 100 epochs into two halves (4 quarters for INP). For regularization parameters, we apply the following ranges: $p = 0.8 \sim 0.9$ for DO/DC/SNP/INP; $\lambda = 10^{-5} \sim 10^{-4}$ for L^2, similar to Hinton (2010); $\mu = 0.01 \sim 0.1$ for $L^2 + AL^1$; $p_0 = 0.5$, $q = 0.7 \sim 0.9$ or the reverse for PDO/PDC. We only make one update to the "partial" dropping rates to maintain simplicity. From the results, we note that unsupervised neural networks tend to need less regularization than FFNNs. We choose the best iteration and regularization parameters over a fixed set of parameter values according to the validation error rates.

6.2 The MNIST Data Set

The MNIST data set consists of 28^2 pixels of handwritten 0-9 digits. There are 50,000 training examples, 10,000 validation and 10,000 testing examples. We first consider the likelihood of the testing data of an RBM with 500 nodes for MNIST. There are two model fitting evaluation criteria: pseudo-likelihood and AIS-likelihood (Salakhutdinov and Murray 2008). The former is a sum of conditional likelihoods, while the latter directly estimates $P(v)$ with AIS.

In Figure 1 below using log-scale, $p = 0.9$ for DO, and $\lambda = 10^{-4}$ for L^2. These figures tend to be representative of the model fitting process. The pseudo-likelihood is a more optimistic estimate of the model fitting. We observe that Dropout outperforms the other two after about 50 epochs, and L^2 regularization does not improve the pseudo-likelihood. In terms of the AIS-likelihood, which is a much more conservative estimate of the model fitting, the fitting process seems to have three stages: (1) initial fitting; (2) "overfitting"; (3) re-fitting. We observe that L^2 improves the likelihood significantly, while Dropout catches up at about 300 epochs. Therefore, Dropout tends to improve model fitting according to both likelihood criteria.

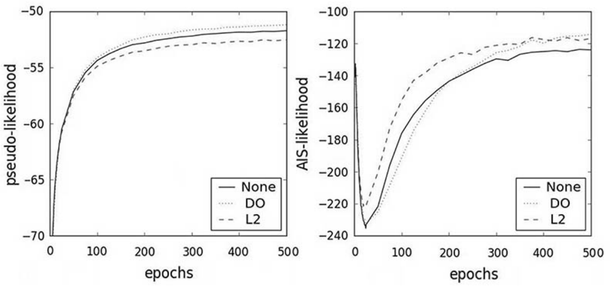

Figure 1: Left: Pseudo-likelihood of the RBM over 500 pretraining epochs. Right: AIS-likelihood of the RBM over 500 pretraining epochs.

In Figure 2, we can observe that more nodes increase the pseudo-likelihood, which is consistent with Proposition 4, but exhibit "overfitting" for the AIS-likelihood. However, such "overfitting" does not exist for pretraining purposes as well. Thus we suggest the pseudo-likelihood, and the AIS-likelihood should be viewed as too conservative.

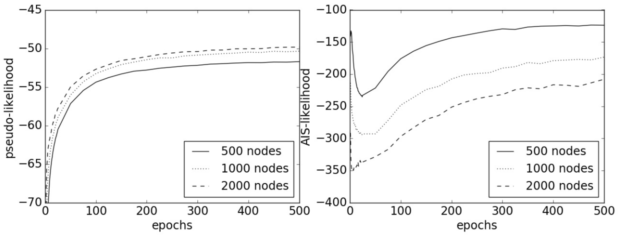

Figure 2: Left: Pseudo-likelihood of the Dropout RBM over 500 pretraining epochs. Right: AIS-likelihood of the Dropout RBM over 500 pretraining epochs.

Classification error rates tend to be a more practical measure. We first consider a 3-hidden-layer DBN with $1{,}000$ nodes per layer, pretraining learning rate 0.01, and batch size 10; see Table 1. We tried DBNs of 1, 2, and 4 hidden layers and found the aforementioned structure to perform best with None as baseline. The same was done for all other structures. We calculate the means of the classification errors for each regularization method averaged over 5 random replicates and their standard deviations. In each table, we stress in bold the top 3 performers with ties broken by deviation. We note that most of the regularization methods tend to improve the classification error rates, with DC and PDO yielding slightly higher error rates than no regularization.

	None	DO	DC	L^2	$L^2 + AL^1$
m.	1.35%	$\mathbf{1.30}\%$	1.37%	1.31%	1.35%
sd.	0.02%	0.04%	0.02%	0.03%	0.01%
	SNP	INP	PDO	PDC	
m.	$\mathbf{1.30}\%$	1.32%	1.36%	$\mathbf{1.30}\%$	
sd.	0.03%	0.03%	0.04%	0.03%	

Table 1: Classification errors for a 3-layer DBN for the MNIST data set.

In Table 2, we consider a 3-hidden-layer DBM with 1,000 nodes per layer. For simplicity, we only classify based on the original features. We let the pretraining learning rate be 0.03 and the batch size be 10.

	None	DO	DC	L^2	$L^2 + AL^1$
m.	1.22%	1.21%	1.20%	**1.14%**	**1.15%**
sd.	0.02%	0.02%	0.02%	0.02%	0.04%
	SNP	INP	PDO	PDC	
m.	1.18%	1.26%	1.21%	**1.12%**	
sd.	0.03%	0.02%	0.03%	0.02%	

Table 2: Classification errors for a 3-layer DBM for the MNIST data set.

It can be observed that regularization tends to yield more improvement for DBM than DBN, possibly because a DBM doubles both the visible layer and the third hidden layer, resulting in a "larger" neural network structure in general. Only INP proves to be unsuitable for the DBM; all other regularization methods work better, with PDC being the best.

6.3 The NORB Data Set

The NORB data set has 5 categories of images of 3D objects. There are 24,300 training examples, with 2,300 validation examples held out, and 24,300 testing examples. We follow preprocessing of Nair and Hinton (2009), and apply a sparse two-hidden-layer DBN with 4,000 nodes per layer as in Lee, Ekanadham, and Ng (2007) with a sparsity regularization coefficient of 10.0 and the first hidden layer being a Gaussian RBM. The pretraining learning rates are 0.001 and 0.01 for the first and second hidden layer, and the batch sizes for pretraining and finetuning are 100 and 20. Because the validation error often goes to zero, we choose the 300-th epoch and fix the regularization parameters as follows based on the best values of other data sets: $p = 0.9$ for DO/DC/SNP/INP, $\lambda = 10^{-4}$ for L^2, $\mu = 0.1$ for L^2+AL^1, $(p_0, q) = (0.5, 0.8)$ for PDO and $(p_0, q) = (0.8, 0.5)$ for PDC. In Table 3, only weight decay and PDO/PDC perform better than None, with PDC again being the best.

	None	DO	DC	L^2	$L^2 + AL^1$
m.	11.00%	11.15%	11.19%	**10.93%**	**10.91%**
sd.	0.15%	0.12%	0.10%	0.18%	0.17%
	SNP	INP	PDO	PDC	
m.	11.04%	11.14%	10.95%	**10.81%**	
sd.	0.18%	0.20%	0.15%	0.13%	

Table 3: Classification errors for the NORB data set.

6.4 The 20 Newsgroups Data Set

The 20 Newsgroups data set is a collection of news documents with 20 categories. There are 11,293 training examples, from which 6,293 validation examples are randomly held out, and 7,528 testing examples. We adopt the stemmed version, retain the most common 5,000 words, and train an RSM with 1,000 hidden nodes in a single layer. We consider this as a simple case of deep learning since it is a two-step procedure. The pretraining learning rate is 0.02 and the batch size is 50. We apply logistic regression to classify the trained features, i.e. hidden values of the RSM, as in Srivastava, Salakhutdinov, and Hinton (2013). This setting is quite challenging for unsupervised neural networks. In Table 4, Dropout performs best with other regularization methods yielding improvements except DropConnect.

	None	DO	DC	L^2	$L^2 + AL^1$
m.	30.8%	**28.8%**	35.2%	30.1%	30.1%
sd.	0.70%	0.23%	0.91%	0.30%	0.65%
	SNP	INP	PDO	PDC	
m.	**29.7%**	29.7%	30.1%	**29.7%**	
sd.	0.26%	0.48%	0.71%	0.34%	

Table 4: Classification errors for the trained features of RSM for the 20 Newsgroups data set.

6.5 The Reuters21578 Data Set

The Reuters21578 data set is a collection of newswire articles. We adopt the stemmed R-52 version which has 52 categories, 6,532 training examples, from which 1,032 validation examples are randomly held out, and 2,568 testing examples. We retain the most common 2,000 words, and train an RSM with 500 hidden nodes in a single layer. The pretraining learning rate is 0.1 and the batch size is 50. We make the learning rate large because the cost function is quite bumpy. From Table 5, we note that PDC works best, and PDO improves the performance of Dropout.

	None	DO	DC	L^2	$L^2 + AL^1$
m.	10.50%	11.91%	10.10%	10.06%	9.99%
sd.	0.64%	0.70%	0.32%	0.28%	0.41%
	SNP	INP	PDO	PDC	
m.	**9.99%**	10.10%	**9.98%**	**9.84%**	
sd.	0.27%	0.30%	0.24%	0.23%	

Table 5: Classification errors for the trained features from RSM and the Reuters21578 data set.

6.6 The ISOLET Data Set

The ISOLET data set consists of voice recordings of the Latin alphabet (a-z). There are 6,138 training examples, from which 638 validation examples are randomly held out, and 1,559 testing examples. We train a 1,000-hidden-node Gaussian RBM with pretraining learning rate 0.005, batch size 20, and initialize a FFNN, which can be viewed as a single-hidden-layer DBN. From Table 6, it is evident that all regularization methods work better then None, with PDC again being the best.

	None	DO	DC	L^2	$L^2 + AL^1$
m.	3.98%	3.87%	3.88%	**3.83%**	**3.86%**
sd.	0.09%	0.06%	0.11%	0.10%	0.07%
	SNP	INP	PDO	PDC	
m.	**3.86%**	3.86%	3.96%	**3.78%**	
sd.	0.07%	0.10%	0.08%	0.05%	

Table 6: Classification errors for the ISOLET data set.

6.7 Summary

From the above results, we observe that regularization does improve the structure of unsupervised deep neural networks and yields lower classification error rates for each data set studied herein. The most robust methods which yield improvements for all six instances are L^2, L^2+AL^1, and PDC. SNP is also acceptable, and preferable over INP. PDO can yield improvements for Dropout when Dropout is unsuitable for the network structure. PDC turns out to be the most stable method of all, and thus the recommended choice.

7 Conclusion

Regularization for deep learning has aroused much interest, and in this paper, we extend regularization to unsupervised deep learning, i.e. for DBNs and DBMs. We proposed several approaches, demonstrated their performance, and empirically compared the different techniques. For the future, we suggest that it would be of interest to consider more variants of model averaging regularization for supervised deep learning as well as novel methods of unsupervised learning; for instance, Kingma and Welling (2015) provided an interesting variational Bayesian auto-encoder approach.

References

Ba, L., and Frey, B. 2013. Adaptive dropout for training deep neural networks. In *Advances in Neural Information Processing Systems 26*. MIT Press.

Baldi, P., and Sadowski, P. 2013. Understanding dropout. In *Advances in Neural Information Processing Systems 26*. MIT Press.

Bengio, Y. 2007. Learning deep architectures for ai. https://www.iro.umontreal.ca/˜lisa/pointeurs/TR1312.pdf.

Cho, K.; Ilin, A.; and Taiko, T. 2012. Tikhonov-type regularization for restricted boltzmann machines. In *22nd International Conference on Artificial Neural Networks*.

Goh, H.; Thome, N.; Cord, M.; and Lim, J. 2013. Top-down regularization of deep belief networks. In *Advances in Neural Information Processing Systems 26*. MIT Press.

Hinton, G.; Osindero, S.; and Teh, Y. 2006. A fast learning algorithm for deep belief nets. *Neural Computation* 18:1527–1554.

Hinton, G. 2010. A practical guide to training restricted boltzmann machines. https://www.cs.toronto.edu/˜hinton/absps/guideTR.pdf.

Kang, G.; Li, J.; and Tao, D. 2016. Shakeout: A new regularized deep neural network training scheme. In *30th AAAI Conference on Artificial Intelligence*.

Kingma, D., and Welling, M. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*.

Kingma, D.; Salimans, T.; and Welling, M. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems 28*.

Lee, H.; Ekanadham, C.; and Ng, A. 2007. Sparse deep belief net model for visual area v2. In *Advances in Neural Information Processing Systems 20*. MIT Press.

Nair, V., and Hinton, G. 2009. 3d object recognition with deep belief nets. In *Advances in Neural Information Processing Systems 22*. MIT Press.

Reed, R. 1993. Pruning algorithms: a survey. *IEEE Transactions on Neural Networks* 4:740–747.

Salakhutdinov, R., and Hinton, G. 2009a. Deep boltzmann machines. In *12th International Conference on Artificial Intelligence and Statistics*.

Salakhutdinov, R., and Hinton, G. 2009b. Replicated softmax: an undirected topic model. In *Advances in Neural Information Processing Systems 22*. MIT Press.

Salakhutdinov, R., and Murray, I. 2008. On the quantitative analysis of deep belief networks. In *25th International Conference on Machine Learning*.

Salakhutdinov, R.; Mnih, A.; and Hinton, G. 2007. Restricted boltzmann machines for collaborative filtering. In *24th International Conference on Machine Learning*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 14:1929–1958.

Srivastava, N.; Salakhutdinov, R.; and Hinton, G. 2013. Modeling documents with a deep boltzmann machine. In *29th Conference on Uncertainty in Artificial Intelligence*.

Tieleman, T. 2008. Training restricted boltzmann machines using approximations to the likelihood gradient. In *25th International Conference on Machine Learning*.

Wager, S.; Wang, S.; and Liang, P. 2013. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems 26*. MIT Press.

Wan, L.; Zeiler, M.; Zhang, S.; LeCun, Y.; and Fergus, R. 2013. Regularization of neural networks using dropconnect. In *30th International Conference on Machine Learning*.

Wang, B., and Klabjan, D. 2016. Supplementary material for "regularization for unsupervised deep neural nets". http://www.dynresmanagement.com/publications.html.

Wang, S., and Manning, C. 2013. Fast dropout training. In *30th International Conference on Machine Learning*.

Witten, I.; Frank, E.; and Hall, M. 2011. *Data Mining: Practical Machine Learning Tools and Techniques*. Burlington, Massachusetts, USA: Morgan Kaufmann Publishers.

Zou, H., and Hastie, T. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67:301320.

Zou, H. 2006. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association* 101:1418–1429.