Noise Benefits in Backpropagation and Deep Bidirectional Pre-training

Kartik Audhkhasi, Osonde Osoba, Bart Kosko

Abstract—We prove that noise can speed convergence in the backpropagation algorithm. The proof consists of two separate results. The first result proves that the backpropagation algorithm is a special case of the generalized Expectation-Maximization (EM) algorithm for iterative maximum likelihood estimation. The second result uses the recent EM noise benefit to derive a sufficient condition for backpropagation training. The noise adds directly to the training data. A noise benefit also applies to the deep bidirectional pre-training of the neural network as well as to the backpropagation training of the network. The geometry of the noise benefit depends on the probability structure of the neurons at each layer. Logistic sigmoidal neurons produce a forbidden noise region that lies below a hyperplane. Then all noise on or above the hyperplane can only speed convergence of the neural network. The forbidden noise region is a sphere if the neurons have a Gaussian signal or activation function. These noise benefits all follow from the general noise benefit of the EM algorithm. Monte Carlo sample means estimate the population expectations in the EM algorithm. We demonstrate the noise benefits using MNIST digit classification.

Index Terms—Backpropagation, bidirectional associative memory, noise benefit, neural network, stochastic resonance, Expectation-Maximization algorithm

I. NOISE BENEFITS IN BACKPROPAGATION

We prove for the first time that noise can speed convergence in the popular backpropagation gradient-descent algorithm for training feedforward multilayer-perceptron neural networks [1], [2]. The proof casts backpropagation in terms of maximum likelihood estimation [3] and then shows that the iterative backpropagation algorithm is a special case of the general Expectation-Maximization (EM) algorithm. We then invoke the new noisy EM (NEM) theorem [4]–[6] that gives a sufficient condition for speeding convergence in the EM algorithm. Then the NEM result speeds convergence in the backpropagation algorithm. Figure 1 shows the noise benefit for squared-error training. Figure 4 shows a similar figure for cross-entropy training. The NEM version displays a 5.3% median decrease in squared error and a 4.2% median decrease in cross entropy per iteration compared with noiseless backpropagation training. We also observed that noise improves the recognition accuracy in both cases. There was little difference between the backpropagation and EM convergence rates.

We further show that a related NEM result holds for the pre-training of the individual layers of neurons in the multilayer perceptron. These so-called restricted Boltzmann machine (RBM) [7]–[9] layers are in fact simple bidirectional associative memories (BAMs) [1], [10], [11] that undergo synchronous updating of the neurons. They are BAMs because the neurons in contiguous layers use a connection matrix M in the forward pass and the corresponding matrix transpose $M^T$ in the backward pass and because the neurons have no within-layer connections. The general BAM convergence theorem [1], [10], [11] guarantees that all such rectangular matrices M are bidirectionally stable for synchronous or asynchronous neuron updates and for quite general neuronal activation nonlinearities because the RBM energy function is a Lyapunov function for the BAM network. This gives almost immediate convergence to a BAM fixed point after only a small number of synchronous back-and-forth updates when both layers use logistic neurons.

The NEM Theorem gives a type of “forbidden” condition [4]–[6], [12], [13] that ensures a noise speed up so long as the noise lies outside of a specified region in the noise state space. Figures 2 and 3 show that the noise must lie outside such regions to speed convergence. The neuron probability distribution function (pdf) and network connection or synaptic weights control the geometry of this forbidden region. Logistic neurons give the forbidden region as a half-space while Gaussian neurons give it as a sphere.

Theorems 3 and 4 give the sufficient conditions for a noise benefit in the popular cases of neural networks with logistic and Gaussian output neurons. Theorems 5 and 6 give similar sufficient conditions for Bernoulli-Bernoulli and Gaussian-Bernoulli BAMs. This is a type of “stochastic resonance” effect where a small amount of noise improves the performance of a nonlinear system while too much noise harms the system [12]–[21].

Some prior research has found an approximate regularizing effect of adding white noise to backpropagation [22]–[25]. We instead add non-white noise that satisfies a simple geometric condition that depends on both the network parameters and the output activations.

The next section casts the backpropagation algorithm as ML estimation. Section III presents the EM algorithm for neural network training and proves that it reduces to the backpropagation algorithm. Section IV reviews the NEM theorem. Section V proves the noise benefit sufficient conditions for a neural network. Section VI reviews RBMs or BAMs. Section VII derives sufficient conditions for a noise benefit in ML training of Bernoulli-Bernoulli and Gaussian-Bernoulli RBMs. Section VIII presents simulation results.
Fig. 1. This figure shows the training-set squared error for backpropagation and NEM-backpropagation (NEM-BP) training of an auto-encoder neural network on the MNIST digit classification data set. There is a 5.3% median decrease in the squared error per iteration for NEM-BP when compared with backpropagation training. We added annealed independent and identically-distributed (i.i.d.) Gaussian noise to the target variables. The noise had mean \( \mathbf{a}^t - \mathbf{t} \) and a variance that decayed with the training epochs as \{0.1, 0.1/2, 0.1/3, . . . \} where \( \mathbf{a}^t \) is the vector of activations of the output layer and \( \mathbf{t} \) is the vector of target values. The network used three logistic (sigmoidal) hidden layers with 20 neurons each. The output layer used 784 logistic neurons.

II. BACKPROPAGATION AS MAXIMUM LIKELIHOOD ESTIMATION

We show that the backpropagation algorithm performs ML estimation of a neural network’s parameters. We use a 3-layer neural network for notational convenience. All results in this paper extend to “deep” networks with more hidden layers. \( \mathbf{x} \) denotes the neuron values at the input layer that consists of \( I \) neurons. \( \mathbf{a}^h \) is the vector of hidden neuron sigmoidal activations whose \( j^{th} \) element is

\[
a^h_j = \frac{1}{1 + \exp \left( - \sum_{i=1}^{I} w_{ji} x_i \right)} = \sigma \left( \sum_{i=1}^{I} w_{ji} x_i \right)
\]  

where \( w_{ji} \) is the weight of the link that connects the \( j^{th} \) visible and \( j^{th} \) hidden neuron. \( \mathbf{y} \) denotes the \( K \)-valued target variable and \( \mathbf{t} \) is its 1-in-\( K \) encoding. \( t_k \) is the \( k^{th} \) output neuron’s value with activation

\[
a^t_k = \frac{1}{\exp \left( \sum_{j=1}^{J} u_{kj} a^h_j \right) + 1} = \sigma \left( \sum_{j=1}^{J} u_{kj} a^h_j \right)
\]  

where \( u_{kj} \) is the weight of the link that connects the \( j^{th} \) hidden and \( k^{th} \) target neuron. \( a^h_k \) depends on input \( \mathbf{x} \) and parameter matrices \( \mathbf{U} \) and \( \mathbf{W} \). Backpropagation minimizes the following cross entropy:

\[
E = - \sum_{k=1}^{K} t_k \log (a^t_k) .
\]  

The cross-entropy equals the negative conditional log-likelihood \( L \) of the target \( y \) given the inputs \( \mathbf{x} \) because

\[
E = - \log \left( \prod_{k=1}^{K} (a^t_k)^{t_k} \right) = - \log \left( \prod_{k=1}^{K} p(y = k|\mathbf{x}, \Theta)^{t_k} \right)
\]  

\[
= - \log p(y|\mathbf{x}, \Theta) = -L .
\]  

Backpropagation updates the network parameters \( \Theta \) using gradient ascent to maximize the log likelihood \( \log p(y|\mathbf{x}, \Theta) \). The partial derivative of this log-likelihood with respect to \( u_{kj} \) is

\[
\frac{\partial L}{\partial u_{kj}} = (t_k - a^t_k) a^h_j
\]  

and with respect to \( w_{ji} \) is

\[
\frac{\partial L}{\partial w_{ji}} = a^h_j (1 - a^h_j) x_i \sum_{k=1}^{K} (t_k - a^t_k) u_{kj} .
\]  

So (7) and (8) give the partial derivatives that perform gradient ascent on the log-likelihood \( L \).

A linear signal function often replaces the Gibbs function at the output layer for regression:

\[
a^t_k = \sum_{j=1}^{J} u_{kj} a^h_j .
\]  

The target values \( t \) of the output neuron layer can assume any real values for regression. Then backpropagation minimizes
the following squared-error function:

\[ E = \frac{1}{2} \sum_{k=1}^{K} (t_k - a_k^t)^2. \]  
(10)

We assume that \( t \) is Gaussian with mean \( a^t \) and identity covariance matrix \( I \). Then backpropagation maximizes the following log-likelihood function:

\[ L = \log p(t|x, \Theta) = \log \mathcal{N}(t; a^t, I) \]  
(11)

for

\[ \mathcal{N}(t; a^t, I) = \frac{1}{(2\pi)^{d/2}} \exp \left\{ -\frac{1}{2} \sum_{k=1}^{K} (t_k - a_k^t)^2 \right\}. \]  
(12)

Thus the gradient partial derivatives of this log-likelihood function are the same as those for the \( K \)-class classification case in (7) and (8).

### III. EM ALGORITHM FOR NEURAL NETWORK ML ESTIMATION

Both backpropagation and the EM algorithm find the ML estimate of a neural network’s parameters. The next theorem shows that backpropagation is a special case of the generalized EM algorithm.

**Theorem 1. Backpropagation is the GEM Algorithm**

The backpropagation update equation for a differentiable likelihood function \( p(y|x, \Theta) \) at epoch \( n \)

\[ \Theta^{n+1} = \Theta^n + \eta \nabla_{\Theta} \log p(y|x, \Theta)|_{\Theta=\Theta^n} \]  
(13)

equals the GEM update equation at epoch \( n \)

\[ \Theta^{n+1} = \Theta^n + \eta \nabla_{\Theta} Q(\Theta|\Theta^n)|_{\Theta=\Theta^n} \]  
(14)

where GEM uses the differentiable Q-function

\[ Q(\Theta|\Theta^n) = \mathbb{E}_{p(h|x, y, \Theta^n)} \left\{ \log p(y, h|x, \Theta) \right\}. \]  
(15)

**Proof:** We know that [3], [26]

\[ \log p(y|x, \Theta) = Q(\Theta|\Theta^n) + H(\Theta|\Theta^n) \]  
(16)

if \( H(\Theta|\Theta^n) \) is the following cross entropy [27]:

\[ H(\Theta|\Theta^n) = -\int \log p(h|x, y, \Theta) \, dp(h|x, y, \Theta^n). \]  
(17)

Hence

\[ H(\Theta|\Theta^n) = \log p(y|x, \Theta) - Q(\Theta|\Theta^n). \]  
(18)

Now expand the Kullback-Leibler divergence [28]:

\[ D_{\text{KL}}(\Theta^n||\Theta) = \int \log \frac{p(h|x, y, \Theta^n)}{p(h|x, y, \Theta^n)} \, dp(h|x, y, \Theta^n) \]  
(19)

\[ = \int \log p(h|x, y, \Theta^n) \, dp(h|x, y, \Theta^n) \]  
(20)

\[ - \int \log p(h|x, y, \Theta^n) \, dp(h|x, y, \Theta^n) \]  
(21)

So \( H(\Theta|\Theta^n) \geq H(\Theta^n|\Theta^n) \) for all \( \Theta \) because \( D_{\text{KL}}(\Theta^n||\Theta) \geq 0 \) [28]. Thus \( \Theta^n \) minimizes \( H(\Theta|\Theta^n) \) and hence \( \nabla_{\Theta} H(\Theta|\Theta^n) = 0 \) at \( \Theta = \Theta^n \). Putting this in (18) gives

\[ \nabla_{\Theta} \log p(y|x, \Theta)|_{\Theta=\Theta^n} = \nabla_{\Theta} Q(\Theta|\Theta^n)|_{\Theta=\Theta^n}. \]  
(22)

Hence the backpropagation and GEM update equations are identical.
The GEM algorithm uses a probabilistic description of the hidden layer neurons. We assume that the hidden layer neurons are Bernoulli random variables. So their activation is the following conditional probability:

$$a_j^h = p(h_j = 1|x, \Theta) \ .$$

(23)

We can now formulate an EM algorithm for ML estimation of a feedforward neural network’s parameters. The E-step computes the Q-function in (15). Computing the expectation in (15) requires $2^J$ values of $p(h|x, y, \Theta^n)$. This is computationally-intensive for large values of $J$. So we use Monte Carlo sampling to approximate the above Q-function. The strong law of large numbers ensures that this Monte Carlo approximation converges almost surely to the true Q-function. Bayes theorem gives $p(h|x, y, \Theta^n)$ as

$$p(h|x, y, \Theta^n) = \frac{p(h|x, \Theta^n)p(y|h, \Theta^n)}{\sum_h p(h|x, \Theta^n)p(y|h, \Theta^n)} \ .$$

(24)

We can sample more easily from $p(h|x, \Theta^n)$ than from $p(h|x, y, \Theta^n)$ because the $h_j$ terms are independent given $x$. We replace $p(h|x, \Theta^n)$ by its Monte Carlo approximation using $M$ independent and identically-distributed (i.i.d.) samples:

$$p(h|x, \Theta^n) \approx \frac{1}{M} \sum_{m=1}^{M} \delta_K (h - h^m) \ .$$

(25)

where $\delta_K$ is the $J$-dimensional Kronecker delta function. The Monte Carlo approximation of the hidden data conditional PDF becomes

$$p(h|x, y, \Theta^n) \approx \frac{\sum_{m=1}^{M} \delta_K (h - h^m)p(y|h, \Theta^n)}{\sum_h \sum_{m=1}^{M} \delta_K (h - h^m)p(y|h, \Theta^n)} \ .$$

(26)

$$= \frac{\sum_{m=1}^{M} \delta_K (h - h^m)p(y|h^m, \Theta^n)}{\sum_{m=1}^{M} p(y|h^m, \Theta^n)} \ .$$

(27)

$$= \sum_{m=1}^{M} \delta_K (h - h^m) \gamma^m \ .$$

(28)

where $\gamma^m = \frac{p(y|h^m, \Theta^n)}{\sum_{m=1}^{M} p(y|h^m, \Theta^n)}$ is the “importance” of $h^m$. (28) gives an importance-sampled approximation of $p(h|x, y, \Theta^n)$ where each sample $h^m$ has weight $\gamma^m$. We can now approximate the Q-function as

$$Q(\Theta|\Theta^n) \approx \sum_h \sum_{m=1}^{M} \gamma^m \delta_K (h - h^m) \log p(y|h, x, \Theta) \ .$$

(30)

$$= \sum_{m=1}^{M} \gamma^m \log p(y, h^m|x, \Theta) \ .$$

(31)

$$= \sum_{m=1}^{M} \gamma^m \left[ \log p(h^m|x, \Theta) + \log p(y|h^m, \Theta) \right] \ .$$

(32)

where

$$\log p(h^m|x, \Theta) = \sum_{j=1}^{J} [h_j^m \log a_j^h + (1 - h_j^m) \log (1 - a_j^h)] \ .$$

(33)

for sigmoidal hidden layer neurons. Gibbs activation neurons at the output layer give

$$\log p(y|h^m, \Theta) = \sum_{k=1}^{K} t_k \log a_k^{mt} \ .$$

(34)

where $a_k^{mt}$ is as in (1) and

$$a_k^{mt} = \frac{\exp \left( \sum_{j=1}^{J} u_{kj} a_j^{mh} \right)}{\sum_{k=1}^{K} \exp \left( \sum_{j=1}^{J} u_{kj} a_j^{mh} \right)} \ .$$

(35)

Gaussian output layer neurons give

$$\log p(y|h^m, \Theta) = -\frac{1}{2} \sum_{k=1}^{K} (t_k - a_k^{mt})^2 \ .$$

(36)

The Q-function in (32) equals a sum of log-likelihood functions for two 2-layer neural networks between the visible-hidden and hidden-output layers. The M-step maximizes this Q-function by gradient ascent. So it is equivalent to two distinct backpropagation steps performed on these two 2-layer neural networks.

IV. THE NOISY EXPECTATION-MAXIMIZATION THEOREM

The Noisy Expectation-Maximization (NEM) algorithm [4], [5] modifies the EM scheme and achieves faster convergence times on average. The NEM algorithm injects additive noise into the data at each EM iteration. The noise decays with the iteration count to guarantee convergence to the optimal parameters of the original data model. The additive noise must also satisfy the NEM condition below that guarantees that the NEM parameter estimates will climb faster up the likelihood surface on average.

A. NEM THEOREM

The NEM Theorem [4], [5] states a general sufficient condition when noise speeds up the EM algorithm’s convergence to a local optimum. The NEM Theorem uses the following notation. The noise random variable $N$ has pdf $p(n|x)$. So the noise $N$ can depend on the data $x$. $h$ are the latent variables in the model. $\{\Theta^n\}$ is a sequence of EM estimates for $\Theta$. $\Theta_* = \lim_{n \to \infty} \Theta^n$ is the converged EM estimate for $\Theta$. Define the noisy Q function $Q_N(\Theta|\Theta^n) = E_{h|x, \Theta_*} [\log p(x + N, h|\Theta)]$. Assume that the differential entropy of all random variables is finite and that the additive noise keeps the data in the likelihood function’s support. Then we can state the NEM theorem [4], [5].

Theorem 2. Noisy Expectation Maximization (NEM) The EM estimation iteration noise benefit

$$Q(\Theta_*|\Theta_* - Q(\Theta^n|\Theta_*) \geq Q(\Theta_*|\Theta_* - Q_N(\Theta^n|\Theta_*) \ .$$

(37)
or equivalently
\[ Q_N(\Theta^{(n)}|\Theta_s) \geq Q(\Theta^{(n)}|\Theta_+) \] holds on average if the following positivity condition holds:
\[ E_{x,h,N|\Theta^*} \left[ \ln \left( \frac{p(x + N, h|\Theta^*)}{p(x, h|\Theta^*)} \right) \right] \geq 0 \, . \] (39)

The NEM Theorem states that each iteration of a suitably noisy EM algorithm gives higher likelihood estimates on average than do the regular EM’s estimates. So the NEM algorithm converges faster than EM if we can identify the data model. The faster NEM convergence occurs both because the likelihood function has an upper bound and because the NEM algorithm takes larger average steps up the likelihood surface.

Maximum A Posteriori (MAP) estimation for missing information problems can use a modified version of the EM algorithm. The MAP version modifies the \( Q \)-function by adding a log prior term \( G(\Theta) = \ln p(\Theta) \) [29], [30]:
\[ Q(\Theta|\Theta_s) = E_{h|x,\Theta_s}[\ln p(x + N, h|\Theta_s)] + G(\Theta) \, . \] (40)

The MAP version of the NEM algorithm applies a similar modification to the \( Q \)-function:
\[ Q_N(\Theta|\Theta_s) = E_{h|x,\Theta_s}[\ln p(x + N, h|\Theta_s)] + G(\Theta) \, . \] (41)

Many latent-variable models are not identifiable [33] and thus need not have global optima. These models include Gaussian mixture models [31], hidden Markov models [32], and neural networks. The EM and NEM algorithms converge to local optima in these cases. The additive noise in the NEM algorithm helps the NEM estimates search other nearby local optima. The NEM Theorem still guarantees that NEM estimates have higher likelihood on average than EM estimates for non-identifiable models.

V. NOISE BENEFITS IN NEURAL NETWORK ML ESTIMATION

Consider adding noise \( n \) to the 1-in-\( K \) encoding \( t \) of the target variable \( y \). We first present the noise benefit sufficient condition for Gibbs activation output neurons used in \( K \)-class classification.

Theorem 3. Forbidden Hyperplane Noise Benefit Condition
The NEM positivity condition holds for ML training of feedforward neural network with Gibbs activation output neurons if
\[ E_{t,h,n|x,\Theta^*}\{n^T \log(a^t)\} \geq 0 \, . \] (42)

Proof: We add noise to the target 1-in-\( K \) encoding \( t \). The likelihood ratio in the NEM sufficient condition becomes
\[ \frac{p(t + n, h|x, \Theta)}{p(t, h|x, \Theta)} = \frac{p(t + n|h, \Theta)p(h|x, \Theta)}{p(t|h, \Theta)p(h|x, \Theta)} \]
\[ = \frac{p(t + n|h, \Theta)}{p(t|h, \Theta)} \]
\[ = \prod_{k=1}^{K} \frac{(a_k^t)^{t_k+n_k}}{(a_k^t)^{t_k}} = \prod_{k=1}^{K} (a_k^t)^{n_k} \, . \] (45)

So the NEM positivity condition becomes
\[ E_{t,h,n|x,\Theta^*}\log \left( \prod_{k=1}^{K} (a_k^t)^{n_k} \right) \geq 0 \] (46)

This condition is equivalent to
\[ E_{t,h,n|x,\Theta^*}\left( \sum_{k=1}^{K} n_k \log(a_k^t) \right) \geq 0 \] (47)

We can rewrite this positivity condition as the following matrix inequality:
\[ E_{t,h,n|x,\Theta^*}\{n^T \log(a^t)\} \geq 0 \] (48)

where \( \log(a^t) \) is the vector of output neuron log-activations.

The above sufficient condition requires that the noise \( n \) lie above a hyperplane with normal \( \log(a^t) \). The next theorem gives a sufficient condition for a noise benefit in the case of Gaussian output neurons.

Theorem 4. Forbidden Sphere Noise Benefit Condition
The NEM positivity condition holds for ML training of a feedforward neural network with Gaussian output neurons if
\[ E_{t,h,n|x,\Theta^*}\{\|n - a^t + t\|^2 - \|a^t - t\|^2\} \leq 0 \] (49)

where \( ||.|| \) is the \( L_2 \) vector norm.

Proof: We add noise \( n \) to the \( K \) output neuron values \( t \). The log-likelihood in the NEM sufficient condition becomes
\[ \frac{p(t + n, h|x, \Theta)}{p(t, h|x, \Theta)} = \frac{p(t + n|h, \Theta)p(h|x, \Theta)}{p(t|h, \Theta)p(h|x, \Theta)} \]
\[ = \frac{N(t + n, a^t, I)}{N(t; a^t, I)} \]
\[ = \exp \left( \frac{1}{2} \|t - a^t\|^2 - \|t + n - a^t\|^2 \right) \] (52)

So the NEM sufficient condition becomes
\[ E_{t,h,n|x,\Theta^*}\{\|n - a^t + t\|^2 - \|a^t - t\|^2\} \leq 0 \] (53)

The above sufficient condition defines a forbidden noise region outside a sphere with center \( t - a^t \) and radius \( \|t - a^t\| \). All noise inside this sphere speeds convergence of ML estimation in the neural network on average.

This section presented sufficient conditions for a noise benefit in training a neural network that uses the EM algorithm. We now discuss noise benefit in pre-training or initialization of the parameters of a neural network using RBMs or the equivalent BAMs.

VI. TRAINING BAMs OR RESTRICTED BOLTZMANN MACHINES

Restricted Boltzmann Machines [7], [8] are a special type of bidirectional associative memory (BAM) [1], [10], [11]. So they enjoy rapid convergence to a bidirectional fixed point for synchronous updating of the neurons. A BAM is a two-layer heteroassociative network that uses the synaptic
connection matrix $M$ on the forward pass of the neuronal signals from the lower layer to the higher layer but also uses the transpose matrix $M^T$ on the backward from the higher layer to the lower layer. The lower layer is visible during training of “deep” neural networks [8] while the higher field is hidden. The general BAM Theorem ensures that any such matrix $M$ is bidirectionally stable for threshold neurons as well for most continuous neurons. Logistic neurons satisfy the BAM Theorem because logistic signal functions are bounded and monotone decreasing. The following results use the term RBM instead of BAM for simplicity.

Consider an RBM with $I$ visible neurons and $J$ hidden neurons. Let $x_i$ and $h_j$ denote the values of the $i$th visible and $j$th hidden neuron. Let $E(x, h; \Theta)$ be the energy function for the network. Then the joint probability density function of $x$ and $h$ is the Gibbs distribution:

$$p(x, h; \Theta) = \frac{\exp \left( - E(x, h; \Theta) \right)}{Z(\Theta)}$$

where $Z(\Theta) = \sum_x \sum_h \exp(-E(x,h;\Theta))$. 

Integrals replace sums for continuous variables in the above partition function $Z(\Theta)$. The Gibbs energy function $E(v, h; \Theta)$ depends on the type of RBM. A Bernoulli(visible)-Bernoulli(hidden) RBM has logistic conditional PDFs at the hidden and visible layers and has the following BAM energy or Lyapunov function [1], [10], [11]:

$$E(x, h; \Theta) = -\sum_{i=1}^I \sum_{j=1}^J w_{ij} x_i h_j - \sum_{i=1}^I b_i x_i - \sum_{j=1}^J a_j h_j$$

where $w_{ij}$ is the weight of the connection between the $i$th visible and $j$th hidden neuron, $b_i$ is the bias for the $i$th visible neuron, and $a_j$ is the bias for the $j$th hidden neuron. A Gaussian(visible)-Bernoulli(hidden) RBM has Gaussian conditional PDFs at the visible layer, logistic conditional PDFs at the hidden layer, and the energy function [8, 34]

$$E(x, h; \Theta) = -\sum_{i=1}^I \sum_{j=1}^J w_{ij} x_i h_j + \frac{1}{2} \sum_{i=1}^I (x_i - b_i)^2 - \sum_{j=1}^J a_j h_j .$$

The neural network uses an RBM or BAM as a building block. The systems finds ML estimates of each RBM’s parameters and then stacks up the resulting RBMs on top of each other. Then backpropagation trains the neural network. The next section reviews ML training for an RBM.

A. ML Training for RBMs using Contrastive Divergence

The maximum likelihood (ML) estimate of the parameters $\Theta$ for a RBM is

$$\Theta^* = \arg \max_{\Theta} \log p(x; \Theta) .$$

Gradient ascent can iteratively solve this optimization problem. We estimate $w_{ij}$ in the quadratic forms in (53) and (54) because the terms are the same for a Bernoulli-Bernoulli and Gaussian-Bernoulli RBM. The gradient of $\log p(x; \Theta)$ with respect to $w_{ij}$ is

$$\frac{\partial \log p(x; \Theta)}{\partial w_{ij}} = E_{p(h|x, \Theta)} \{ x_i, h_j \} - E_{p(x, h; \Theta)} \{ x_i, h_j \}.$$ (59)

So the update rule for $w_{ij}$ at iteration $(n+1)$ becomes

$$w_{ij}^{(n+1)} = w_{ij}^{(n)} + \eta \left( E_{p(h|x, \Theta^{(n)})} \{ x_i, h_j \} - E_{p(x, h; \Theta^{(n)})} \{ x_i, h_j \} \right) .$$ (60)

where $\eta > 0$ is the learning rate. We can easily compute $p(h|x, \Theta^{(n)})$ for the RBM because there are no connections between any two hidden or two visible neurons. This gives the expectation $E_{p(h|x, \Theta^{(n)})} \{ x_i, h_j \}$. But we cannot so easily compute $p(x, h|\Theta^{(n)})$ due to the partition function $Z(\Theta)$ in (55). Contrastive divergence (CD) [8] approximates $Z(\Theta)$ through activations that derive from a forward and a backward pass in the RBM.

B. ML Training for RBMs using the EM algorithm

The EM algorithm provides an iterative method for learning RBM parameters. Consider an RBM with $I$ visible neurons and $J$ hidden neurons. EM maximizes a simpler lower bound on $\log p(x; \Theta)$ because the log-likelihood can be intractable to compute. This lower bound at $\Theta = \Theta^{(n)}$ is

$$Q(\Theta|\Theta^{(n)}) = E_{h|x, \Theta^{(n)}} \{ \log p(x, h; \Theta) \}$$

$$= E_{h|x, \Theta^{(n)}} \{ -E(x, h; \Theta) - \log Z(\Theta) \} .$$ (61) (62)

A generalized EM (GEM) algorithm uses gradient descent to iteratively maximize the above Q-function. The gradient with respect to $w_{ij}$ is

$$\frac{\partial Q(\Theta|\Theta^{(n)})}{\partial w_{ij}} = \frac{\partial E_{h|x, \Theta^{(n)}} \{ -E(x, h; \Theta) - \log Z(\Theta) \}}{\partial w_{ij}}$$

$$= E_{h|x, \Theta^{(n)}} \left\{ \frac{\partial E(x, h; \Theta)}{\partial w_{ij}} - \frac{\partial \log Z(\Theta)}{\partial w_{ij}} \right\}$$

$$= E_{h|x, \Theta^{(n)}} \left\{ x_i h_j - \frac{1}{Z(\Theta)} \frac{\partial Z(\Theta)}{\partial w_{ij}} \right\} .$$ (63) (64) (65)
But the partition function term expands as

$$
\frac{1}{Z(\Theta)} \frac{\partial Z(\Theta)}{\partial w_{ij}} = \frac{1}{Z(\Theta)} \sum_{x} \sum_{h} \frac{\partial \{ x, h \} / \partial w_{ij}}{Z(\Theta)}
$$

(66)

$$
= \frac{1}{Z(\Theta)} \sum_{x} \sum_{h} \left\{ \frac{\partial \{ x, h \} / \partial w_{ij}}{Z(\Theta)} \right\}
$$

(67)

$$
= \frac{1}{Z(\Theta)} \sum_{x} \sum_{h} \left\{ \exp(-E(x, h; \Theta)) x_i h_j \right\}
$$

(68)

$$
= \frac{1}{Z(\Theta)} \sum_{x} \sum_{h} \left\{ \exp(-E(x, h; \Theta)) x_i h_j \right\}
$$

(69)

$$
= \sum_{x} \sum_{h} \left\{ p(x, h; \Theta) x_i h_j \right\} = E_{x, h|\Theta} \{ x_i h_j \} .
$$

(70)

So the partial derivative of the Q-function becomes

$$
\frac{\partial Q(\Theta)(\text{n})}{\partial w_{ij}} = E_{h|x, \Theta(n)} \{ x_i h_j - E_{x, h|\Theta} \{ x_i h_j \} \}
$$

(71)

$$
= E_{h|x, \Theta(n)} \{ x_i h_j \} - E_{x, h|\Theta} \{ x_i h_j \} .
$$

(72)

This leads to the key GEM gradient ascent equation:

$$
w_{ij}^{(n+1)} = w_{ij}^{(n)} + \eta \left( E_{p(h|x, \Theta(n))} \{ x_i h_j \} - E_{x, h|\Theta} \{ x_i h_j \} \right).
$$

(73)

The above update equation is the same as the contrastive divergence update equation in (60). The next section shows that this equivalence between CD and GEM lets us to derive a NEM sufficient condition for RBM training.

VII. Noise Benefits in RBM ML Estimation

Consider now addition of noise \( n \) to the input data \( x \). A NEM noise benefit exists if

$$
E_{x, h, n|\Theta} \left\{ \log \left( \frac{p(x + n, h; \Theta)}{p(x, h; \Theta)} \right) \right\} \geq 0 .
$$

(74)

The noisy complete data likelihood is

$$
p(x + n, h; \Theta) = \frac{\exp(-E(x + n, h; \Theta))}{Z(\Theta)}. \tag{75}
$$

So a NEM noise benefit for an RBM occurs if

$$
E_{x, h, n|\Theta} \left\{ \log \left( \frac{\exp(-E(x + n, h; \Theta))}{\exp(-E(x, h; \Theta))} \right) \right\} \geq 0 .
$$

(76)

This is equivalent to the RBM noise benefit inequality:

$$
E_{x, h, n|\Theta} \left\{ -E(x + n, h; \Theta) + E(x, h; \Theta) \right\} \geq 0 . \tag{77}
$$

Theorem 5. Forbidden Hyperplane Noise Benefit Condition

The NEM positivity condition holds for Bernoulli-Bernoulli RBM training if

$$
E_{x, h, n|\Theta} \left\{ n^T (W h + b) \right\} \geq 0 . \tag{78}
$$

Proof: The noise benefit for a Bernoulli(visual)-Bernoulli(hidden) RBM results if we apply the energy function from (56) to the expectation in (77) to get

$$
E_{x, h, n|\Theta} \left\{ \sum_{i=1}^l \sum_{j=1}^J w_{ij} n_i h_j + \sum_{i=1}^l n_i b_i \right\} \geq 0 . \tag{79}
$$

The term in brackets is equivalent to

$$
\sum_{i=1}^l \sum_{j=1}^J w_{ij} n_i h_j + \sum_{i=1}^l n_i b_i = n^T (W h + b) . \tag{80}
$$

So the noise benefit sufficient condition becomes

$$
E_{x, h, n|\Theta} \left\{ n^T (W h + b) \right\} \geq 0 . \tag{81}
$$

The above sufficient condition is similar to the hyperplane condition for neural network training in Theorem 3. All noise above a hyperplane based on the RBM’s parameters gives a noise benefit. The next theorem states a spherical separation condition that guarantees a noise benefit in the Bernoulli-Bernoulli RBM.

Theorem 6. Forbidden Sphere Noise Benefit Condition

The NEM positivity condition holds for Gaussian-Bernoulli RBM training if

$$
E_{x, h, n|\Theta} \left\{ \frac{1}{2} \| n \|^2 - n^T (W h + b - x) \right\} \leq 0 . \tag{82}
$$

Proof: Putting the energy function in (57) into (77) gives the noise benefit condition for a Gaussian(visual)-Bernoulli(hidden):

$$
E_{x, h, n|\Theta} \left\{ \sum_{i=1}^l \sum_{j=1}^J w_{ij} n_i h_j + \sum_{i=1}^l n_i b_i - \frac{1}{2} \sum_{i=1}^l n_i^2 \right\} \tag{83}
$$

(79)

$$
\sum_{i=1}^l n_i x_i \right\} \geq 0 .
$$

The term in brackets equals the following matrix expression:

$$
\sum_{i=1}^l \sum_{j=1}^J w_{ij} n_i h_j + \sum_{i=1}^l n_i b_i - \frac{1}{2} \sum_{i=1}^l n_i^2 - \sum_{i=1}^l n_i x_i
$$

(84)

$$
= n^T (W h + b - x) - \frac{1}{2} \| n \|^2 . \tag{84}
$$

So the noise benefit sufficient condition becomes

$$
E_{x, h, n|\Theta} \left\{ \frac{1}{2} \| n \|^2 - n^T (W h + b - x) \right\} \leq 0 . \tag{85}
$$

The above condition bisects the noise space. But the bisecting surface for (82) is a hypersphere. This condition is also similar to the noise benefit condition for neural network ML training in Theorem 4.
VIII. SIMULATION RESULTS

We modified the Matlab code available in [35] to inject noise during EM-backpropagation training of a neural network. We used 10,000 training instances from the training set of the MNIST digit classification data set. Each image in the data set had 28 × 28 pixels with each pixel value lying between 0 and 1. We fed each pixel into the input neuron of a neural network. We used a 5-layer neural network with 20 neurons in each of the three hidden layers and 10 neurons in the output layer for classifying the 10 digits. We also trained an auto-encoder neural network with 20 neurons in each of the three hidden layers and 784 neurons in the output layer for estimating the pixels of a digit's image.

The output layer used the Gibbs activation function for the 10-class classification network and logistic activation function for the auto-encoder. We used logistic activation functions in all other layers. Simulations used 10 Monte Carlo samples for approximating the Q-function in the 10-class classification network and 5 Monte Carlo samples for the auto-encoder. Figure 1 shows the training-set squared error for the auto-encoder neural network for backpropagation and NEM-backpropagation when we added annealed Gaussian noise with mean \( \mu = 1 \) and variance 0.1 epoch\(^{-1} \). Figure 4 shows the training-set cross entropy for the two cases when we added annealed Gaussian noise with mean 0 and variance 0.2 epoch\(^{-1} \). We used 10 Monte Carlo samples to approximate the Q-function. We observed a 5.3\% median decrease in squared error and 4.2\% median decrease in cross entropy per iteration for the NEM-backpropagation algorithm compared to standard backpropagation.

IX. CONCLUSIONS

The backpropagation algorithm is a special case of the generalized EM algorithm. So proper noise injection speeds backpropagation convergence because it speeds EM convergence. These sufficient conditions use the recent noisy EM (NEM) theorem. Similar sufficient conditions hold for a noise benefit in pre-training neural networks based on the NEM theorem. Noise-injection simulations on the MNIST digit recognition data set reduced both network squared error and cross entropy.

REFERENCES
