Neural Networks 129 (2020) 359–384

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Noise can speed backpropagation learning and deep bidirectional pretraining

Bart Kosko [a,*], Kartik Audhkhasi [c,a], Osonde Osoba [b,a]

[a] Department of Electrical and Computer Engineering, Signal and Image Processing Institute, University of Southern California, Los Angeles, CA 90089-2564, USA
[b] RAND Corporation, Santa Monica, CA 90401-3208, USA
[c] Google, Inc., NY, USA

ARTICLE INFO

Article history:
Received 11 March 2019
Received in revised form 19 February 2020
Accepted 2 April 2020
Available online 11 April 2020

Keywords:
Backpropagation
Noise benefit
Stochastic resonance
Expectation–Maximization algorithm
Bidirectional associative memory
Contrastive divergence

ABSTRACT

We show that the backpropagation algorithm is a special case of the generalized Expectation–Maximization (EM) algorithm for iterative maximum likelihood estimation. We then apply the recent result that carefully chosen noise can speed the average convergence of the EM algorithm as it climbs a hill of probability or log-likelihood. Then injecting such noise can speed the average convergence of the backpropagation algorithm for both the training and pretraining of multilayer neural networks. The beneficial noise adds to the hidden and visible neurons and related parameters. The noise also applies to regularized regression networks. This beneficial noise is just that noise that makes the current signal more probable. We show that such noise also tends to improve classification accuracy. The geometry of the noise-benefit region depends on the probability structure of the neurons in a given layer. The noise-benefit region in noise space lies above the noisy-EM (NEM) hyperplane for classification and involves a hypersphere for regression. Simulations demonstrate these noise benefits using MNIST digit classification. The NEM noise benefits substantially exceed those of simply adding blind noise to the neural network. We further prove that the noise speed-up applies to the deep bidirectional pretraining of neural-network bidirectional associative memories (BAMs) or their functionally equivalent restricted Boltzmann machines. We then show that learning with basic contrastive divergence also reduces to generalized EM for an energy-based network probability. The optimal noise adds to the input visible neurons of a BAM in stacked layers of trained BAMs. Global stability of generalized BAMs guarantees rapid convergence in pretraining where neural signals feed back between contiguous layers. Bipolar coding of inputs further improves pretraining performance.

1. Noise benefits in backpropagation

We generalize and extend the recent result (Audhkhasi, Osoba, & Kosko, 2016) that the backpropagation (BP) algorithm (Rumelhart, Hinton, & Williams, 1986; Werbos, 1974) is a special case of the generalized Expectation–Maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977). The new result extends to what we call *BP invariance*: The parameter gradient of the neural network's layer log-likelihood L must give back the BP learning laws for that layer. We demonstrate this BP invariance for classification and regression as well as for logistic networks.

We then show how noise can boost BP based on the general noise-boosting strategy for EM. This allows EM-based noise injection into the hidden layers as well as into the output layers as in Audhkhasi et al. (2016). This EM-based noise takes different forms for classification and regression networks because of BP invariance. The injected EM-based noise differs from the simple blind white noise or dither of earlier noise-injection schemes. It is just that noise that makes the current signal more likely on average. Simulations on the MNIST image data set confirm that this noise-boosted BP climbs the nearest hill of likelihood faster on average than does noiseless BP or dithered BP. It also tends to improve classification accuracy. A new discrete convergence theorem for bidirectional associative memories shows that contrastive-divergence learning in such associative memories or restricted Boltzmann machines is also a form of generalized EM. We then derive sufficient conditions for noise-boosting contrastive divergence learning in pretraining for logistic and Gaussian layers.

BP remains the workhorse of neural networks and deep learning (Gulshan et al., 2016; Hinton, 2018; Jordan & Mitchell, 2015; LeCun, Bengio, & Hinton, 2015; Schmidhuber, 2015). EM performs maximum likelihood estimation for the general case of missing

* Corresponding author.
E-mail address: kosko@usc.edu (B. Kosko).

https://doi.org/10.1016/j.neunet.2020.04.004
0893-6080/

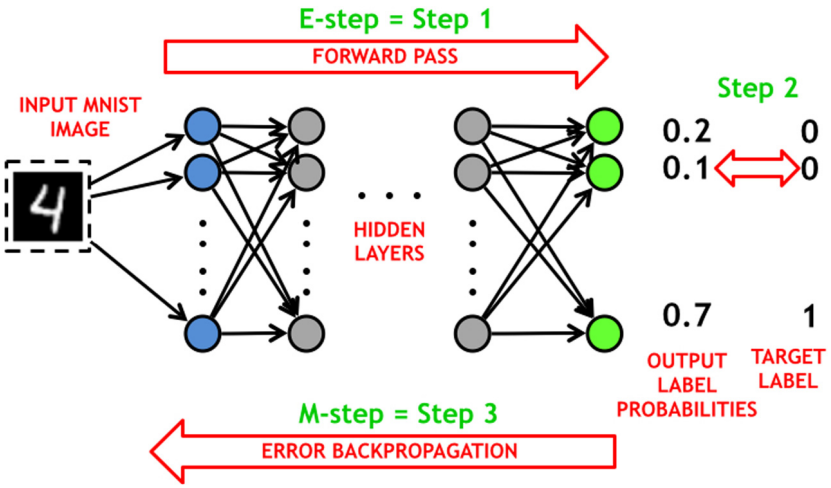

Fig. 1. Backpropagation as generalized Expectation–Maximization. The diagram shows how the backpropagation (BP) algorithm behaves as the general Expectation–Maximization (EM) algorithm when it recognizes different digits after training on handwritten MNIST digit samples. The forward pass of input data through the neural network corresponds to the Expectation (E) step. The backpropagation of the gradients corresponds to the Maximization (M) step. BP's hidden units correspond to EM's hidden or latent variables. Theorem 1 states the formal equivalence between BP and EM in terms of their gradient updates.

data or hidden parameters (Dempster et al., 1977; McLachlan & Krishnan, 2007; Moon, 1996; Xu & Wunsch, 2008).

BP remains a popular way to attack large-scale problems of pattern recognition and signal processing. BP scales well because its time complexity is only $O(n)$ for n training samples. This holds because both the forward and backward passes have $O(n)$ time complexity during training. Support vector machines and other kernel methods have $O(n^2)$ complexity in general (Kung, 2014). Key BP applications include speech recognition (Dahl, Ranzato, Mohamed, & Hinton, 2010; Mohamed, Dahl, & Hinton, 2009, 2012; Mohamed et al., 2011; Mohamed, Yu, & Deng, 2010; Sainath et al., 2011; Seide, Li, & Yu, 2011), machine translation of text (Deselaers, Hasan, Bender, & Ney, 2009), audio processing (Hamel & Eck, 2010), artificial intelligence (Bengio, 2009), computer vision (Ciresan, Meier, Gambardella, & Schmidhuber, 2010; Nair & Hinton, 2009; Susskind, Hinton, Movellan, & Anderson, 2008), medicine (Hu et al., 2013), biomedical modeling (Guo, Zhou, Nie, Ruan, & Li, 2019; Hou, Zhou, Nie, Liu, & Ruan, 2019), and general multilayered or deep learning (Jordan & Mitchell, 2015; LeCun et al., 2015).

We generalize and extend the BP-as-EM theorem and then use it to speed the average convergence of the BP training of multilayer neural networks for both classification and regression. The beneficial noise must satisfy a likelihood-based inequality in all cases. We also show that this EM-based noise also tends to improve classification accuracy. Simulations on MNIST handwritten digit data confirm that this noise benefit substantially exceeds the slight benefit (if any) of adding small amounts of blind noise to the neural network. The MNIST data set is the Modified National Institute of Standards and Technology image data set of the ten handwritten digits $0, 1, \ldots, 9$. The data set contains 60,000 digitized images for training and 10,000 images for testing.

Related theorems show that similarly chosen noise can speed the bidirectional pre-training of stacked layers. They show further that contrastive-divergence is also a form of generalized EM. We present a discrete bidirectional-associative-memory convergence theorem that applies to such pre-training and ensures rapid convergence for recall and learning. Using bipolar coding of inputs further speeds convergence compared with binary coding.

1.1. Backpropagation invariance and the EM connection

The proof that BP is generalized EM casts BP as maximum likelihood estimation. It then shows that the iterative BP algorithm has the same gradient update at iteration n as does the generalized EM algorithm in the master equation of (94):

$$\nabla_{\Theta} \ln p(\mathbf{y}|\mathbf{x}, \Theta^n) = \nabla_{\Theta} Q(\Theta^n|\Theta^n) \tag{1}$$

as we explain below. This gradient identity applies far beyond neural networks. We show that it follows from the concavity of the logarithm and the related fact that Shannon entropy minimizes cross entropy.

The left side of (1) implies that a pass through a neural classifier with 1-in-K encoding corresponds to rolling a K-sided die. This holds because the likelihood of the output layer is a simple type of multinomial distribution. It is a vector normal in the case of a regression network. Then both output layers give back the same BP learning law. We call this *BP invariance*: The parameter gradient of a layer's log-likelihood must equal that layer's BP learning law for a given network configuration. We also show how this applies to layers of logistic or other hidden neurons. This allows EM-based noise boosting of hidden neurons because of the layer-likelihood factorization in (133).

The right side of (1) shows that increasing the network's log-likelihood $\ln p(\mathbf{y}|\mathbf{x}, \Theta^n)$ increases EM's surrogate likelihood function Q. This means that a BP learning iteration takes a step up the network's likelihood or log-likelihood surface. We explain below how the simple "EM trick" leads to the EM algorithm itself. The trick in (68) swaps the left side of the definition of conditional probability $P(B|A) = \frac{P(A \cap B)}{P(A)}$ with the denominator. This swap gives the arbitrary and unconditional probability $P(A)$ as the ratio $P(A) = \frac{P(A \cap B)}{P(B|A)}$ for any "hidden" or other measurable event B whatsoever. The entire EM theory unfolds from this representation of the likelihood $P(A)$.

Fig. 1 shows the high-level BP–EM correspondence for a feedforward neural network with hidden layers. The BP–EM correspondence still holds for recurrent BP (Adigun & Kosko, 2017). The correspondence also holds for the new *bidirectional* BP algorithm (Adigun & Kosko, 2016, 2019a) and its application to generative adversarial neural networks trained on CIFAR-10 image data (Adigun & Kosko, 2019b).

BP's forward pass corresponds to EM's expectation step. BP's backward pass corresponds to EM's maximization step. The maximization here is the *partial* maximization of a gradient step. BP's

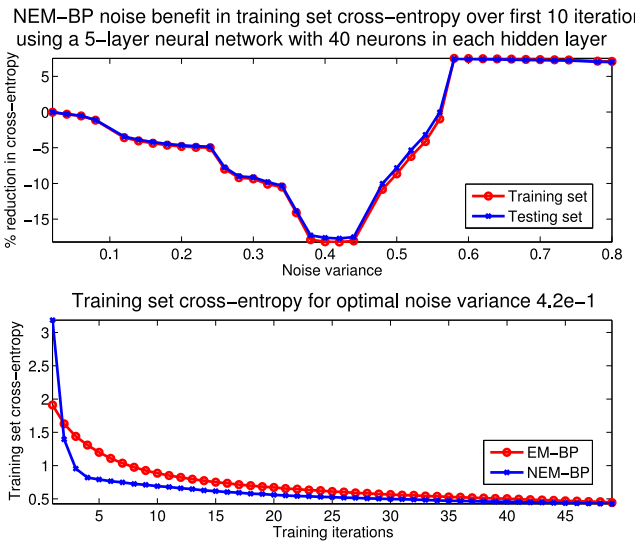

Fig. 2. NEM-noise convergence benefit: NEM noise injection in the 10 output neurons of a multilayer classifier network. The top figure shows the percent median reduction in per-iteration cross entropy for NEM-backpropagation (NEM–BP) training compared with noiseless BP training of a 10-class classification neural network trained on 1000 digit images from the MNIST data set. NEM noise reduced the cross entropy by 18% for the training set and the test set at the optimal noise standard deviation of 0.42. The neural network used three logistic (sigmoidal) hidden layers with 40 neurons each. The input layer used 784 logistic neurons. The output layer used 10 neurons with softmax activations. The bottom figure shows the training-set cross entropy as iterations proceeded for noiseless BP and for NEM–BP training that used the optimal noise variance of 0.42. The knee-point of the NEM–BP curve at iteration 4 achieved the same cross entropy as noiseless BP did at iteration 15.

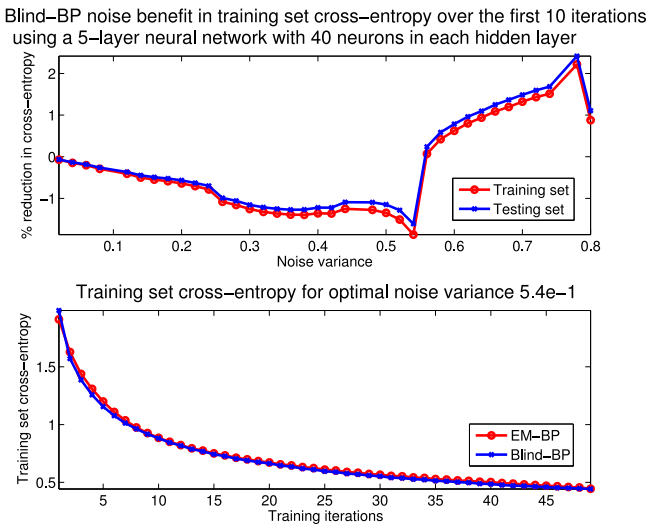

Fig. 3. Minimal benefits from blind-noise injection. The top figure shows the percent median reduction in per-iteration cross entropy for EM–BP training with blind noise (Blind-BP) relative to noiseless BP training of a 10-class classification neural network that trained on 1000 images from the MNIST data set. Blind noise produced only a small reduction in cross entropy of 1.7% for the training and the test set at the optimal noise standard deviation of 0.54. The neural network used three logistic (sigmoidal) hidden layers with 40 neurons each. The input layer used 784 logistic neurons. The output layer used 10 neurons with softmax activation functions. The bottom figure shows the training-set cross entropy as iterations proceeded for noiseless BP and blind-BP training that used the optimal noise variance of 0.54. Both blind-noise BP and noiseless BP gave similar cross entropies for all iterations.

hidden neurons and other hidden parameters correspond to EM's latent variables. The proof of Theorem 1 in the appendix gives the formal details of the correspondence. It shows that the BP and generalized EM gradients have the same parameter learning or update equations. Figs. 8 and 9 show the geometry of the noise-benefit sufficient condition for the special cases of cross-entropy and either logistic or Gaussian output neurons.

1.2. Noise boosting BP via the noisy EM theorem

The gradient identity (1) and BP invariance allows EM-based noise boosting of BP by invoking the recent noisy EM (NEM) theorem (Osoba & Kosko, 2013, 2016b; Osoba, Mitaim, & Kosko, 2011b, 2013). This theorem gives a sufficient condition for speeding the average convergence of the EM algorithm so long as the noise obeys the likelihood-ratio positivity condition in (114). NEM noise depends only on the gradient connection in (1). It does not depend on second-order Hessian information as in the Adam (adaptive moment estimation) variable-rate optimizer (Kingma & Ba, 2014).

We state the NEM Theorem as Theorem 2 for completeness. The NEM Theorem ensures on average that at each iteration a proper noise injection results in a larger step up the hill of probability than does a noiseless EM step. So the NEM Theorem ensures that proper noise injection speeds the average convergence of the BP algorithm because of Theorem 1. The layer-likelihood factorization (133) and the proof of Theorem 5 show how to inject NEM noise in hidden layers.

The NEM noise benefit counts as a type of "*stochastic resonance*" effect: a small amount of noise improves the performance of a nonlinear system while too much noise harms the system (Bulsara, Boss, & Jacobs, 1989; Franzke & Kosko, 2011; Gammaitoni, Hänggi, Jung, & Marchesoni, 1998; Kosko, 2006; McDonnell, Stocks, Pearce, & Abbott, 2008; Mitaim & Kosko, 1998, 2014; Patel & Kosko, 2008, 2009, 2010, 2011; Wilde & Kosko, 2009).

The NEM noise benefit differs from ordinary stochastic resonance in two ways. The first way is that the NEM noise benefit does not rely on a neuron's threshold. The NEM regression result in (182) applies even to identity neurons in output or hidden layers. The second way is that stochastic-resonance noise is blind or dither noise in general. NEM noise is instead just that noise n that makes the current signal y more probable:

$$p(y+n|\Theta) \geqslant p(y|\Theta) \tag{2}$$

for some parameter vector Θ. Then taking the average of the resulting log-likelihood-ratio inequality $\ln \frac{p(y+n|\Theta)}{p(y|\Theta)} \geqslant 0$ gives the sufficient positivity condition in (114) for a NEM noise benefit.

The NEM Theorem ensures only that NEM noise will improve the average convergence at each iteration. It does not describe the magnitude of the speed-up. Our simulations on MNIST handwritten-digit data show that the speed-up can be substantial when injecting noise in just the output softmax neurons alone. Fig. 2 shows the noise benefit for cross-entropy training of a feedforward neural network. The NEM version shows an 18% median decrease in cross entropy per iteration compared with noiseless backpropagation training. We also show how NEM noise injection into the hidden neurons further speeds convergence and accuracy.

NEM noise is *not* blind noise. Fig. 3 shows that adding blind noise gives only a minuscule improvement of 1.7% in cross entropy over the noiseless EM–BP algorithm. Reed used a Taylor-series expansion to argue that these slight boosts from adding small-amplitude blind noise resemble Tikhonov regularization (Reed, Marks, & Oh, 1995; Reed, Oh, & Marks, 1992). Bishop published a similar result (Bishop, 1995). We do expect that the NEM noise

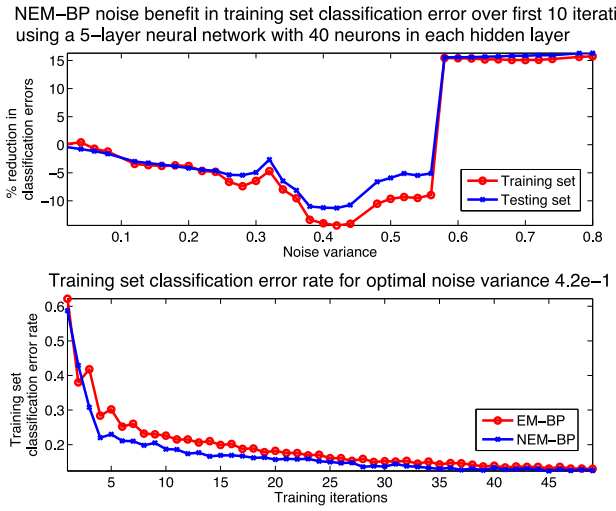

Fig. 4. NEM noise improved classification accuracy. The figure shows the percent median reduction in the per-iteration classification error rate for the NEM-backpropagation (NEM–BP) training compared with noiseless BP training. The neural network was a 10-class classification network trained on 1000 images from the MNIST data set. NEM noise injection reduced the classification error rate by 15% for the training set and about 10% for the test set at the optimal noise standard deviation of 0.42. The neural network used three hidden layers with 40 logistic (sigmoidal) neurons each. The input layer used 784 logistic neurons. The output layer used 10 neurons with softmax activations. The bottom figure shows the training-set classification error rate as iterations proceeded for noiseless BP and NEM–BP training that used the optimal noise variance of 0.42. The knee-point of the NEM–BP curve at iteration 4 had the same classification error rate as noiseless BP did at iteration 11.

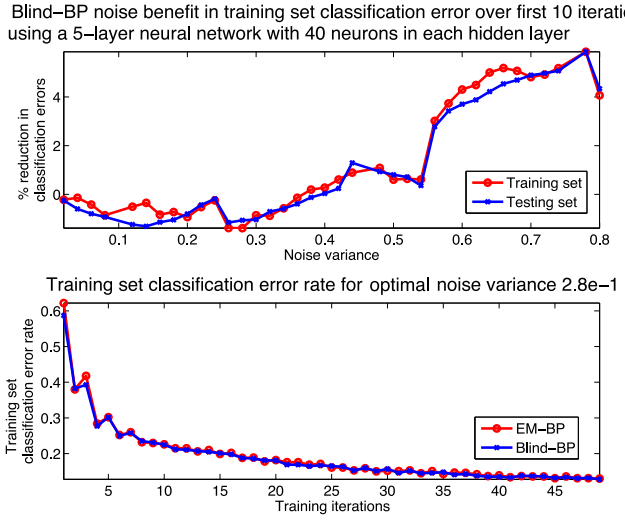

Fig. 5. Minimal accuracy benefits for blind noise. Percent median reduction in per-iteration classification error rate for EM-backpropagation training with blind noise (Blind-BP) compared with the noiseless EM–BP training of a 10-class classification neural network trained on 1000 images from the MNIST data set. Optimal noise (with standard deviation 0.28) gave only a minor reduction in classification error rate of 1% for the training and the test set. The classifier network used three logistic hidden layers with 40 neurons each. The input layer used 784 logistic neurons and the output layer used 10 softmax neurons. The bottom figure shows the training-set classification error rate over iterations for EM–BP and blind-BP training that used the optimal noise variance of 0.28. Both curves show similar classification error rates for all iterations.

benefit will fall off as the sample size grows because NEM noise tends to act as synthetic random sample data (Osoba et al., 2013).

NEM–BP noise can add to all the neurons or other parameters in the network. It can add to both the output and hidden neurons. It can multiply any signal or parameter. Theorems 3 and 4 prove the NEM noise benefit for adding noise to the output neurons of a respective softmax classifier or regressor. Section 6 shows that a NEM noise benefit also applies to the hidden neurons. Fig. 10 shows the effects of NEM-noise versus no noise injection in the hidden layers of a classifier network and a regression network. NEM noise gave a 60.44% relative reduction in the per-iteration training-set cross-entropy compared with standard noiseless BP. It gave a 54.39% relative reduction in the per-iteration test-set cross-entropy.

NEM–BP also tends to give better classification accuracy at each training iteration than the noiseless EM–BP algorithm. This occurs both because NEM noise improves the cross entropy on average at each iteration and because cross entropy approximates the classification error rate. Theorem 6 recasts this explanation in terms of likelihood: The network likelihood gives a lower bound on the classification accuracy. NEM noise boosts just this likelihood. Fig. 4 shows that NEM–BP gives a 15% median improvement in the per-iteration classification error rate for the training set. It gives a 10% improvement for the testing set at the optimal noise variance of 0.42. Fig. 5 shows that this noise benefit disappears if we inject blind noise in place of NEM noise.

A related NEM result holds for the feedback pre-training of the individual layers of neurons in a multilayer neural network. These so-called restricted Boltzmann machine (RBM) (Hinton et al., 2012; Hinton, Osindero, & Teh, 2006; Smolensky, 1986) layers are simple bidirectional associative memories (BAMs) (Kosko, 1987, 1988, 1991) that undergo synchronous updating of the neurons. They are BAMs because the neurons in contiguous layers use the same connection matrix $\mathbf{W}$ in the forward pass that they use in transposed form $\mathbf{W}^T$ in the backward pass. The neurons have no within-layer connections but can in more general BAM topologies.

The general BAM convergence theorem (Kosko, 1987, 1988, 1991) guarantees that all such rectangular matrices $\mathbf{W}$ are globally bidirectionally stable for either synchronous or asynchronous neuron updates. This theorem holds for general neuronal activation nonlinearities because the RBM energy function is a Lyapunov function for the BAM network. The theorem ensures 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. We present a special discrete case of the BAM convergence theorem. It holds for a discrete version of the adaptive BAM theorem (Kosko, 1987, 1988, 1991) for simple Hebbian correlation learning. These results help explain the observed rapid convergence in stacked RBM layers. They do not invoke Markov-chain convergence or other stochastic asymptotic properties.

Fig. 6 shows the noise benefit for NEM training of a logistic–logistic BAM with 784 visible and 40 hidden neurons. All the neurons in both fields have logistic sigmoidal activations. A "swamping" result still achieves this rapid BAM convergence even if a hidden layer uses Gaussian activations. Adding enough logistic neurons to the contiguous layer can always swamp or overcome any convergence problems that the non-sigmoidal Gaussian neurons might otherwise produce. Fig. 11 further shows how *bipolar* coding of the MNIST images rapidly speeds up BAM convergence compared with binary coding. This result follows from a correlation-coding theorem in the Appendix of the original BAM paper (Kosko, 1988). It simply requires that the input neurons encode data using the bipolar interval $[-1, 1]$ rather than the binary interval $[0, 1]$. The two theorems in the last section show that training these contiguous BAM layers with contrastive divergence is a form of generalized EM. So the training benefits from NEM noise. These noise benefits include the case where one layer has logistic neurons and the next layer has Gaussian neurons.

NEM training also gave about a 16% improvement in the per-iteration squared reconstruction error over noiseless training. Fig. 7 shows that BAM training with blind noise gave no significant benefit.

The NEM Theorem defines a type of "forbidden" condition that ensures a noise speed-up so long as the noise lies outside of a specified region in the noise state–space. The adjective "forbidden" comes from the noise-benefit theorems that describe adding blind white noise to a threshold system so as to increase the system's mutual-information bit count or its cross-correlation or to reduce its probability of detection error (Mitaim & Kosko, 2014; Osoba & Kosko, 2013; Osoba et al., 2011b, 2013; Patel & Kosko, 2008, 2009). The simplest forbidden-interval theorem states that a threshold signal system with bipolar and *sub-threshold* signal amplitudes $-A < A < \theta$ will have a mutual-information (Kosko & Mitaim, 2003, 2004; Mitaim & Kosko, 2004) or cross-correlation noise benefit (Kosko, Lee, Mitaim, Patel, & Wilde, 2009; Lee, Liu, Zhou, & Kosko, 2006; Mitaim & Kosko, 2014) if and only if the average noise $E[N]$ does *not* lie in the parameter interval $(\theta - A, \theta + A)$ for scalar threshold θ. This result applies to a threshold system $Y_t = \text{signum}(S_t + N - \theta)$ for input Bernoulli signal S_t with amplitude A. Forbidden-interval theorems extend to far more general nonlinear neural models and more general noise processes that include Levy jump processes (Patel & Kosko, 2009). Stochastic-resonance noise may well benefit the use of trained neural networks given their nonlinear structure. This paper focuses only on noise that benefits neural training.

NEM forbidden regions are more complicated subsets of noise space. Figs. 8 and 9 show that the noise must lie outside or inside such regions to speed convergence. BP invariance and the likelihood structure of a neural layer control the geometry of the forbidden region. So a layer's neural activations controls its likelihood structure. Theorem 3 describes how the logistic output neurons in Fig. 8 give the forbidden region as a hyperplane-based half-space in noise space. A regression network's output identity neurons and BP invariance imply a Gaussian likelihood structure. Theorem 4 describes the resulting spherical region in noise space for a NEM-noise benefit. Reversing the inequalities of the NEM Theorem gives a dual noise-harm condition for such forbidden regions: Noise drawn from within such regions can only slow the average convergence of the EM and BP algorithms.

Theorems 9 and 10 give similar noise-benefit sufficient conditions in feedback networks for respective Bernoulli–Bernoulli (logistic–logistic) BAMs and Gaussian–Bernoulli (Gaussian-logistic) BAMs.

1.3. Earlier noise injection in backpropagation

The reduction of BP to EM differs in kind from earlier efforts that applied EM to BP or that used BP in EM (Cook & Robinson, 1995; Ng & McLachlan, 2004). These earlier efforts treated EM and BP as different algorithms. They did not show or suggest that one subsumed the other. Nor did they inject specially chosen noise to speed BP training or improve its accuracy.

Adding *blind* or unconditional noise to learning algorithms has a long history in neural networks and machine learning. Minsky observed in his 1961 overview of artificial intelligence that "one may use noise added to each variable" in state–space search based on random hill climbing (Minsky, 1961). Widrow showed in 1976 that adding blind noise to the gradient parameters of the LMS algorithm can improve convergence (Widrow & McCool, 1976). LMS applies to a minimal linear network with no hidden neurons.

The NEM approach does not add blind noise to a network. It adds specially chosen NEM noise to the data or the network neurons or related parameters. Amari analyzed a "stochastic perceptron" regression network (Amari, 1995) in the context of EM.

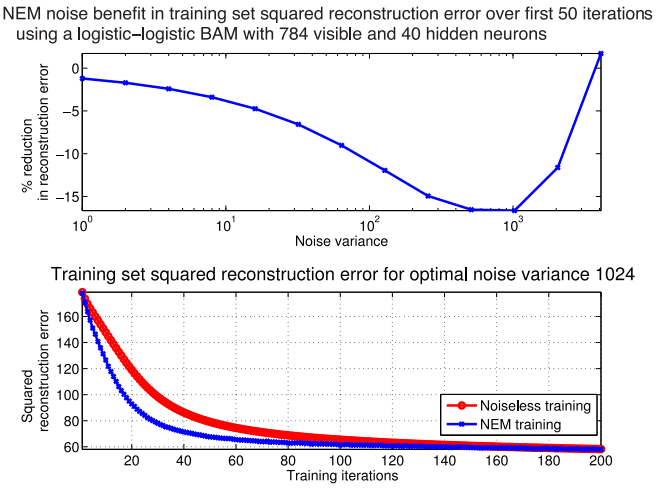

Fig. 6. NEM benefits in BAM training. The figures show the percent median reduction in per-iteration squared reconstruction error for training with NEM noise compared with the noiseless training of a 2-layer bidirectional associative memory (BAM) on 1000 images from the MNIST data set. NEM noise gave a 16% reduction in the training-set squared reconstruction error at the optimal noise variance of 1024. The BAM used one hidden layer with 40 logistic neurons and an input layer with 784 logistic neurons. The bottom figure shows the training-set squared reconstruction error over iterations for NEM and noiseless training that used the optimal noise variance of 1024.

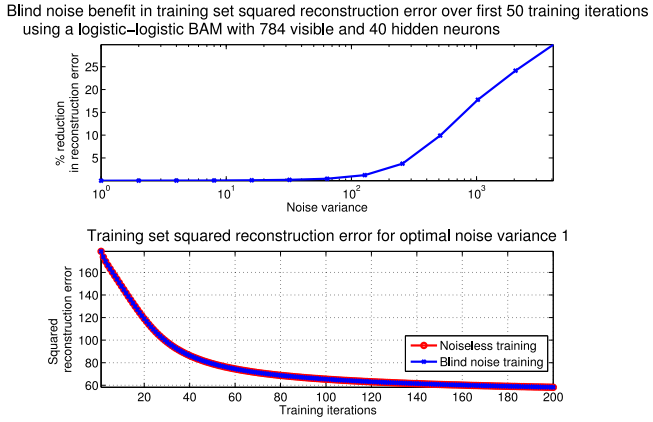

Fig. 7. No blind-noise benefit in BAM training. The figures show the percent median reduction in per-iteration squared reconstruction error for training with blind noise compared with the noiseless training of a 2-layer BAM on 1000 images from the MNIST data set. The per-iteration squared reconstruction error did not differ significantly for the two cases. The BAM used one hidden layer with 40 logistic neurons and an input layer with 784 logistic neurons.

He came close to finding the BP-as-EM result in Theorem 1. But Amari used a variance-based squared error for minimization rather than the unweighted squared error that ties BP regression to EM. NEM works with noise that has positive and often large variance or dispersion.

More recent noise-injection efforts have found an approximate regularizing effect from adding faint blind white noise to BP (An, 1996; Bishop, 1995; Hayakawa, Marumoto, & Sawada, 1995; Matsuoka, 1992; Reed et al., 1995, 1992). The drop-out neural algorithm similarly applies blind *multiplicative* Bernoulli or Gaussian noise to hidden activations (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Denoising autoencoders likewise randomly zeros out input values in autoencoder networks to reduce reconstruction error (Vincent, Larochelle, Lajoie, Bengio, & Manzagol, 2010). Holmstrom and Koistinen (1992) earlier showed that injecting additive Gaussian noise in mean-square BP can

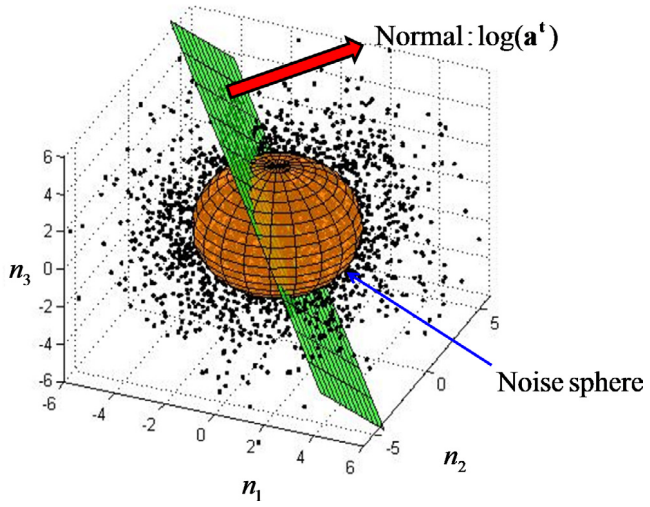

Fig. 8. Noise-benefit region for a multilayer neural network with logistic output neurons: NEM noise speeds the maximum-likelihood parameter estimation of the neural network if the injected noise lies above the NEM hyperplane in noise space. Theorem 3 defines the hyperplane in this case. The likelihood structure of this logistic layer was a product of Bernoulli densities. The activation signal $\mathbf{a}^t$ of the output layer controlled the normal to the hyperplane. The hyperplane changed as learning proceeded because the parameters and hidden-layer activations changed. The independent and identically distributed Gaussian noise had mean 0 and variance 3. The vector (3, 1, 1) was the normal to the hyperplane.

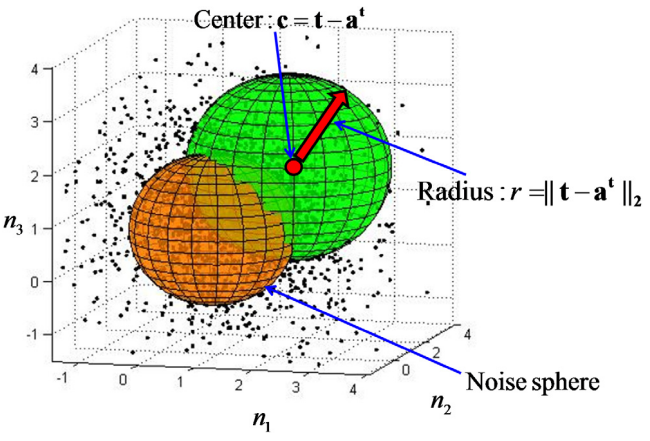

Fig. 9. Noise-benefit region for a regression network with linear or identity output neurons: NEM noise speeds the maximum-likelihood parameter estimation of the neural network if the noise lies inside a hypersphere in accord with Theorem 4. The likelihood structure of the output layer is a vector normal density. The activation signal $\mathbf{a}^t$ of the output layer and the target signal $\mathbf{t}$ controlled the center and radius of the hypersphere. This hypersphere changed as learning proceeded because the parameters and hidden-layer activations changed. The independent and identically distributed Gaussian noise had mean 0 and variance 3. It had center $\mathbf{t} - \mathbf{a}^t = (1, 1, 1)$.

improve the network's generalization ability because such noise acts as a Parzen-window estimate of the data density. The authors did not prove a sufficient condition for this noise benefit. Azamimi, Uwate, and Nishio (2008) later found through simulations that adding tent-map chaotic noise to mean-square BP improved its convergence. The injected chaotic noise outperformed blind random noise.

We stress again that injecting such blind noise differs from injecting NEM noise. The geometry of the main NEM noise result also shows that blindly picking noise from both above and below the NEM hyperplane should not on average produce a noise benefit. This holds because on average noise from above the NEM hyperplane improves convergence or accuracy while noise from below it only degrades performance on average. We also show below that all the main noise-boost theorems still hold for any additive regularizer if the noise does not appear in the regularizer term itself.

The NEM noise-injection results also differ from "noise contrastive estimation" (Gutmann & Hyvärinen, 2012; Mnih & Kavukcuoglu, 2013) This perturbation technique uses a type of Monte Carlo randomization to simplify the computation of a normalization or partition function in logistic regression. It does *not* inject noise into the data. Nor does it work with BP-based deep learning on multi-neuron networks. It instead compares training with data to training with blind noise. So the NEM noise boost could in principle apply to its data training. Noise contrastive estimation also randomly picks subsets of data for processing. The BAM convergence theorem below does allow random selection of neurons for updating. But that asynchronous updating does not involve the NEM noise-injection process.

1.4. Overview of subsequent sections

The next section casts the BP algorithm as maximum likelihood estimation. This maximum-likelihood framework includes classification and regression networks as well as logistic networks. Section 3 presents the EM algorithm for neural-network training and proves that it reduces to the backpropagation algorithm per the gradient equality in (1). The proof shows that BP's gradient updates at each iteration are the same as the gradient updates of generalized EM. Monte Carlo importance sampling simplifies some of the gradient computations. Section 4 reviews the NEM theorem that states a sufficient condition for noise-boosting the EM algorithm and its progeny.

Section 5 derives noise-benefit sufficient conditions for a feedforward neural network. It shows how to inject NEM noise into the output neurons of a classifier or regression network. The method applies to any network so long as the choice of neurons and network likelihood leaves the BP laws invariant. Section 6 further shows how to inject NEM noise into hidden neurons. Section 7 shows how NEM noise-boosting the network likelihood can improve the classification accuracy of classifier networks. The accuracy bound also applies to networks whose output neurons are logistic neurons.

Section 8 reviews RBMs or BAMs and extends an important version of the BAM global stability theorem for discrete networks. The BAM network converges exponentially quickly to a bidirectional fixed point if the neurons at both layers are logistic. This result extends at once with a "swamping" argument. We can always add more logistic neurons to a hidden layer to ensure rapid BAM convergence even when the contiguous layer consists of Gaussian neurons or other neurons with non-monotonic but bounded activations. We further extend this BAM convergence to include Hebbian correlation learning and use it to explain convergence in the contrastive-divergence setting. A related BAM result shows that using bipolar neuron coding tends to improve performance. This means that the neuron activations should have the range $[-1, 1]$ rather than $[0, 1]$. Fig. 11 shows that bipolar encoding speeds up BAM convergence by more than an order of magnitude compared with binary encoding.

The penultimate section shows that contrastive-divergence learning law is also a special case of generalized EM. It then derives sufficient conditions for a NEM noise benefit for maximum-likelihood training of Bernoulli–Bernoulli and Gaussian–Bernoulli BAMs or RBMs. Section 10 presents the related simulation results.

2. Backpropagation as maximum likelihood estimation

This section shows that the BP algorithm performs maximum-likelihood (ML) estimation of a neural network's parameters. The next section shows that BP is just one form of the EM algorithm for ML estimation. Then the section after that shows how to noise-boost EM and thus noise-boost BP.

We use a 3-layer neural network for notational convenience. All results extend to deep networks with any number of hidden layers. Most of the simulations in the figures used five-layer networks with three hidden layers of logistic neurons.

The network consists of I input neurons, J hidden neurons, and K output neurons. The $I \times J$ weight matrix $\mathbf{W}$ connects the I input neurons to the J hidden neurons. The $J \times K$ matrix $\mathbf{U}$ connects the hidden neurons to the K output neurons. Let the I-vector $\mathbf{x}$ denote the I input neuron values. The input neurons may just act as data registers and thus have identity activations: $a_j^i(x_j) = x_j$ for the jth identity activation in the input i layer. We allow them to have nonlinear activations and use logistic input neurons in the simulations.

The J hidden units can have arbitrary nonlinear activations. They often in practice have sigmoidal or monotone nondecreasing activations. They can also have non-sigmoidal or Gaussian activations as in radial-basis networks and the more general fuzzy function approximators (Jang & Sun, 1993; Kosko, 1994, 1996; Osoba, Mitaim, & Kosko, 2011a) and their representations as generalized probability mixtures (Kosko, 2018). They can also have quasi-linear rectilinear-unit or "ReLU" activations as we discuss below. Different hidden layers can contain both sigmoidal and non-sigmoidal or Gaussian hidden neurons. The penultimate section explores this for pre-training of deep networks.

The most common sigmoidal activation remains the logistic activation. Let $\mathbf{a}^h$ denote the vector of hidden-neuron activations. Then the jth hidden neuron is (binary) logistic if

$$a_j^h(o_j^h) = \frac{1}{1 + \exp(-o_j^h)} \tag{3}$$

$$= \frac{1}{1 + \exp\left(-\sum_{i=1}^{I} w_{ji} x_i\right)} \tag{4}$$

if w_{ji} is the weight of the directed link or edge or synapse that connects the ith visible neuron to the jth hidden neuron. The term o_j^h denotes the hidden neuron's inner-product input:

$$o_j^h = \sum_{i=1}^{I} w_{ji} x_i \ . \tag{5}$$

The jth input neuron can also have a nonlinear activation $a_j^i(x_j)$. Then (3) implies that the partial derivative of a_j^h with respect to its input o_j^h has a simple nonnegative form:

$$\frac{\partial a_j^h}{\partial o_j^h} = a_j^h(1 - a_j^h) \ . \tag{6}$$

Large inputs can quickly saturate a steep logistic. Then the product term in (6) implies that logistic hidden units can lead to vanishing gradients in deep networks. This explains the increasing use of rectified-linear-unit or ReLu activations $a_j^h(o_j^h) = \max(0, o_j^h)$: ReLu activations are also monotone nondecreasing but do not saturate for large inputs.

The sigmoidal hidden activations sometimes have a related hyperbolic-tangent form:

$$a_j^h(o_j^h) = \frac{e^{o_j^h} - e^{-o_j^h}}{e^{o_j^h} + e^{-o_j^h}} \ . \tag{7}$$

The hyperbolic tangent is just a scaled bipolar version of the logistic activation l_j in (3): $a_j^h(o_j^h) = 2l_j(2o_j^h) - 1$. This also leads to a simple and nonnegative derivative:

$$\frac{\partial a_j^h}{\partial o_j^h} = 1 - (a_j^h)^2 \ . \tag{8}$$

The K output neurons can have arbitrary activations so long as they leave the BP learning laws in (1) invariant. Classification networks usually have output neurons with Gibbs or softmax activations so that the output vector defines a discrete probability distribution. This ratio of exponentials follows from rewriting the Bayes-theorem ratio in terms of exponentials for K-class classification (Bishop, 2006). Regression networks often use output neurons with linear or logistic activations. We show below that BP invariance requires that output neurons with identity (or linear) activations need Gaussian target vectors to preserve the BP update equations. We address each in turn to derive the invariant BP laws as maximum likelihood.

Consider first a multilayer *classification* network. Let $\mathbf{y}$ denote the K-valued target or output variable. Let $\mathbf{t}$ denote its 1-in-K binary encoding. So the target vector $\mathbf{t}$ is a unit binary vector and thus a simple probability distribution. Then t_k is the kth output neuron's value with softmax or Gibbs activation

$$a_k^t = \frac{\exp(o_k)}{\sum_{l=1}^{K} \exp(o_l)} \tag{9}$$

$$= \frac{\exp\left(\sum_{j=1}^{J} u_{kj} a_j^h\right)}{\sum_{l=1}^{K} \exp\left(\sum_{j=1}^{J} u_{lj} a_j^h\right)} \tag{10}$$

$$= p_k(y = t_k | \mathbf{x}, \Theta) \tag{11}$$

where u_{kj} is the weight of the directed link that connects the jth hidden to the kth target neuron and where o_k denotes the output neuron's inner-product input:

$$o_k = \sum_{j=1}^{J} u_{kj} a_j^h \ . \tag{12}$$

So a_k^t depends on the input $\mathbf{x}$ and on the parameter matrices $\mathbf{U}$ and $\mathbf{W}$. The vector Θ denotes all network parameters.

The total output vector $\mathbf{a}^t$ defines a discrete probability density

$$\mathbf{a}^t = p(\mathbf{y} = \mathbf{t} | \mathbf{x}, \Theta) \tag{13}$$

because of the exponential-sum normalizer or "partition function" in the denominator of (10). Then taking the logarithm gives the neural network's log-likelihood function $L(\Theta)$:

$$L(\Theta) = \ln p(\mathbf{t} | \mathbf{x}, \Theta) \ . \tag{14}$$

The maximum-likelihood parameters Θ^* for the neural network solve the optimization problem

$$\Theta^* = \arg\max_{\Theta} \ln p(\mathbf{y} | \mathbf{x}, \Theta) \ . \tag{15}$$

The basic NEM Theorem shows that the NEM-noise boosted parameter vector $\Theta_{NEM}^{(n)}$ converges in fewer steps to the maximum-likelihood network parameter vector Θ^* than does the noiseless parameter vector $\Theta^{(n)}$. Each noise-boosted step up the likelihood surface is at least as large on average as is the noiseless step.

The partition function in the softmax activation (10) leads to a more complicated partial derivative with respect to the K inner products $o_1, \ldots, o_K$:

$$\frac{\partial a_k^t}{\partial o_j} = \begin{cases} -a_j^t a_k^t & \text{if } k \neq j \\ a_k^t(1 - a_k^t) & \text{if } k = j \ . \end{cases} \tag{16}$$

The cross entropy $E(\Theta)$ is the usual scalar performance measure for a classifier network (Bishop, 2006) . The cross entropy compares the target pdf $\mathbf{t}$ with the output softmax pdf $\mathbf{a}^t$ as the expected information $E_t[\ln \frac{1}{a^t}]$:

$$E(\Theta) = -\sum_{k=1}^{K} t_k \ln a_k^t \ . \tag{17}$$

We first show that the network log-likelihood $L(\Theta)$ equals the negative cross entropy: $L(\Theta) = -E(\Theta)$. This equality follows by rewriting the target-weighted sum of logarithms (17) as the product of logarithms:

$$E(\Theta) = -\sum_{k=1}^{K} \ln(a_k^t)^{t_k} \tag{18}$$

$$= -\ln\Big[\prod_{k=1}^{K}(a_k^t)^{t_k}\Big] \tag{19}$$

$$= -\ln\Big[\prod_{k=1}^{K} p_k(y = t_k|\mathbf{x}, \Theta)\Big] \tag{20}$$

$$= -\ln p(\mathbf{y}|\mathbf{x}, \Theta) \tag{21}$$

$$= -L(\Theta) \ . \tag{22}$$

The probability density factorization (20) holds because we assume that the K output neurons are conditionally independent of one another given the input $\mathbf{x}$. Such statistical independence also reflects the network structure that there are no synaptic connections among the output neurons. The output layer's intra-layer connection matrix is a null matrix.

So $p(\mathbf{y}|\mathbf{x}, \Theta) = \exp(-E(\Theta))$: Minimizing the cross entropy $E(\Theta)$ maximizes the log-likelihood L and conversely. So such cross-entropy estimators enjoy the same statistical properties that ML estimators do. They are consistent and asymptotically normal in general. They also obey the invariance principle: $\widehat{g(\Theta)}^{ML} = g(\hat{\Theta}^{ML})$ for an arbitrary function g (Hogg, McKean, & Craig, 2013).

The same derivation shows that $-E(\Theta) = \ln a_k^t$ if k is the correct target label for input pattern $\mathbf{x}$: $\mathbf{x} \in C_k$ for input decision or pattern class C_k when the K classes C_j partition the input pattern space. This holds because the target vector $\mathbf{t}$ is the unit bit vector with a 1 in the kth slot and 0s elsewhere. But the above derivation that $p(\mathbf{y}|\mathbf{x}, \Theta) = \exp(-E(\Theta))$ still holds if the target values $t_1, \ldots, t_K$ are not binary but instead define an arbitrary discrete probability distribution. That also holds for the gradient derivations below.

We show next that minimizing the cross entropy minimizes the discrete Kullback–Leibler divergence $KL(\mathbf{t}\|\mathbf{a}^t)$ between the target vector $\mathbf{t}$ and the vector $\mathbf{a}^t$ of output activations. This equivalence holds because both $\mathbf{t}$ and $\mathbf{a}^t$ are discrete pdfs:

$$KL(\mathbf{t}\|\mathbf{a}^t) = \sum_{k=1}^{K} t_k \ln \frac{t_k}{a_k^t} \tag{23}$$

$$= \sum_{k=1}^{K} t_k \ln t_k - \sum_{k=1}^{K} t_k \ln a_k^t \tag{24}$$

$$= -H(\mathbf{y}) + E(\Theta) \tag{25}$$

from (17). The output entropy $H(\mathbf{y})$ does not affect the minimization because $H(\mathbf{y})$ does not depend on θ. So minimizing the Kullback–Leibler divergence $KL(\mathbf{t}\|\mathbf{a}^t)$ also maximizes the network log-likelihood $L(\Theta)$.

BP updates a classifier network's parameters Θ through gradient descent to minimize the cross entropy $E(\Theta)$. The above arguments show that this gradient descent also minimizes the Kullback–Leibler divergence. It also maximizes the log-likelihood $\ln p(\mathbf{y}|\mathbf{x}, \Theta)$ and thus maximizes $L(\Theta)$. So we can write the estimate of Θ at the $(n+1)$th iteration or training epoch as $\Theta^{(n+1)}$ in three equivalent ways:

$$\Theta^{(n+1)} = \Theta^{(n)} - \eta_n \nabla_\Theta E(\Theta)\Big|_{\Theta=\Theta^{(n)}} \tag{26}$$

$$= \Theta^{(n)} + \eta_n \nabla_\Theta \ln p(y|\mathbf{x}, \Theta)\Big|_{\Theta=\Theta^{(n)}} \tag{27}$$

$$= \Theta^{(n)} + \eta_n \nabla_\Theta L(\Theta)\Big|_{\Theta=\Theta^{(n)}} \tag{28}$$

where η_n is a positive learning rate or a sequence of (usually decreasing) learning rates.

We next derive the two key partial derivatives of the network log-likelihood $L(\Theta)$ that underlie BP's gradient descent or ascent for a classifier neural network. The argument below shows that the same partial derivatives result for regression and logistic. So BP invariance holds. The argument assumes that all functions are sufficiently smooth to apply the chain rule of differential calculus (Kosko, 1991).

The first result is that the partial derivative of the log-likelihood L with respect to the synaptic weight u_{kj} is

$$\frac{\partial L}{\partial u_{kj}} = (t_k - a_k^t)a_j^h \tag{29}$$

where the weights u_{kj} connect the hidden neurons to the output neurons. The second result is that the partial derivative of L with respect to w_{ji} is

$$\frac{\partial L}{\partial w_{ji}} = a_j^h(1 - a_j^h)x_i \sum_{k=1}^{K}(t_k - a_k^t)u_{kj} \ . \tag{30}$$

where the weights w_{ji} connect the input neurons or data registers to the hidden neurons. This second result assumes that the hidden neurons have logistic activations and thus have derivatives of the form (6). Using hidden neurons with hyperbolic-tangent activations gives

$$\frac{\partial L}{\partial w_{ji}} = (1 - (a_j^h))^2 x_i \sum_{k=1}^{K}(t_k - a_k^t)u_{kj} \tag{31}$$

from (8). So (29) and (30) give the partial derivatives that perform gradient ascent on the network log-likelihood L. They constitute the BP gradient algorithm for a standard classifier neural network.

The first partial-derivative result (29) follows from

$$\frac{\partial L}{\partial u_{kj}} = \frac{\partial L}{\partial o_k}\frac{\partial o_k}{\partial u_{kj}} \tag{32}$$

$$= \Big(\sum_{i=1}^{K} \frac{\partial L}{\partial a_i^t}\frac{\partial a_i^t}{\partial o_k}\Big)\frac{\partial o_k}{\partial u_{kj}} \tag{33}$$

$$= \Big(t_k \frac{1}{a_k^t}\frac{\partial a_k^t}{\partial o_k} + \sum_{i\neq k}^{K} t_i \frac{1}{a_i^t}\frac{\partial a_i^t}{\partial o_k}\Big)\frac{\partial o_k}{\partial u_{kj}} \tag{34}$$

$$= \Big(t_k \frac{1}{a_k^t}a_k^t(1 - a_k^t) - \sum_{i\neq k}^{K} t_i \frac{1}{a_i^t}a_i^t a_k^t\Big)\frac{\partial o_k}{\partial u_{kj}} \tag{35}$$

from (16)

$$= \Big(t_k - a_k^t \sum_{i=1}^{K} t_i\Big)\frac{\partial o_k}{\partial u_{kj}} \tag{36}$$

$$= (t_k - a_k^t)\frac{\partial o_k}{\partial u_{kj}} \tag{37}$$

$$= (t_k - a_k^t)a_j^h \tag{38}$$

from (12). The derivation confirms that the target values $t_1, \ldots, t_K$ can be any discrete pdf.

The second partial-derivative result (30) follows for logistic neurons from

$$\frac{\partial L}{\partial w_{ji}} = \frac{\partial L}{\partial a_j^h}\frac{\partial a_j^h}{\partial o_j^h}\frac{\partial o_j^h}{\partial w_{ji}} \tag{39}$$

$$= \Big(\sum_{k=1}^{K}\frac{\partial L}{\partial o_k}\frac{\partial o_k}{\partial a_j^h}\Big)\frac{\partial a_j^h}{\partial o_j^h}\frac{\partial o_j^h}{\partial w_{ji}} \tag{40}$$

$$= \Big(\sum_{k=1}^{K}(t_k - a_k^t)\frac{\partial o_k}{\partial a_j^h}\Big)\frac{\partial a_j^h}{\partial o_j^h}\frac{\partial o_j^h}{\partial w_{ji}} \tag{41}$$

from (32)–(37)

$$= \Big(\sum_{k=1}^{K}(t_k - a_k^t)\frac{\partial o_k}{\partial a_j^h}\Big)a_j^h(1 - a_j^h)\frac{\partial o_j^h}{\partial w_{ji}} \tag{42}$$

from (6) since the hidden units a_j^h are logistic

$$= \Big(\sum_{k=1}^{K}(t_k - a_k^t)u_{kj}\Big)a_j^h(1 - a_j^h)x_i \tag{43}$$

from (12) and since

$$\frac{\partial o_j^h}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}}\Big(\sum_{n=1}^{I}x_n w_{jn}\Big) = x_i \ . \tag{44}$$

The partial derivative (44) shows that the input-layer neurons can have logistic or other nonlinear activations a_n^i without changing the basic form of the gradient learning law. Then the partial derivative $\frac{\partial L}{\partial w_{ji}}$ in (30) becomes the slightly more general gradient term

$$\frac{\partial L}{\partial w_{ji}} = a_j^h(1 - a_j^h)a_i^i(x_i)\sum_{k=1}^{K}(t_k - a_k^t)u_{kj} \ . \tag{45}$$

The hidden activations can also have other forms such as the quasi-linear rectilinear (ReLU) form $\max(0, x)$ and its variants.

We turn next to the BP training of a *regression* neural network. We will show that the BP learning laws remain invariant if we correctly pick the network likelihood function and the structure of its output neurons.

This type of supervised neural network corresponds to the classical case (Haykin, 1998; Kosko, 1991; Rumelhart et al., 1986) of minimizing the network's output squared error *SE* to approximate some sampled function $f : \mathbb{R}^{\mathbb{I}} \to \mathbb{R}^{\mathbb{K}}$. The network uses labeled input–output samples $(\mathbf{x_1}, \mathbf{t_1})$, $(\mathbf{x_2}, \mathbf{t_2})$, ... for training. Then BP minimizes the squared-error function SE

$$\text{SE} = \frac{1}{2}\sum_{k=1}^{K}(t_k - a_k^t)^2 \ . \tag{46}$$

for all such training samples. The simplest assumption is that the training samples are direct samples from the function and involve no randomness.

The more general random framework views the training samples as *realizations* or footprints of random vectors $\mathbf{x}$ and $\mathbf{t}$. Then the functional assumption is that some joint or conditional probability density $p(\mathbf{t}|\mathbf{x})$ connects the input and output random vectors and thus that we ultimately sample from a joint density $p(\mathbf{x}, \mathbf{t})$. So the network can use the paired realizations to estimate the governing but unknown density $p(\mathbf{t}|\mathbf{x})$. The argument below assumes this more general random framework.

Function approximation of $f : \mathbb{R}^{\mathbb{I}} \to \mathbb{R}^{\mathbb{K}}$ requires that the K output neurons model any real number. So a linear or identity activation function replaces the Gibbs softmax function at the output layer for regression:

$$a_k^t = o_k = \sum_{j=1}^{J}u_{kj}a_j^h \ . \tag{47}$$

The hidden units still have nonlinear activations. It is just this hidden-layer nonlinearity that allows a multilayer feedforward network with enough hidden units to uniformly approximate any continuous function on a compact set (Barron, 1993; Cybenko, 1989; Hornik, Stinchcombe, & White, 1989).

The random view of neural regression makes a further assumption when minimizing the output squared error (46). We assume in this regression squared-error case that the random target vector $\mathbf{t}$ is a Gaussian K-vector (Bishop, 2006) with mean $\mathbf{a}^t$ and with an identity or white covariance matrix $\mathbf{I}$:

$$\mathbf{t} \sim \mathcal{N}(\mathbf{t}|\mathbf{a}^t, \mathbf{I}) = p_{reg}(\mathbf{t}|\mathbf{x}, \Theta) \tag{48}$$

where

$$\mathcal{N}(\mathbf{t}|\mathbf{a}^t, \mathbf{I}) = \frac{1}{(2\pi)^{K/2}}\exp\left\{-\frac{1}{2}\sum_{k=1}^{K}(t_k - a_k^t)^2\right\} \tag{49}$$

because the target covariance matrix is the K-by-K identity matrix $\mathbf{I}$ and thus has a unit determinant.

Then BP maximizes the *regression* log-likelihood function L_{reg}:

$$L_{reg} = \ln p_{reg}(\mathbf{t}|\mathbf{x}, \Theta) \tag{50}$$

$$= \ln \mathcal{N}(\mathbf{t}|\mathbf{a}^t, \mathbf{I}) \tag{51}$$

$$= \ln(2\pi)^{-\frac{K}{2}} - \frac{1}{2}\sum_{k=1}^{K}(t_k - a_k^t)^2 \ . \tag{52}$$

So maximizing the regression log-likelihood L_{reg} just minimizes the network squared error (46) because the additive constant $\ln(2\pi)^{-\frac{K}{2}}$ does not affect the optimization. This log-likelihood approach to neural regression plainly generalizes to richer probabilistic and constraint models.

We can now show that the BP gradient-update equations for regression are the same as those for classification. This shows that both networks obey BP invariance at their output layers:

$$\frac{\partial L_{reg}}{\partial u_{kj}} = \frac{\partial L_{reg}}{\partial a_k^t}\frac{\partial a_k^t}{\partial o_k}\frac{\partial o_k}{\partial u_{kj}} \tag{53}$$

$$= \frac{\partial L_{reg}}{\partial a_k^t}\frac{\partial o_k}{\partial u_{kj}} \tag{54}$$

from (47)

$$= (t_k - a_k^t)\frac{\partial o_k}{\partial u_{kj}} \tag{55}$$

from (52)

$$= (t_k - a_k^t)a_j^h \tag{56}$$

from (12).

So the regression update (56) for the output layer is the same as the classification update (29). So BP invariance holds for these different output layers with very different layer likelihoods. The regression network also has the same update (30) or (45) for the hidden layers because both types of network have the same hidden structure and because both use the same gradient result (56) to initialize the update process for the hidden parameters. So classification and regression networks have identical BP gradient learning laws. This BP invariance is essential for the result in Theorem 1 that BP gradients equal EM gradients. We show below

that such BP invariance must hold at each layer when noise-boosting. The general factorization of the multiplication theorem in (133) permits this layer decomposition and allows an EM structure to hold at each layer.

We show last that a multilayer network with logistic output neurons also has the same BP gradient updates as classification and regression networks. We call these *logistic* networks if the BP learning laws remain invariant. They apply to multi-class classification where the input may properly belong to more than one pattern class.

A logistic network can approximate vector-valued functions $f : \mathbb{R}^{\mathbb{I}} \to [0, 1]^K$ that map real vectors to the unit hypercube $[0, 1]^K$. So a logistic network can act as a regression network. It can also learn or approximate fuzzy-set outputs since the unit hypercube $[0, 1]^K$ is the power set of all finite fuzzy sets of length K (Carpenter, Grossberg, & Rosen, 1991; Kosko, 1991).

We also expect that a logistic network has a relationship to classification networks since the Gibbs or softmax activation (10) reduces to the logistic activation (3) if the network has just one output neuron and thus if $K = 1$. This one-neuron-output case reflects the binary Bayesian classification involved when the system must decide between a hypothesis or class H and its opposite H^c given the input $\mathbf{x}$ as evidence. Then Bayes Theorem gives the posterior $p(H|\mathbf{x})$ as

$$p(H|\mathbf{x}) = \frac{p(H)p(\mathbf{x}|H)}{p(H)p(\mathbf{x}|H) + p(H^c)p(\mathbf{x}|H^c)} \tag{57}$$

$$= \frac{1}{1 + e^{-\phi(\mathbf{x})}} \,. \tag{58}$$

This logistic structure holds if $\phi(\mathbf{x})$ has the log-odds form

$$\phi(\mathbf{x}) = \ln \frac{p(H)p(\mathbf{x}|H)}{p(H^c)p(\mathbf{x}|H^c)} \,. \tag{59}$$

This binary classification suggests in turn how to define the appropriate log-likelihood function L_{log} for a logistic network with K conditionally independent output neurons and target vector $\mathbf{t}$. Define the network likelihood $p_{log}(\mathbf{y}|\mathbf{x}, \Theta)$ as a product of independent Bernoulli densities

$$p_{log}(\mathbf{y}|\mathbf{x}, \Theta) = \prod_{k=1}^{K} (a_k^t)^{t_k} (1 - a_k^t)^{1-t_k} \,. \tag{60}$$

Then the logistic network's log-likelihood L_{log} adds two cross-entropy sums:

$$L_{log} = \ln p_{log}(\mathbf{y}|\mathbf{x}, \Theta) \tag{61}$$

$$= \sum_{k=1}^{K} t_k \ln a_k^t + \sum_{k=1}^{K} (1 - t_k) \ln(1 - a_k^t) \,. \tag{62}$$

Then the BP gradient-update equations for a logistic network are the same as those for classification and regression networks:

$$\frac{\partial L_{log}}{\partial u_{kj}} = \frac{\partial L_{log}}{\partial a_k^t} \frac{\partial a_k^t}{\partial o_k} \frac{\partial o_k}{\partial u_{kj}} \tag{63}$$

$$= \frac{\partial L_{log}}{\partial a_k^t} a_k^t (1 - a_k^t) a_j^h \tag{64}$$

from (6) and (12)

$$= [t_k \frac{1}{a_k^t} - (1 - t_k) \frac{1}{1 - a_k^t}] a_k^t (1 - a_k^t) a_j^h \tag{65}$$

$$= [t_k (1 - a_k^t) - (1 - t_k) a_k^t] a_j^h \tag{66}$$

$$= (t_k - a_k^t) a_j^h \,. \tag{67}$$

So the logistic network's gradient update for the output layer is the same as the classification update (29) and the regression update (56). This confirms BP invariance for the logistic layer log-likelihood. Their hidden structure is also the same.

So the BP learning laws remain invariant for all three networks because they have the same BP gradient partial derivatives. This means that Theorem 1 applies to all three networks and indeed to many more.

3. Backpropagation as generalized expectation maximization

Both BP and the EM algorithm find the ML estimate of a neural network's parameters. So both algorithms climb a local hill of probability or log-likelihood. Both algorithms are iterative algorithms that involve many forward and backward sweeps. Both algorithms also involve hidden or latent parameters. This raises the question whether there is a formal relationship between BP and EM. Theorem 1 declares that there is: Backpropagation is a special case of the generalized EM algorithm. We first develop the EM algorithm.

The EM algorithm is an iterative maximum likelihood method for the general case of missing data or latent variables Z (Dempster et al., 1977; Efron & Hastie, 2016).

The EM algorithm maximizes the log-likelihood $\ln p(y|\mathbf{x}, \Theta)$ by maximizing the lower-bound *surrogate* likelihood or Q-function $Q(\Theta|\Theta^n)$. The expectation or E-step computes the current Q-function $Q(\Theta|\Theta^n)$. The maximization or M-step maximizes $Q(\Theta|\Theta^n)$ over the parameters Θ given the data and given the current parameter estimate Θ^n. This maximization gives the new parameter estimate Θ^{n+1} for the next round of E–M steps.

EM's "ascent property" ensures that increasing $Q(\Theta|\Theta^n)$ can only increase $\ln p(y|\mathbf{x}, \Theta)$ (Dempster et al., 1977). We derive this result below for network parameters and show in the next section how noise can boost the ascent. The updates Θ^{n+1} converge to the local ML maximum Θ^*. The E-step and M-step have an especially simple form for tuning the parameters of a convex mixture of Gaussian pdfs (McLachlan & Krishnan, 2007). A key connection with BP is that EM's "latent" or hidden variables Z correspond to the hidden units h in the multilayer neural network.

The EM algorithm arises from the definition of conditional probability $P(B|A) = \frac{P(A \cap B)}{P(A)}$ for any probability measure P. We assume that all probabilities are positive.

The key EM insight is that we can write *any* marginal probability $P(A)$ in terms of *any* measurable event B:

$$P(A) = \frac{P(A \cap B)}{P(B|A)} \,. \tag{68}$$

Event B can represent missing data or latent or hidden variables or any other quantity. We call this the "EM trick". Taking logarithms in (68) gives the basic EM-like equality

$$\ln P(A) = \ln P(A \cap B) - \ln P(B|A). \tag{69}$$

These probabilities can condition on a set of parameters Θ. This gives the parametrized form for the log-likelihood:

$$\ln P(A|\Theta) = \ln P(A \cap B|\Theta) - \ln P(B|A, \Theta). \tag{70}$$

The next step takes expectations on both sides of (70) with respect to the discrete density $P(B|A, \Theta)$: $P(B|A, \Theta) + P(B^c|A, \Theta) = 1$ for any parameter set Θ. This expectation does not affect the log-likelihood $\ln P(A|\Theta)$ because the log-likelihood does not involve B. The first term on the right of (70) is the "complete" likelihood in $\mathbb{E}_{\mathbf{B}|A,\Theta}[\ln P(\mathbf{B} \cap A|\Theta)]$. It is the complete or joint probability of the observed data A and the unobserved or hidden or latent data B. Then this expectation defines the surrogate likelihood $Q(\Theta|\Theta^n)$ in the EM algorithm if the expectation is with respect to the

parametrized density $P(B|A, \Theta^n)$ for the nth parameter set Θ^n in the parameter sequence $\Theta^1, \Theta^2, \ldots, \Theta^n$. The other expectation $\mathbb{E}_{\mathbf{B}|A,\Theta^n}[\ln P(\mathbf{B}|A, \Theta)]$ is just an entropy term and does not affect the maximization of $Q(\Theta|\Theta^n)$.

We now recast this basic EM formulation in terms of the neural network's pdf structure. Then we derive EM's ascent property.

The EM algorithm iteratively maximizes the neural network's log-likelihood pdf $\ln p(\mathbf{y}|\mathbf{x}, \Theta)$ for network parameters Θ. The output y often depends on the input $\mathbf{x}$ through the hidden units $\mathbf{h}$. So we could simply write the network log-likelihood as $\ln p(\mathbf{y}|\mathbf{h}, \Theta)$ in such cases. But the output y may also depend directly on the input $\mathbf{x}$ as in "skip-layer" networks (Intrator & Intrator, 2001; Ripley, 1994) or in networks with still richer connection topologies. So we write the network log-likelihood as $\ln p(\mathbf{y}|\mathbf{x}, \Theta)$ or as $\ln p(\mathbf{y}|\mathbf{h}, \mathbf{x}, \Theta)$ for full generality.

The EM trick brings the hidden neurons $\mathbf{h}$ into the network pdf $p(\mathbf{y}|\mathbf{x}, \Theta)$ as in (68):

$$p(\mathbf{y}|\mathbf{x}, \Theta) = \frac{p(\mathbf{y}, \mathbf{x}|\Theta)}{p(\mathbf{x}|\Theta)} \tag{71}$$

$$= \frac{p(\mathbf{h}, \mathbf{y}, \mathbf{x}|\Theta)}{p(\mathbf{x}|\Theta)} \frac{p(\mathbf{y}, \mathbf{x}|\Theta)}{p(\mathbf{h}, \mathbf{y}, \mathbf{x}|\Theta)} \tag{72}$$

$$= \frac{p(\mathbf{h}, \mathbf{y}|\mathbf{x}, \Theta)}{p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta)} \tag{73}$$

Then taking logarithms gives the crucial EM log-likelihood equation in terms of the complete likelihood and hidden posterior:

$$\ln p(\mathbf{y}|\mathbf{x}, \Theta) = \ln p(\mathbf{h}, \mathbf{y}|\mathbf{x}, \Theta) - \ln p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta). \tag{74}$$

This log-likelihood equation underlies both EM's ascent property below and the proof of Theorem 1 that BP is generalized EM. EM's ascent property (Dempster et al., 1977) is a hill-climbing property. It states that any parameter choice Θ that increases $Q(\Theta|\Theta^n)$ can only increase the log-likelihood difference $\ln p(\mathbf{y}|\mathbf{x}, \Theta) - \ln p(\mathbf{y}|\mathbf{x}, \Theta^n)$. This result follows from Jensen's inequality and the concavity of the logarithm (Hogg et al., 2013). Those same two properties apply in the proof of Theorem 1.

The EM algorithm conditions on the hidden posterior $p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n)$ to estimate the hidden parameters $\mathbf{h}$ given all observed information $\mathbf{y}$ and $\mathbf{x}$ and given the current parameter estimate Θ^n. Taking this expectation on both sides of (74) gives

$$\ln p(\mathbf{y}|\mathbf{x}, \Theta) = \mathbb{E}_{\mathbf{h}|\mathbf{y},\mathbf{x},\Theta^n}\{\ln p(\mathbf{h}, \mathbf{y}|\mathbf{x}, \Theta)\} \tag{75}$$

$$- \mathbb{E}_{\mathbf{h}|\mathbf{y},\mathbf{x},\Theta^n}\{\ln p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta)\}$$

$$= Q(\Theta|\Theta^n) - \mathbb{E}_{\mathbf{h}|\mathbf{y},\mathbf{x},\Theta^n}\{\ln p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta)\} \tag{76}$$

$$= Q(\Theta|\Theta^n) + H(\Theta|\Theta^n) \tag{77}$$

where the differentiable cross entropy $H(\Theta|\Theta^n)$ is

$$H(\Theta|\Theta^n) = -\int_{\mathbf{h}} p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n) \ln p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta) d\mathbf{h} \,. \tag{78}$$

A similar version of the equality (77) also appears in Bishop (2006) and Oakes (1999).

We now state the network EM algorithm. The EM algorithm performs an E-step and then an M-step at each iteration or epoch n given some initial parameter value Θ^0. The E-step at n computes the above expectation $Q(\Theta|\Theta^n) = \mathbb{E}_{\mathbf{h}|\mathbf{y},\mathbf{x},\Theta^n}\{\ln p(\mathbf{h}, \mathbf{y}|\mathbf{x}, \Theta)\}$. This can involve approximation techniques for complicated expectations. Below we use a form of Monte Carlo importance sampling to estimate $Q(\Theta|\Theta^n)$.

The M-step maximizes the Q-function to find the next parameter estimate Θ^{n+1}:

$$\Theta^{n+1} = \arg\max_{\Theta} Q(\Theta|\Theta^n) \,. \tag{79}$$

This gives an inequality for the choice $\Theta = \Theta^n$:

$$Q(\Theta^{n+1}|\Theta^n) \geqslant Q(\Theta^n|\Theta^n) \,, \tag{80}$$

We show now that the Q-function inequality (80) and Jensen's inequality imply EM's ascent property for ML estimation:

$$\ln p(\mathbf{y}|\mathbf{x}, \Theta^{n+1}) \geqslant \ln p(\mathbf{y}|\mathbf{x}, \Theta^n) \,. \tag{81}$$

The proof is closely related to the proof of Theorem 1 that BP is generalized EM.

The ascent property (81) follows from the entropy inequality

$$H(\Theta|\Theta^n) \geqslant H(\Theta^n|\Theta^n) \quad \text{for all } \Theta \tag{82}$$

because (77) gives the inequality

$$\ln p(\mathbf{y}|\mathbf{x}, \Theta) - \ln p(\mathbf{y}|\mathbf{x}, \Theta^n) \tag{83}$$

$$= [Q(\Theta|\Theta^n) - Q(\Theta^n|\Theta^n)]$$

$$+ [H(\Theta|\Theta^n) - H(\Theta^n|\Theta^n)] \tag{84}$$

$$\geqslant Q(\Theta|\Theta^n) - Q(\Theta^n|\Theta^n). \tag{85}$$

Then (80) implies the result (81) for the parameter choice $\Theta = \Theta^{n+1}$ from the M-step (79).

The entropy inequality (82) follows from Jensen's inequality (Hogg et al., 2013) for convex functions because the logarithm is concave and thus its negative is convex:

$$H(\Theta^n|\Theta^n) - H(\Theta|\Theta^n) = \mathbb{E}_{\mathbf{h}|\mathbf{y},\mathbf{x},\Theta^n}\{\ln \frac{p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta)}{p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n)}\} \tag{86}$$

$$\leqslant \ln \mathbb{E}_{\mathbf{h}|\mathbf{y},\mathbf{x},\Theta^n}\{\frac{p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta)}{p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n)}\} \tag{87}$$

$$= \ln \int_{\mathbf{h}} \frac{p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta)}{p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n)} p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n) d\mathbf{h} \tag{88}$$

$$= \ln \int_{\mathbf{h}} p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta) d\mathbf{h} \tag{89}$$

$$= \ln 1 = 0 \tag{90}$$

since the pdf $p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta)$ integrates to unity. So Shannon entropy minimizes cross entropy: $H(\Theta|\Theta^n) \geqslant H(\Theta^n|\Theta^n)$ holds for all choices of parameter vector Θ.

This proof of the entropy inequality (82) also shows that the continuous K–L divergence is nonnegative: $KL(\Theta^n x \,\|\Theta) \geqslant 0$ because

$$KL(\Theta^n\|\Theta) = \int_{\mathbf{h}} p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n) \ln\left(\frac{p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n)}{p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta)}\right) d\mathbf{h} \tag{91}$$

$$= H(\Theta|\Theta^n) - H(\Theta^n|\Theta^n) \tag{92}$$

upon expanding the logarithm and distributing the integral.

A weaker form of the EM algorithm is the *generalized* EM (GEM) algorithm. The GEM algorithm only increases $Q(\Theta|\Theta^n)$ at each iteration n. GEM need not maximize the Q-function. GEM performs this partial optimization through gradient ascent:

$$\Theta^{n+1} = \Theta^n + \eta \nabla_{\Theta} Q(\Theta|\Theta^n)\Big|_{\Theta=\Theta^n} \tag{93}$$

where again η is a positive learning coefficient or a (usually decreasing) sequence of such coefficients. This still leads to the ascent property (81). The Noisy EM Theorem in the next section gives a sufficient condition for injected noise to increase the ascent at each iteration.

We can now state and easily prove Theorem 1. This fundamental theorem shows that BP is a special case of the GEM algorithm because their gradient updates coincide at each iteration n so long as BP invariance holds. This result follows from the gradient identity $\nabla_{\Theta} \ln p(\mathbf{y}|\mathbf{x}, \Theta^n) = \nabla_{\Theta} Q(\Theta^n|\Theta^n) + \nabla_{\Theta} H(\Theta^n|\Theta^n) =$

$\nabla_\Theta Q(\Theta^n|\Theta^n)$ since the null gradient $\nabla_\Theta H(\Theta^n|\Theta^n) = \mathbf{0}$ holds given the entropy inequality (82) and given Fermat's Theorem for gradients. This gives the master gradient equation at a given layer (the output layer in particular):

$$\nabla_\Theta \ln p(\mathbf{y}|\mathbf{x}, \Theta^n) = \nabla_\Theta Q(\Theta^n|\Theta^n). \tag{94}$$

The proof in the Appendix gives the complete details.

Theorem 1 (*Backpropagation as the GEM Algorithm*)**.** *The backpropagation update equation for a differentiable likelihood function* $p(\mathbf{y}|\mathbf{x}, \Theta)$ *at epoch* n

$$\Theta^{n+1} = \Theta^n + \eta \nabla_\Theta \ln p(\mathbf{y}|\mathbf{x}, \Theta)\Big|_{\Theta=\Theta^n} \tag{95}$$

equals the GEM update equation at epoch n

$$\Theta^{n+1} = \Theta^n + \eta \nabla_\Theta Q(\Theta|\Theta^n)\Big|_{\Theta=\Theta^n} \tag{96}$$

where GEM uses the differentiable Q-function

$$Q(\Theta|\Theta^n) = \mathbb{E}_{\mathbf{h}|\mathbf{y},\mathbf{x},\Theta^n}\left\{\ln p(\mathbf{y}, \mathbf{h}|\mathbf{x}, \Theta)\right\}. \tag{97}$$

We show next how Monte Carlo importance sampling can approximate the Q-function expectation in (97).

The approximation assumes that the hidden-layer neurons are Bernoulli random variables. Then the activation a_j^h of the jth hidden neuron defines the two conditional probabilities

$$p(h_j = 1|\mathbf{x}, \Theta) = a_j^h \tag{98}$$

and

$$p(h_j = 0|\mathbf{x}, \Theta) = 1 - a_j^h. \tag{99}$$

This gives the jth hidden unit's pdf as the Bernoulli pdf

$$p(h_j|\mathbf{x}, \Theta) = (a_j^h)^{h_j}(1 - a_j^h)^{1-h_j} \tag{100}$$

where $h_j = 0$ or $h_j = 1$.

The hidden units are conditionally independent within a layer given the input $\mathbf{x}$ and network parameters Θ. So the hidden *prior* pdf $p(\mathbf{h}|\mathbf{x}, \Theta)$ factors and has a product Bernoulli form

$$p(\mathbf{h}|\mathbf{x}, \Theta) = \prod_{j=1}^{J} p(h_j|\mathbf{x}, \Theta) = \prod_{j=1}^{J} (a_j^h)^{h_j}(1 - a_j^h)^{1-h_j}. \tag{101}$$

So the probability structure of the hidden layer corresponds to flipping the same coin J times.

The EM algorithm's E-step computes the Q-function in (97). Computing the expectation in (97) requires 2^J values of $p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n)$. This is computationally intensive for large values of J. So we can use ordinary 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 with enough random samples (Hogg et al., 2013).

Bayes theorem gives the hidden *posterior* density $p(\mathbf{h}|\mathbf{x}, y, \Theta^n)$ as the ratio

$$p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n) = \frac{p(\mathbf{h}|\mathbf{x}, \Theta^n)p(\mathbf{y}|\mathbf{h}, \mathbf{x}, \Theta^n)}{\sum_{\mathbf{h}} p(\mathbf{h}|\mathbf{x}, \Theta^n)p(\mathbf{y}|\mathbf{h}, \mathbf{x}, \Theta^n)}. \tag{102}$$

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

$$p(\mathbf{h}|\mathbf{x}, \Theta^n) \approx \frac{1}{M}\sum_{m=1}^{M} \delta_K(\mathbf{h} - \mathbf{h}^m) \tag{103}$$

where δ_K is the J-dimensional Kronecker delta function. The standard error in the approximation falls off as the inverse of the square root of the sample size M. Then the Monte Carlo approximation of the hidden posterior becomes

$$p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n) \approx \frac{\sum_{m=1}^{M} \delta_K(\mathbf{h} - \mathbf{h}^m)p(\mathbf{y}|\mathbf{h}, \mathbf{x}, \Theta^n)}{\sum_{\mathbf{h}}\sum_{m_1=1}^{M} \delta_K(\mathbf{h} - \mathbf{h}^{m_1})p(\mathbf{y}|\mathbf{h}, \mathbf{x}, \Theta^n)} \tag{104}$$

$$= \frac{\sum_{m=1}^{M} \delta_K(\mathbf{h} - \mathbf{h}^m)p(\mathbf{y}|\mathbf{h}^m, \mathbf{x}, \Theta^n)}{\sum_{m_1=1}^{M} p(\mathbf{y}|\mathbf{h}^{m_1}, \mathbf{x}, \Theta^n)} \tag{105}$$

$$= \sum_{m=1}^{M} \delta_K(\mathbf{h} - \mathbf{h}^m)\gamma^m \tag{106}$$

where the weights γ^m have the Bayesian form

$$\gamma^m = \frac{p(\mathbf{y}|\mathbf{h}^m, \mathbf{x}, \Theta^n)}{\sum_{m_1=1}^{M} p(\mathbf{y}|\mathbf{h}^{m_1}, \mathbf{x}, \Theta^n)} \tag{107}$$

and give the relative importance or "responsibility" (Bishop, 2006) of $\mathbf{h}^m$. So (106) gives an importance-sampled approximation of $p(\mathbf{h}|\mathbf{y}, \mathbf{x}, \Theta^n)$ where each sample $\mathbf{h}^m$ has weight γ^m.

Approximate the surrogate likelihood Q-function as

$$Q(\Theta|\Theta^n) \approx \sum_{\mathbf{h}}\sum_{m=1}^{M} \gamma^m \delta_K(\mathbf{h} - \mathbf{h}^m) \ln p(\mathbf{y}, \mathbf{h}|\mathbf{x}, \Theta) \tag{108}$$

$$= \sum_{m=1}^{M} \gamma^m \ln p(\mathbf{y}, \mathbf{h}^m|\mathbf{x}, \Theta) \tag{109}$$

$$= \sum_{m=1}^{M} \gamma^m \Big[\ln p(\mathbf{h}^m|\mathbf{x}, \Theta) + \ln p(\mathbf{y}|\mathbf{h}^m, \mathbf{x}, \Theta)\Big] \tag{110}$$

since $p(\mathbf{y}, \mathbf{h}^m|\mathbf{x}, \Theta) = p(\mathbf{h}^m|\mathbf{x}, \Theta)p(\mathbf{y}|\mathbf{h}^m, \mathbf{x}, \Theta)$. Then the above Bernoulli structure of the hidden prior $p(\mathbf{h}^m|\mathbf{x}, \Theta)$ gives

$$\ln p(\mathbf{h}^m|\mathbf{x}, \Theta) = \ln \prod_{j=1}^{J} p(h_j^m|\mathbf{x}, \Theta) \tag{111}$$

since again the hidden neurons in a layer are conditionally independent of one another given the input $\mathbf{x}$

$$= \ln \prod_{j=1}^{J} (a_j^h)^{h_j^m}(1 - a_j^h)^{1-h_j^m} \tag{112}$$

$$= \sum_{j=1}^{J} \Big[h_j^m \ln a_j^h + (1 - h_j^m)\ln(1 - a_j^h)\Big] \tag{113}$$

if the hidden-layer activations approximate Bernoulli probabilities.

The Q-function in (110) equals a sum of log-likelihood functions for two 2-layer neural networks between the visible-hidden layer and the hidden-output layer. The M-step maximizes or improves this Q-function by gradient ascent. So the gradient ascent corresponds to taking two distinct BP steps on the two 2-layer neural networks.

4. The Noisy Expectation–Maximization theorem

The Noisy Expectation–Maximization (NEM) algorithm (Osoba et al., 2011b, 2013) modifies the EM iterative scheme at each step and convergences faster on average than does noiseless EM. It injects additive noise into the data at each EM iteration. Injecting multiplicative noise or other signal-noise-combined noise still improves the ascent property of EM (Dempster et al., 1977) at each iteration on average (Osoba & Kosko, 2016a).

The NEM noise intensity or variance also decays slightly with the iteration count. This guarantees convergence to the optimal

parameters of the original data model. The estimates would otherwise only jitter around the optimal value. But the noise must satisfy the NEM positivity condition below that guarantees that the NEM parameter estimates will climb faster up the likelihood surface on average.

The motivation for the NEM positivity condition stems from a simple likelihood inequality. Suppose that there is some additive noise n that makes the signal observation y more probable given some parameter Θ. Then the pdf inequality $p(y+n|\Theta) \geqslant p(y|\Theta)$ holds. The values n and y are realizations of the respective random variables $\mathbf{n}$ and $\mathbf{y}$. Then the pdf inequality holds if and only if $\ln \frac{p(y+n|\Theta)}{p(y|\Theta)} \geqslant 0$. This latter term is the log-likelihood ratio often found in ML estimation (Hogg et al., 2013). Then taking the expectation over all random variables gives the NEM positivity (non-negativity) condition as in Theorem 2. Taking the expectation implies that the log-likelihood-ratio inequality need hold only almost everywhere. It need not hold on sets of zero probability.

The next section presents the formal statement of the NEM Theorem for additive noise injection.

4.1. NEM theorem for additive noise injection

The NEM Theorem (Osoba et al., 2011b, 2013) states a general sufficient condition when noise speeds up the EM algorithm's average convergence to a local maximum of the network probability or log-likelihood.

The NEM Theorem assumes that the noise random variable $\mathbf{n}$ has pdf $p(\mathbf{n}|\mathbf{x})$. So the noise $\mathbf{n}$ can depend on the data $\mathbf{x}$. Such noise dependence implies that NEM noise benefit differs from most "stochastic resonance" noise benefits where the user injects independent noise or dither (Bulsara et al., 1989; Kosko, 2006; McDonnell et al., 2008; Mitaim & Kosko, 2014). The hidden variables $\mathbf{h}$ are the latent variables in the EM model. The vector sequence $\{\Theta^n\}$ is a sequence of EM estimates for parameter vector Θ. So the maximum-likelihood parameter vector $\Theta^* = \lim_{n\to\infty} \Theta^n$ is the converged EM estimate for Θ.

Define the noisy Q function $Q_{\mathbf{n}}(\Theta|\Theta^n)$ as the expected log-likelihood $Q_{\mathbf{n}}(\Theta|\Theta^n) = \mathbb{E}_{\mathbf{h}|\mathbf{x},\Theta_n}[\ln p(\mathbf{x}+\mathbf{n},\mathbf{h}|\Theta)]$. So $Q_{\mathbf{n}}(\Theta|\Theta^n)$ is a random variable because the expectation does not average out the noise random variable $\mathbf{N}$. Assume again that the differential entropy of all random variables is finite. Assume also that the additive noise keeps the data in the likelihood function's support. Then we can state the NEM theorem (Osoba et al., 2011b, 2013) in the special but important case of additive noise injection.

Theorem 2 (*Noisy Expectation–Maximization (NEM)*)**.** *Suppose the average positivity condition holds at iteration n:*

$$\mathbb{E}_{\mathbf{x},\mathbf{h},\mathbf{n}|\Theta^*}\left[\ln\left(\frac{p(\mathbf{x}+\mathbf{n},\mathbf{h}|\Theta^n)}{p(\mathbf{x},\mathbf{h}|\Theta^n)}\right)\right] \geqslant 0\,. \tag{114}$$

Then the EM noise benefit

$$Q(\Theta^n|\Theta^*) \leqslant Q_{\mathbf{n}}(\Theta^n|\Theta^*) \tag{115}$$

holds on average at iteration n:

$$\begin{aligned}&\mathbb{E}_{\mathbf{x},\mathbf{n}|\Theta^n}\left[Q\left(\Theta^*|\Theta^*\right) - Q_{\mathbf{n}}\left(\Theta^n|\Theta^*\right)\right]\\ &\leqslant \mathbb{E}_{\mathbf{x}|\Theta^n}\left[Q\left(\Theta^*|\Theta^*\right) - Q\left(\Theta^n|\Theta^*\right)\right]\,.\end{aligned} \tag{116}$$

The NEM Theorem states that each iteration of a suitably noisy EM algorithm gives higher likelihood estimates on average than do the noiseless EM's estimates. So the NEM algorithm converges faster than EM on average (and almost always in practice). 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.

A natural question is whether the NEM positivity inequality (114) can hold at all: Is the inequality vacuous? The inequality may appear to violate intuitions about the concavity and Jensen's inequality that dictate the related entropy inequality (82). But (114) does hold in general because the expectation conditions on the converged parameter vector Θ^* and not on a simpler pdf.

We show this result with Jensen't inequality. Consider the expectation of an ordinary log-likelihood ratio $\ln \frac{f(y|\Theta)}{g(y|\Theta)}$ (Hogg et al., 2013). Take the expectation of $\ln \frac{f(y|\Theta)}{g(y|\Theta)}$ with respect to the pdf $g(y|\Theta)$ to get $\mathbb{E}_g[\ln \frac{f(y|\Theta)}{g(y|\Theta)}]$. But the logarithm is concave. So Jensen's inequality gives $\mathbb{E}_g[\ln \frac{f(y|\Theta)}{g(y|\Theta)}] \leqslant \ln \mathbb{E}_g[\frac{f(y|\Theta)}{g(y|\Theta)}]$. Then the pdf $g(y|\Theta)$ cancels: $\ln \mathbb{E}_g[\frac{f(y|\Theta)}{g(y|\Theta)}] = \ln \int_Y \frac{f(y|\Theta)}{g(y|\Theta)} g(y|\Theta)\, dy = \ln \int_Y f(y|\Theta)\, dy = \ln 1 = 0$ because $f(y|\Theta)$ is a pdf. So $\mathbb{E}_g[\ln \frac{f(y|\Theta)}{g(y|\Theta)}] \leqslant 0$. So strict positivity condition is impossible in this case. But the cancellation argument does not apply to the NEM expectation in (114) in general because the integrating pdf depends on Θ^* in (114) and not on Θ^n. So cancellation occurs only when the NEM algorithm has converged because then $\Theta^n = \Theta^*$.

Modified EM (and NEM) can perform *maximum a posteriori* (MAP) estimation for problems of missing information. The MAP or Bayesian version modifies the Q-function by adding the log-prior term $G(\Theta) = \ln p(\Theta)$ (Dempster et al., 1977; McLachlan & Krishnan, 2007):

$$Q(\Theta|\Theta^n) = \mathbb{E}_{\mathbf{h}|\mathbf{x},\Theta_t}[\ln p(\mathbf{x},\mathbf{h}|\Theta)] + G(\Theta)\,. \tag{117}$$

The MAP version of the NEM algorithm makes a similar change to the $Q_{\mathbf{n}}$-function:

$$Q_{\mathbf{n}}(\Theta|\Theta^n) = \mathbb{E}_{\mathbf{h}|\mathbf{x},\Theta_t}[\ln p(\mathbf{x}+\mathbf{n},\mathbf{h}|\Theta)] + G(\Theta)\,. \tag{118}$$

This NEM extension resembles the recent noise-boost of simulated and quantum annealing and more generally Markov Chain Monte Carlo (MCMC) statistical estimation (Franzke & Kosko, 2015).

Many latent-variable models are not identifiable (Teicher, 1963). So they need not have global optima. These models include Gaussian mixture models (McLachlan & Peel, 2000), hidden Markov models (Rabiner, 1989), 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 do for non-identifiable models. Users can also run several NEM simulations from different random starting points and then pick the best performer.

5. Injecting NEM noise in output neurons

The two theorems in this section show how injecting NEM noise into a neural network's output neurons can speed convergence in classifier/logistic neurons and in regression networks.

The first theorem adds noise $\mathbf{n}$ to the 1-in-K encoding $\mathbf{t}$ of the target variable $\mathbf{y}$ of a classifier network with softmax or logistic output neurons. Both cases yield simple hyperplane noise conditions. They define different forbidden regions in noise space. Fig. 2 shows a typical speed-up in BP convergence when NEM noise adds only to the 10 output softmax neurons in a 5-layer network The noise-boosted network hit the knee of the convergence curve after just 4 iterations while noiseless BP took 15 iterations to get to the same place.

The second theorem derives a spherical noise region for a regression network with identity output neurons. The spherical structure arises from the vector-Gaussian target vector $\mathbf{t}$. All proofs are in Appendix.

The next section extends these results to allow NEM noise injection into the hidden neurons. This hidden-noise injection requires using the proper layer NEM condition for the hidden neurons based on their activation type and the corresponding layer log-likelihood.

Theorem 3 (*Hyperplane Noise Benefit for Injecting Noise in a Classifier Network's Output Layer*)**.** *The NEM positivity condition* (114) *holds for maximum-likelihood training of a classifier neural network with output Gibbs or softmax activations if the following average hyperplane condition holds at iteration n:*

$$\mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\left\{\mathbf{n}^T \ln \mathbf{a}^t\right\} \geqslant 0 \,. \tag{119}$$

The NEM condition (114) *also holds for injecting noise in output* logistic *neurons if*

$$\mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\{\mathbf{n}^T \ln \mathbf{a}^t\} \geqslant \mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\{\mathbf{n}^T \ln(\mathbf{1} - \mathbf{a}^t)\} \,. \tag{120}$$

The above sufficient NEM condition (A.14) requires that the noise vector $\mathbf{n}$ lies above a hyperplane with normal $\ln \mathbf{a}^t$. So the logistic NEM noise-injection algorithm uses noise samples $\mathbf{n}$ that obey the noise-weighted log-odds inequality

$$\sum_{k=1}^{K} n_k \ln \frac{a_k^t}{1 - a_k^t} \geqslant 0 \,. \tag{121}$$

The next section uses this result to inject NEM noise into the hidden neurons because they are logistic. Other types of hidden neurons must use the appropriate log-likelihood function in the above derivation.

The next theorem gives a sufficient condition for a noise benefit in a regression neural network with a Gaussian target vector $\mathbf{t} \sim \mathcal{N}(\mathbf{t}|\mathbf{a}^t, \mathbf{I})$ from (49). The condition defines a spherical noise-benefit region in noise space.

Theorem 4 (*Regression Hypersphere Noise Benefit*)**.** *The NEM positivity condition* (114) *holds at iteration n for maximum-likelihood training of a regression neural network with Gaussian target vector* $\mathbf{t} \sim \mathcal{N}(\mathbf{t}|\mathbf{a}^t, \mathbf{I})$ *if*

$$\mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|,\mathbf{x},\Theta^*}\left\{\|\mathbf{n} - \mathbf{a}^t + \mathbf{t}\|^2 - \|\mathbf{a}^t - \mathbf{t}\|^2\right\} \leqslant 0 \tag{122}$$

where $\|.\|$ *is the Euclidean vector norm.*

The spherical NEM condition defines a forbidden-noise region outside a sphere in noise space with center $\mathbf{t} - \mathbf{a}^t$ and radius $\|\mathbf{t} - \mathbf{a}^t\|$. All noise inside this sphere speed the average ML convergence of the neural network.

The proof of Theorem 4 shows that we can also perturb the network *parameters* to achieve a NEM-noise benefit. The additive structure of the above NEM condition shows that we can add the NEM noise directly to the mean-vector parameter $\mathbf{a}^t$ instead of to the target vector $\mathbf{t}$.

We can also multiplicatively perturb the normal density's identity covariance matrix $\mathbf{I}$ by a variance $\sigma^2 > 0$ to give the new covariance matrix $\sigma^2\mathbf{I}$. This gives the NEM-perturbation likelihood ratio as an exponential since the likelihoods are Gaussian:

$$\frac{\mathcal{N}(\mathbf{t}|\mathbf{a}^t, \sigma^2\mathbf{I})}{\mathcal{N}(\mathbf{t}|\mathbf{a}^t, \mathbf{I})} = \frac{\exp\{\frac{1}{\sigma^2}(-\frac{1}{2}\sum_{k=1}^{K}(t_k - a_k^t)^2)\}}{\exp\{-\frac{1}{2}\sum_{k=1}^{K}(t_k - a_k^t)^2\}} \tag{123}$$

$$= \exp\{(1 - \frac{1}{\sigma^2})(\frac{1}{2}\sum_{k=1}^{K}(t_k - a_k^t)^2)\} \,. \tag{124}$$

The NEM sufficient condition (114) takes the logarithm of this likelihood ratio and demands that its average be nonnegative. This gives a noise perturbation benefit when

$$\sigma^2 \geqslant 1 \tag{125}$$

when the quadratic term is nonzero. The same argument shows that the NEM condition for an additive covariance perturbation $\mathbf{I} + n\mathbf{I} = (1 + n)\mathbf{I}$ is just $n \geqslant 0$.

The proof of Theorem 4 also shows that the same spherical noise-benefit condition (A.22) holds for a *regularized* regression network. A Tikhonov regularizer adds the squared-norm parameter term $\lambda \sum_l \theta_l^2$ as a λ-scaled penalty term to the squared norm of $\mathbf{t} - \mathbf{a}$ (Girosi, Jones, & Poggio, 1995). This regularizer corresponds to a normal prior in a Bayesian probabilistic interpretation (Bishop, 2006). So it is proportional to an exponential that contains the regularizer term. Then the network posterior density is the product of the normal likelihood and the normal prior and thus is still normal. This normal posterior $\mathcal{N}_{\text{regularized}}(\mathbf{t}|\mathbf{a}^t, \mathbf{I})$ now includes the regularizer sum as an additive term in its exponent. But we do not add noise to this term because it involves only network parameters. So the regularizer term cancels out from the NEM ratio to give

$$\frac{\mathcal{N}_{\text{regularized}}(\mathbf{t} + \mathbf{n}|\mathbf{a}^t, \mathbf{I})}{\mathcal{N}_{\text{regularized}}(\mathbf{t}|\mathbf{a}^t, \mathbf{I})} = \frac{\mathcal{N}(\mathbf{t} + \mathbf{n}|\mathbf{a}^t, \mathbf{I})}{\mathcal{N}(\mathbf{t}|\mathbf{a}^t, \mathbf{I})} \tag{126}$$

as in (A.20). So (A.22) still holds for a regularized network. This remains true for an l_1 or *lasso* regularizer because it enters the posterior as a Laplacian or doubly exponential prior (Tibshirani, 1996). So it still results in a ratio of exponentials where the lasso regularizer cancels out of the NEM ratio.

The more general result is that *any* noiseless prior will cancel out of the NEM posterior ratio and give back the likelihood ratio in (A.20) and thus in (A.22).

This section presented sufficient conditions for a noise benefit in training a neural network that uses the BP/EM algorithm. Reversing the inequalities in the noise benefit theorems and proofs yields symmetric noise *harm* results for injecting noise that lies below the NEM hyperplane for a classifier network or outside the NEM sphere for a regression network.

6. Injecting NEM noise in hidden neurons

The previous noise results added noise to only the output neurons. We now derive the NEM noise-benefit condition for injecting NEM noise into hidden neurons a layer at a time during BP training. This applies to both regression and classifier networks as well as to networks with logistic output neurons. NEM-noise injection takes care here because a given hidden layer's log-likelihood L_h may differ from the log-likelihood of the output layer or from other hidden layers.

Fig. 10 shows the substantial training speed-up that occurred when we injected NEM noise into all the neurons of a 3-layer regression network that learned the test function $f(x) = \sin x$. Adding NEM noise to just the regression network's single output neuron reduced the squared error. Further adding NEM noise to the 10 hidden logistic neurons markedly reduced the squared error.

Adding NEM noise to the 10 output softmax neurons of a 4-layer neural classifier markedly reduced the average test-set cross-entropy for the MNIST training data. Further adding NEM noise to the hidden neurons further decreased the cross entropy. More complex classification tasks should see corresponding decrease in cross entropy in much larger deep networks. Injecting NEM noise also improved classification accuracy in apparent accord with the accuracy bounds in the next section. Training with NEM noised produced up to 35% improvement on test data for both regression and classification. Injecting blind noise only hurts performance for both regressors and classifiers.

Hidden-layer noise injection must distinguish between two cases: Injecting the *same* NEM noise from the output layer into the hidden layer versus injecting fresh NEM noise at the hidden

layer after injecting separate NEM noise at the output layer. The first case can over-constrain the NEM noise if the noise injection occurs in multiple hidden layers. We first discuss this constrained case and then present the unconstrained case as a theorem. The simulations in Fig. 10 used the unconstrained noise injection in the next theorem and in Algorithm 1.

Suppose we have injected NEM noise $\mathbf{n}$ into the output layer and want to inject the same noise into the last hidden layer $\mathbf{h}_k$. The proof of Theorem 3 shows that the NEM noise $\mathbf{n}$ added to the output targets $\mathbf{t}$ adds to the error $\mathbf{e}^t$

$$\mathbf{e}^t = \mathbf{t} - \mathbf{a}^t \tag{127}$$

of the output neurons. So the noisy error vector $\mathbf{e}_N^t$ is

$$\mathbf{e}_N^t = \mathbf{t} + \mathbf{n} - \mathbf{a}^t \tag{128}$$

$$= \mathbf{e}^t + \mathbf{n} \,. \tag{129}$$

Then this noisy error vector $\mathbf{e}_N^t$ propagates back over the weights to the hidden layer.

The weight matrix $\mathbf{U}$ connects the J hidden units to the K output neurons. So passing $\mathbf{e}_N^t$ backwards uses the matrix $\mathbf{U}$ in this notation (other formulations would use the matrix transpose $\mathbf{U}^T$ throughout) . Then the error $\mathbf{e}_N^h$ that arrives at the hidden layer is

$$\mathbf{e}_N^h = \mathbf{U}\mathbf{e}_N \tag{130}$$

$$= \mathbf{U}\mathbf{e}^t + \mathbf{U}\mathbf{n} \,. \tag{131}$$

The forward pass sees the hidden neuron activations as visible data. So this linearly transformed noise satisfies the NEM sufficient condition at the *logistic* hidden layer if

$$(\mathbf{U}\mathbf{n})^T \ln \mathbf{a}^h \geqslant (\mathbf{U}\mathbf{n})^T \ln(\mathbf{1} - \mathbf{a}^{\mathbf{h}}) \tag{132}$$

from (120) where $\mathbf{a}^h$ are the hidden layer activations. This hidden-layer NEM condition will change in accord with BP invariance if the neurons are not logistic and thus have a different layer log-likelihood L. The same argument shows that the transformed NEM noise $\mathbf{U}\mathbf{n}$ applies to the next hidden layer if it scales by the appropriate weight matrix and obeys the appropriate layer NEM condition for its layer log-likelihood.

We turn next to the more general case of injecting fresh NEM noise at the kth hidden layer in accord with that layer's log-likelihood function. The key idea is that NEM-noise injection in the kth hidden layer depends only on the preceding layers and input $\mathbf{x}$. It does not depend on the higher layers or on the output layer $\mathbf{y}$. This follows from the multiplication theorem of basic probability that factors the total network likelihood $p(\mathbf{y}, \mathbf{h}_k, \ldots, \mathbf{h}_1, |\mathbf{x}, \Theta^n)$ into the product layer likelihoods:

$$\begin{aligned} p(\mathbf{y}, \mathbf{h}_k, \ldots, \mathbf{h}_1, |\mathbf{x}, \Theta^n) = p(\mathbf{y}|\mathbf{h}_k, \ldots, \mathbf{h}_1, \mathbf{x}, \Theta^n) \times \\ p(\mathbf{h}_k|\mathbf{h}_{k-1}, \ldots, \mathbf{h}_1, \mathbf{x}, \Theta^n) \cdots p(\mathbf{h}_2|\mathbf{h}_1, \mathbf{x}, \Theta^n) p(\mathbf{h}_1|\mathbf{x}, \Theta^n). \end{aligned} \tag{133}$$

Taking logarithms in (133) allows unconstrained NEM noise for the kth layer log-likelihood so long as BP invariance holds. The sum structure of these log-likelihoods shows that NEM noise can boost any or all of these layers at a given training epoch. The next theorem presents this general result for the common case of logistic hidden neurons.

Theorem 5 (*NEM Noise in Hidden Logistic Neurons*)**.** *NEM noise* $\mathbf{n}$ *boosts a given hidden layer of logistic neurons if the injected noise satisfies the NEM likelihood inequality*

$$\mathbb{E}_{\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\{\mathbf{n}^T \ln \mathbf{a}^h\} \geqslant \mathbb{E}_{\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\{\mathbf{n}^T \ln (\mathbf{1} - \mathbf{a})^h\} \,. \tag{134}$$

for the hidden-layer activation vector $\mathbf{a}^h$ *with the logistic layer-likelihood structure* (60)–(62).

Data: T input data vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, T target label 1-in-K vectors $\{\mathbf{y}_1, \ldots, \mathbf{y}_T\}$, number of BP epochs R
Result: Trained DNN weight matrices $\mathbf{U}$ and $\mathbf{W}$
while *epoch* $r : 1 \to R$ **do**
  while *training data vector number* $t : 1 \to T$ **do**
    • Propagate the input data vector $\mathbf{x}_t$ forward through the neural network with (3);
    • Compute the K-dimensional output softmax activation vector $\mathbf{a}$ with (10);
    • Generate the output noise vector $\mathbf{n}$;
    if $\mathbf{n}^T \ln \mathbf{a}^t \geqslant 0$ **then**
      • Add the NEM noise: $\mathbf{y}_t \leftarrow \mathbf{y}_t + \mathbf{n}$;
    else
      • Do nothing
    end
    • Compute the error $\mathbf{y}_t - \mathbf{a}$;
    • Back-propagate the error to compute the cross-entropy gradient $\nabla_{\mathbf{U}} E(\Theta)$ or $-\nabla_{\mathbf{U}} L(\Theta)$;
    • Generate the hidden noise vector $\mathbf{m}$;
    if $\mathbf{m}^T \ln \mathbf{a}^h \geqslant \mathbf{m}^T \ln(\mathbf{1} - \mathbf{a}^{\mathbf{h}})$ **then**
      • Add the NEM noise: $\mathbf{h}_t \leftarrow \mathbf{h}_t + \mathbf{m}$;
    else
      • Do nothing
    end
    • Back-propagate the error to compute the cross-entropy gradient $\nabla_{\mathbf{W}} E(\Theta)$ or $-\nabla_{\mathbf{W}} L(\Theta)$;
    • Update the network parameter matrices $\mathbf{U}$ and $\mathbf{W}$ with the gradient descent in (174)
  end
end

Algorithm 1: The NEM–BP algorithm for total NEM noise injection for a neural network with one hidden layer. The NEM noise injects both into the output layer and into the hidden layer. The algorithm extends to deep networks with arbitrarily many hidden layers.

The proof of Theorem 5 also shows how to inject NEM noise in hidden neurons with ReLu or "rectified linear" units $h(x) = \max(x, 0)$ and its variants. ReLu units may help reduce the problem of "vanishing gradients" that sigmoidal units can produce in deep networks (Rawat & Wang, 2017).

ReLu units have identity activations for positive inputs. So we can approximate the ReLu-layer likelihood function with the normal likelihood for identity activations as in the case of a regression network. We simply replace the NEM regression noise $\mathbf{n}$ with $\mathbf{U}^T\mathbf{n}$ in the hyperspherical NEM condition (182). A better but more complex approximation would rework the logistic-likelihood argument in the proof of Theorem 5 with a truncated-normal likelihood.

7. NEM noise benefits in classification accuracy

Noise can improve network classification accuracy as well as speed BP convergence. Fig. 4 demonstrates this boost in accuracy for a classification network with three hidden layers and 10 output neurons. Fig. 5 shows a similar result. NEM noise adds only to the output neurons in both cases. Fig. 10 reports more substantial NEM gains in classification accuracy because noise adds to all the neurons in the network. NEM-boosted recurrent backpropagation also improved classification accuracy of videos (Adigun & Kosko, 2017).

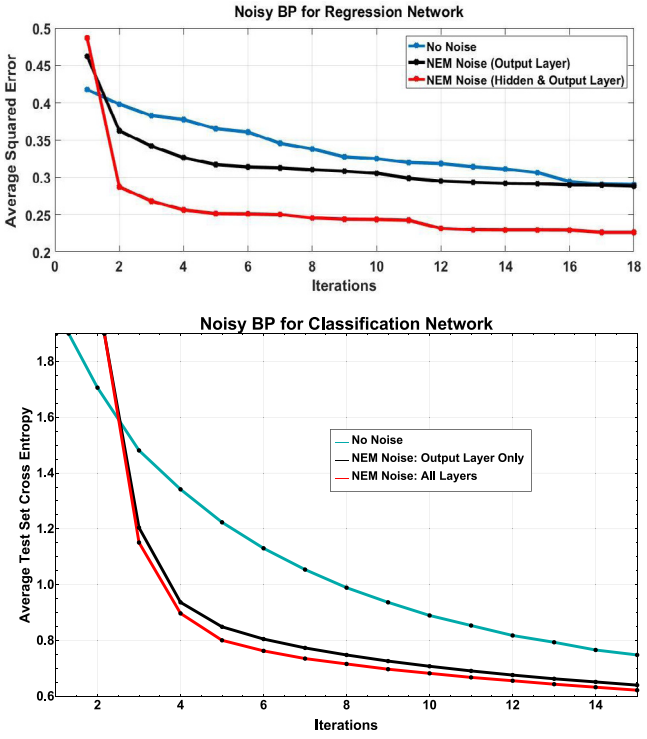

Fig. 10. NEM Noise injection in output and hidden layers for backpropagation training of a regression and classifier network. The first figure shows the noise-boost that results when injecting NEM noise into a 3-layer regression neural network. The regression network approximates the function $f(x) = \sin x$ over the domain $[0, 2\pi]$ using 18,000 randomly drawn training samples. The input and output layers contained just one identity neuron each. The hidden layer contained 10 logistic hidden neurons. Adding NEM noise to just the single output neuron reduced the average squared-error of training over noiseless BP training. Adding NEM noise to both the output and hidden neurons markedly reduced the squared error even further. Adding blind noise to the network only slowed learning convergence. The second figure shows the noise-boost from injecting NEM noise into a 4-layer classifier trained on the MNIST digit set. The 10 output neurons had softmax activations. The 40 neurons in each of the two hidden layers had logistic activations. Injecting NEM noise in the 10 output neurons quickly and markedly reduced the average test-set cross entropy compared with noiseless BP training. Injecting NEM noise into the output and hidden layers further reduced the test-set cross entropy. Adding blind noise performed worse than noiseless BP. Training with NEM noise led to 35% improvement for both the regression and classification tasks.

We offer two explanations of this consistently observed accuracy boost.

The first explanation is a general but indirect argument: Theorem 6 shows that the network log-likelihood of a classifier network gives a lower bound on the classification accuracy. NEM noise only increases the log-likelihood on average. So it can only increase the classification accuracy. The results in the lemma below and Theorem 6 hold in general for statistical classifiers.

The second explanation depends directly on the nature of the injected NEM noise. NEM noise (2) is just that noise that makes the output target more probable. It does this by increasing the activation of the correct output activation and thereby lowering the activations of the other $K-1$ output activations. These output activations have softmax form and so define a length-K probability distribution. So NEM noise only makes correct classification more probable.

Both explanations are only partial because they apply only to the training of the classifier network. The observed accuracy boost occurs later with trained networks. This also suggests adding some form of the above stochastic-resonance noise during post-training use to improve classification.

We start with the first explanation and show that the network likelihood L is a lower bound for the classification accuracy A in binary classification. So noise-boosting L tends only to increase A.

We first develop this result for a classifier network with a single output neuron. The result is the bound $A \geqslant L + \ln 2$ in (139). Theorem 6 extends this result to the general case of K output neurons for both the softmax-based likelihood in (22) and the logistic likelihood in (62).

So assume the classifier network has a single output neuron. Then the output activation is both softmax and logistic since $K = 1$. We need suppose only that this lone output neuron has non-decreasing activation $a^t \in (0, 1]$. Let t denote the binary target value for the output neuron: $t \in \{0, 1\}$. So a '1' codes for one of the two input pattern classes and a '0' codes for its set complement thus the other pattern class.

Complete classification accuracy measures both the true-positive and true-negative classifications. A true-positive classification occurs if both $t = 1$ and $a^t \geqslant \frac{1}{2}$ because then we round off the observed output a^t to 1. So a false positive (or false alarm) occurs if both $t = 0$ and $a^t \geqslant \frac{1}{2}$. A true negative occurs if both $t = 0$ and $a^t < \frac{1}{2}$. A false negative (or miss) occurs if both $t = 1$ and $a^t < \frac{1}{2}$. Then the *complete accuracy* A counts both the true positives and true negatives:

$$A = t\, I\Big(a^t \geqslant \frac{1}{2}\Big) + (1 - t)\, I\Big(a^t < \frac{1}{2}\Big). \tag{135}$$

I is a binary indicator function: $I(E) = 1$ if event E occurs and $I(E) = 0$ if E does not occur. Then $A = 1$ when a true positive or true negative occurs. $A = 0$ when a false positive or false negative occurs. The corresponding log-likelihood L function is

$$L = t\ \ln(a^t) + (1 - t)\ \ln(1 - a^t)\,. \tag{136}$$

The proof of the accuracy-likelihood bound (139) uses the following lemma for real numbers. The lemma gives a logarithmic lower bound on the indicator function $I\Big(x \geqslant \frac{1}{2}\Big)$ and more.

Lemma 1. *Let $x \in (0, 1]$ and $y \in (0, 1]$. Then*

$$I\Big(x \geqslant y\Big) \geqslant \ln\Big(\frac{x}{y}\Big) \tag{137}$$

if $y \geqslant x/e$.

Lemma 1 implies that

$$I\Big(x \geqslant \frac{1}{2}\Big) \geqslant \ln(2x) \tag{138}$$

for all x in $(0, 1]$ because $2x < e$ holds for all such x.

The next result shows that the log-likelihood L in (136) is a lower-bound on the classification accuracy A in (135) if the classification or logistic network has a single output neuron. So the NEM noise benefit during BP/EM training tends only to increase the accuracy. The next theorem extends this one-output-neuron result to K output neurons that have softmax activations or logistic activations subject to a simplification of the complete accuracy A for classifier networks.

We first state the one-output-neuron result. The classification accuracy A in (135) of a single-output neural network exceeds the log-likelihood L in (136):

$$A \geqslant L + \ln 2\,. \tag{139}$$

The bound (139) holds because the inequality (138) gives

$$I\Big(a^t \geqslant \frac{1}{2}\Big) \geqslant \ln(2a^t) \tag{140}$$

for all target activation values $a^t \in (0, 1]$. But $a^t < \frac{1}{2}$ if and only if $1 - a^t > \frac{1}{2}$. So

$$I\Big(a^t < \frac{1}{2}\Big) = I\Big(1 - a^t > \frac{1}{2}\Big)\,. \tag{141}$$

Replace x with $1 - a^t$ in (138):

$$I\left(a^t < \frac{1}{2}\right) \geq \ln(2(1 - a^t)) \,. \tag{142}$$

Then the inequalities (140) and (142) give the accuracy bound (139):

$$\begin{aligned} A &= t\ I\left(a^t \geq \frac{1}{2}\right) + (1 - t)\ I\left(a^t < \frac{1}{2}\right) \\ &\geq t\ \ln(2a^t) + (1 - t)\ln(2(1 - a^t)) \qquad (143) \\ &= [t\ln(a^t) + (1 - t)\ln(1 - a^t)] + t\ln 2 + (1 - t)\ln 2 \qquad (144) \\ &= L + (t + 1 - t)\ln 2 \qquad (145) \\ &= L + \ln 2 \,. \qquad (146) \end{aligned}$$

Consider next the general case of K output neurons with *logistic* activations $a_1^t, \ldots, a_K^t$. Then the *total* complete accuracy $\mathbf{A}$ sums all true positives and all true negatives over all K logistic neurons in (135):

$$\mathbf{A} = \sum_{k=1}^{K} A_k \tag{147}$$

$$= \sum_{k=1}^{K} t_k\ I\left(a_k^t \geq \frac{1}{2}\right) + \sum_{k=1}^{K}(1 - t_k)\ I\left(a_k^t < \frac{1}{2}\right) \,. \tag{148}$$

Most classifier networks with softmax output neurons use a simpler measure of classification accuracy. They just count the number of true positive in n test runs and ignore the true negatives. This count $\mathbf{A}_{class}$ uses a form of the first sum in (148):

$$\mathbf{A}_{class} = \sum_{k=1}^{K} t_k\ I\left(a_k^t = \max_{1 \leq j \leq K} a_j^t\right) \tag{149}$$

because we assign an input pattern $\mathbf{x}$ to the kth decision class C_k if and only if the kth output softmax neuron has the largest activation $a_k^t(\mathbf{x})$ among the K output neurons. We measured classification accuracy with (149) in the MNIST classification simulations. The ratio of the n counts $\mathbf{A}_{class}$ to n trials gives this accuracy as a percentage or relative frequency.

The classification accuracy $\mathbf{A}_{class}$ has a simple probabilistic interpretation. Let $I(A)$ denote the indicator function of any measurable event A. Then the probability of A is just the expectation of its indicator function: $P(A) = E[I(A)]$. This result holds in general. It follows formally from the Radon–Nikodym Theorem of measure theory (Tucker, 2013).

Suppose that $\mathbf{x} \in C_k$ and that the target vector is binary with $t_k = 1$ and $t_j = 0$ if $k \neq j$. Then the average classification accuracy is just the probability that the kth output neuron "wins" the competition for activation given the input $\mathbf{x} \in C_k$:

$$E[\mathbf{A}_{class}(\mathbf{x})] = P(a_k^t(\mathbf{x}) \geq a_j^t(\mathbf{x}) \text{ for } 1 \leq j \leq K). \tag{150}$$

This result extends to the case where the target vector $\mathbf{t}$ is any K-length probability vector in $[0, 1]^K$:

$$E[\mathbf{A}_{class}(\mathbf{x})] = \sum_{k=1}^{K} t_k\ P(a_k^t(\mathbf{x}) = \max_{1 \leq j \leq K} a_j^t(\mathbf{x})). \tag{151}$$

So the expected classification accuracy is a probability mixture of the output "win" probabilities.

The next theorem uses a maximum-based corollary to Lemma 1 that applies to K softmax neurons:

$$I\left(a_k^t = \max_{1 \leq j \leq K} a_j^t\right) \geq \ln\left(\frac{a_k^t}{\max_{1 \leq j \leq K} a_j^t}\right). \tag{152}$$

This inequality holds because $a_k^t \geq \max_{1 \leq j \leq K} a_j^t$ for all $\mathbf{x}$ just in case $a_k^t = \max_{1 \leq j \leq K} a_j^t$. The sufficient condition of Lemma 1 holds because $\max_{1 \leq j \leq K} a_j^t \geq a_k^t \geq \frac{a_k^t}{e}$ since $e > 1$.

The next theorem derives separate likelihood bounds on the accuracy for logistic and softmax classifiers.

Theorem 6 (*Classification Accuracy-Likelihood Bound*)**.** *The classification accuracy* $\mathbf{A}_{class}$ *in* (149) *of a softmax-output neural network exceeds the log-likelihood* $L(\Theta)$ *in* (22)*:*

$$\mathbf{A}_{class} \geq L \,. \tag{153}$$

A network with K logistic output neurons has the bound

$$\mathbf{A} \geq L_{log} + K \ln 2 \tag{154}$$

for the logistic log-likelihood L_{log} *in* (62)*.*

NEM noise should also increase accuracy on average during training. Consider K softmax output neurons with 1-in-K encoding. NEM noise makes these binary target signals more probable in accord with (2) and Theorem 3. So we expect on average a slightly better "win" pattern for NEM-boosted activations a_j^{NEM}: $\max_{1 \leq j \leq K} a_j^{NEM} \geq \max_{1 \leq j \leq K} a_j$. So $\ln \frac{\max_{1 \leq j \leq K} a_j^{NEM}}{\max_{1 \leq j \leq K} a_j} \geq 0$. Then (A.32) gives $L^{NEM}(\Theta) \geq L(\Theta)$ on average for any target pdf $\{t_k\}$ because $L^{NEM}(\Theta) - \ln \max_{1 \leq j \leq K} a_j^{NEM} \geq L(\Theta) - \ln \max_{1 \leq j \leq K} a_j$ holds if and only if

$$L^{NEM}(\Theta) - L(\Theta) \geq \ln \max_{1 \leq j \leq K} a_j^{NEM} - \ln \max_{1 \leq j \leq K} a_j \tag{155}$$

$$= \ln \frac{\max_{1 \leq j \leq K} a_j^{NEM}}{\max_{1 \leq j \leq K} a_j} \geq 0 \,. \tag{156}$$

So on average: $L^{NEM}(\Theta) \geq L(\Theta)$.

Suppose last that $t_k = 1$. Then the NEM noise boost and (150) imply an average NEM accuracy benefit: $\mathbf{A}_{class}^{NEM} \geq \mathbf{A}_{class}$.

8. Pre-training with bidirectional associative memories (BAMs) or restricted Boltzmann machines (RBMs)

Restricted Boltzmann Machines (Hinton et al., 2006; Smolensky, 1986) are a special type of bidirectional associative memory (BAM) (Kosko, 1987, 1988, 1991). So they enjoy rapid convergence to a bidirectional fixed point for synchronous updating of all neurons in each of the two fields or layers of neurons. This convergence depends only on network parameters. It does not require a probabilistic interpretation or the use of stochastic convergence techniques. Bidirectional training also extends to unsupervised learning as we show below. It also extends to supervised backpropagation training (Adigun & Kosko, 2016, 2019a) and thus admits an EM and maximum-likelihood formulation (Adigun & Kosko, 2018).

The simplest BAM is a two-layer heteroassociative network that uses the synaptic connection matrix $\mathbf{W}$ on the forward pass of the neuronal signals from the lower layer to the higher layer. Its defining property is that it uses the adjoint or transpose matrix $\mathbf{W}^T$ on the backward pass from the higher layer to the lower layer. Its neural and synaptic nonlinearities can be quite general. Using both $\mathbf{W}$ and $\mathbf{W}^T$ this way *symmetrizes* the rectangular matrix $\mathbf{W}$. The lower layer is visible during the training of deep neural networks (Hinton et al., 2006) while the higher field is hidden. The general BAM Theorem ensures that any such matrix $\mathbf{W}$ is bidirectionally stable for threshold neurons as well for most continuous neurons. Logistic neurons satisfy the BAM Theorem because logistic activations are bounded and monotone nondecreasing. Fig. 11 shows convergence results for such a logistic BAM. The following results use the term RBM and BAM interchangeably.

The most striking fact about BAMs is their global stability. Every real rectangular matrix W is globally stable for a wide range

of nonlinear neuron activations (Kosko, 1991). Passing state vectors back and forth through W and its transpose W^T always and quickly leads to a two-step limit cycle and thus a bidirectional fixed point of the dynamical system. These nonlinear models range from simple thresholds to Cohen–Grossberg neural dynamics (Cohen & Grossberg, 1983; Grossberg, 1988) where the general activations need be only bounded and monotone nondecreasing. There is no need to appeal to far more complex notions of Gibbs-style Markov-chain stochastic convergence. Global stability follows in a simple deterministic manner for an extremely wide range of BAM systems.

We focus on neurons with soft thresholds such as logistic or hyperbolic-tangent activations. These activations behave in practice as on–off thresholds and yet have the continuous derivatives (6) and (8). The continuous BAM Theorem (Kosko, 1988, 1990, 1991) holds for such smooth activations for a wide range of Cohen–Grossberg nonlinear neural models (Cohen & Grossberg, 1983). But we can apply the simpler discrete BAM Theorem so long as the sigmoids are sufficiently steep to approximate a threshold. The proof uses the quadratic Lyapunov function $E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta) = -\sum_{i=1}^{I}\sum_{j=1}^{J} w_{ij}a_i^v a_j^h$ in (A.41).

Theorem 7 (*Discrete BAM Theorem*)**.** *Every connection matrix W is bidirectionally stable for visible and hidden neurons with sufficiently steep sigmoid activations.*

The Discrete BAM Theorem extends to a simple version of the Adaptive BAM Theorem (Kosko, 1987, 1988, 1991) if the weights w_{ij} adapt through simple Hebbian correlation learning:

$$w_{ij}(t+1) = w_{ij}(t) + a_i^v(t+1)a_j^h(t+1) \tag{157}$$

or just $\Delta w_{ij} = a_i^v a_j^v$. The weight update takes place *after* both the visible and hidden neurons have updated (they all update at the same time in the differential-equation versions (Kosko, 1987, 1988, 1991)). Then the update Δw_{ij} gives

$$\Delta E = -\sum_{i=1}^{I}\sum_{j=1}^{J} \Delta w_{ij}a_i^v a_i^h \tag{158}$$

$$= -\sum_{i=1}^{I}\sum_{j=1}^{J} (a_i^v a_i^h)^2 \tag{159}$$

$$< 0 \tag{160}$$

for any nonzero Hebbian weight change Δw_{ij} in (157). So a discrete version of the ABAM Theorem holds for simple Hebbian learning. We will see below how this general ABAM convergence helps explain convergence in contrastive-divergence learning since (A.41) gives the Hebbian-based gradient term

$$\frac{\partial E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)}{\partial w_{ij}} = -a_i^v a_j^h \,. \tag{161}$$

We next summarize two other BAM results that apply to pretraining in the deep learning of feedforward neural networks. We omit their proofs for reasons of space.

The first result is that the proof of the BAM convergence theorem still holds even if some or all of the neurons in one of the layers have bounded *non-monotone* activations such as Gaussian bell-curve activations. The only condition is that there be enough logistic neurons in the other field to overcome any positive energy changes $\Delta E_i > 0$ and still maintain the global energy decrease for the combined forward pass and backward pass: $\Delta E = \Delta E_{forward} + \Delta E_{backward} < 0$. This result is a type of "swamping" result because the negative energy changes from logistic neurons can always outweigh or swamp any positive changes from bounded non-monotonic neurons if there are enough logistic neurons. This result holds in particular if one of the layers consists of bounded Gaussian activations. The above Discrete ABAM Theorem also holds for such bounded activations. We show below how to inject pre-training NEM noise in this mixed logistic-Gaussian case.

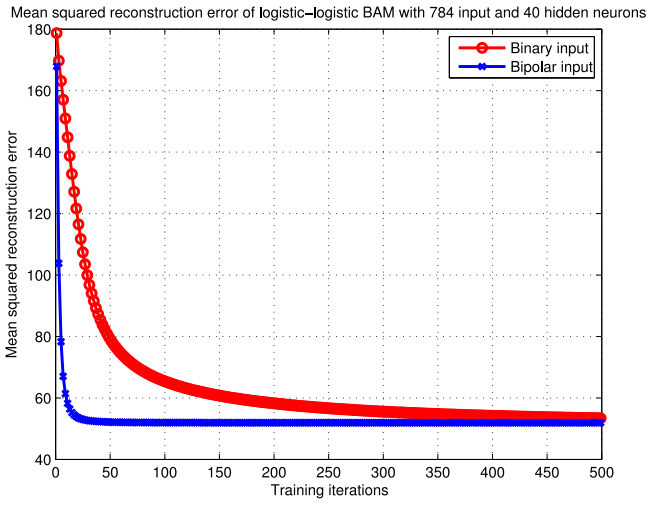

Fig. 11. Benefits of bipolar over binary coding in a logistic–logistic bidirectional associative memory (BAM). The two curves show the reconstruction squared error using binary coding in $[0, 1]$ versus bipolar coding $[-1, 1]$ of the input data in BAM encoding. Bipolar coding gives much faster convergence in terms of reconstruction squared error of the BAM input. The logistic–logistic BAM had 784 input logistic neurons and 40 hidden logistic neurons and trained on 1000 MNIST digit images. Bipolar encoding speeded convergence by more than an order of magnitude: Bipolar encoding of the input image pixels led to convergence in about 25 iterations. Training with binary encoding took nearly 500 iterations to converge.

The second result is that bipolar encoding improves BAM recall or convergence time when compared with binary coding. Bipolar encoding uses state vectors in the bipolar n-cube $[-1, 1]^n$ rather than in the binary n-cube $[0, 1]^n$. The simple bipolar transform $2x - 1$ for binary $x \in [0, 1]$ gives the order-of-magnitude speed-up in convergence in Fig. 11. Bipolar coding led to convergence in 25 bidirectional iterations of the logistic neurons. Binary encoding of the same MNIST images required nearly 500 iterations to converge. This result follows from the corresponding theorem in the Appendix of the original BAM paper (Kosko, 1988) and depends on the l^1 correlation structure of learning in bipolar spaces.

We next show that learning with contrastive divergence is also a special case of learning with generalized EM. Then we show how to noise-boost such RBM or BAM learning when all neurons are logistic and when one layer is logistic and the other layer is Gaussian.

We first show that the contrastive-divergence learning algorithm is also a special case of generalized EM. The next section shows how to noise-boost such two-layer BAMs or RBMs for pre-training. This involves defining the joint pdf $p(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)$ as a Gibbs or softmax function of the network energy $E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)$.

Consider again a BAM or RBM with I visible neurons and J hidden neurons. We can also denote the visible or input layer as the input field F_X and the hidden layer as the adjoining field F_H (Kosko, 1991). Let a_i^v and a_j^h denote the respective activations of the ith visible neuron and the jth hidden neuron:

$$a_i^v = a_i^v\left(\sum_{j=1}^{J} w_{ij}a_j^h + \sum_{j=1}^{J} a_j a_j^h\right) \tag{162}$$

$$a_j^h = a_j^h \left(\sum_{i=1}^{I} w_{ij} a_i^v + \sum_{i=1}^{I} b_i a_i^v \right) \tag{163}$$

for scaling constants a_j and b_i. Define the inputs $\tilde{x}_i$ and $\tilde{h}_j$ as

$$\tilde{x}_i = \sum_{j=1}^{J} w_{ij} a_j^h + \sum_{j=1}^{J} a_j a_j^h \tag{164}$$

$$\tilde{h}_j = \sum_{i=1}^{I} w_{ij} a_i^v + \sum_{i=1}^{I} b_i a_i^v \ . \tag{165}$$

Then we can write the visible and hidden activations more compactly as $a_i^v(\tilde{x}_i)$ and $a_j^h(\tilde{h}_j)$.

We focus on logistic and Gaussian activations because they are the most common in pre-training and in many other applications. This gives rise to two types of connected BAM fields or layers. The first type has logistic or other sigmoid neurons at each layer. Its probability structure is a Bernoulli(visible)–Bernoulli(hidden) BAM. The second type has Gaussian neurons at the lower or visible layer but logistic neurons at the upper or hidden layer. Its probability structure is a Gaussian(visible)–Bernoulli(hidden) BAM. We well embed these networks in the EM framework and then noise-boost them separately.

The probabilistic structure of the BAM or RBM depends on the energy $E(\mathbf{a^v}, \mathbf{a^h}|\Theta)$ of the two-layer network. Then the joint pdf of activation vector $\mathbf{a^v}$ by way of the input $\mathbf{x}$ and hidden activation $\mathbf{a^h}$ is the Gibbs or softmax density as in (10):

$$p(\mathbf{a^v}, \mathbf{a^h}|\Theta) = \frac{\exp(-E(\mathbf{a^v}, \mathbf{a^h}|\Theta))}{Z(\Theta)} \tag{166}$$

with partition function

$$Z(\Theta) = \sum_{\mathbf{a^v}} \sum_{\mathbf{a^h}} \exp(-E(\mathbf{a^v}, \mathbf{a^h}|\Theta)) \ . \tag{167}$$

Integrals can replace sums for continuous variables in the above partition function $Z(\Theta)$.

The energy function $E(\mathbf{a^v}, \mathbf{a^h}|\Theta)$ depends in turn on the type of activations in the visible field and in the hidden field. A Bernoulli(visible)–Bernoulli(hidden) BAM or RBM has logistic conditional pdfs at both the hidden and visible layers. So it has the following BAM energy or Lyapunov function (Kosko, 1987, 1988, 1991) that slightly generalizes (A.41):

$$E(\mathbf{a^v}, \mathbf{a^h}|\Theta) = -\sum_{i=1}^{I} \sum_{j=1}^{J} w_{ij} a_i^v(\tilde{x}_i) a_j^h(\tilde{h}_j) - \sum_{i=1}^{I} b_i a_i^v(\tilde{x}_i) - \sum_{j=1}^{J} a_j a_j^h(\tilde{h}_j) \tag{168}$$

where w_{ij} is the connection weight between the ith visible and jth hidden neuron, b_i is the bias for the ith visible neuron, and a_j is the bias for the jth hidden neuron.

A Gaussian(visible)–Bernoulli(hidden) BAM or RBM has Gaussian conditional pdfs at the visible layer but logistic conditional pdfs at the hidden layer. So its energy function (Hinton et al., 2006; Hinton & Salakhutdinov, 2006) includes an extra quadratic term:

$$E(\mathbf{a^v}, \mathbf{a^h}|\Theta) = -\sum_{i=1}^{I} \sum_{j=1}^{J} w_{ij} a_i^v(\tilde{x}_i) a_j^h(\tilde{h}_j) + \frac{1}{2} \sum_{i=1}^{I} (a_i^v(\tilde{x}_i) - b_i)^2 - \sum_{j=1}^{J} a_j a_j^h(\tilde{h}_j) \ . \tag{169}$$

A key fact for learning is that the weight w_{ij} appears expressly only in the quadratic form in both energy functions (168) and (169). This gives the same Hebbian-based gradient term $-a_i^v a_j^h$ as in (157) when we differentiate either (168) or (169):

$$\frac{\partial E(\mathbf{a^v}, \mathbf{a^h}|\Theta)}{\partial w_{ij}} = -a_i^v(\tilde{x}_i) a_j^h(\tilde{h}_j). \tag{170}$$

The overall deep or multilayer neural network uses RBMs or BAMs as an inter-layer building blocks. The system finds maximum-likelihood estimates for the BAM's or RBM's parameters and then stacks the resulting BAMs or RBMs on top of each other. This uses a form of what Hinton has called *contrastive divergence* learning (Hinton et al., 2006; Hinton & Salakhutdinov, 2006). Then BP trains this pre-trained neural network. We show now how learning with contrastive divergence is just generalized EM for logistic or Gaussian-logistic BAMs or RBMs.

Contrastive divergence approximates the ML training of the RBM or BAM parameters for $\ln p(\mathbf{x}|\Theta)$. See Bengio (2009) for a review of the technique. Gradient ascent can iteratively solve this simplified two-layer maximum-likelihood optimization (15) for the optimal parameters Θ^* as we have shown above when casting BP as maximum likelihood.

We estimate the same matrix weights w_{ij} in the quadratic forms of the network energies (168) or (169) because these terms are the same for a Bernoulli–Bernoulli and Gaussian–Bernoulli BAM or RBM. A marginalization argument shows that the contrastive-divergence gradient estimate of the log-likelihood $\ln p(\mathbf{a^v}, \mathbf{a^h}|\Theta)$ with respect to the weight w_{ij} has the Hebbian difference form (Hinton et al., 2006; Hinton & Salakhutdinov, 2006):

$$\frac{\partial \ln p(\mathbf{a^v}, \mathbf{a^h}|\Theta)}{\partial w_{ij}} = \mathbb{E}_{\mathbf{a^h}|\mathbf{a^v},\Theta}\{a_i^v a_j^h\} - \mathbb{E}_{\mathbf{a^v},\mathbf{a^h}|\Theta}\{a_i^v a_j^h\} \ . \tag{171}$$

Then the learning law for w_{ij} becomes

$$w_{ij}^{n+1} = w_{ij}^n + \eta \left(\mathbb{E}_{\mathbf{a^h}|\mathbf{a^v},\Theta^n}\{a_i^v a_j^h\} - \mathbb{E}_{\mathbf{a^v},\mathbf{a^h}|\Theta^n}\{a_i^v a_j^h\} \right) \tag{172}$$

where again $\eta > 0$ is the learning rate or sequence of such rates.

Learning stops in (172) when the Hebbian averages are equal for the hidden posterior $p(\mathbf{a^h}|\Theta)$ and the joint or complete pdf $p(\mathbf{a^v}, \mathbf{a^h}|\Theta)$: $\mathbb{E}_{\mathbf{a^h}|\mathbf{a^v},\Theta}\{a_i^v a_j^h\} = \mathbb{E}_{\mathbf{a^v},\mathbf{a^h}|\Theta}\{a_i^v a_j^h\}$.

This stopping rule corresponds roughly to a Hebbian ABAM equilibrium from (159)–(161) when the encoding $w_{ij} = a_i^v a_j^h$ holds at a local energy minimum. The ABAM converges quickly in the discrete case and exponentially quickly in the continuous case (Kosko, 1987, 1991). This deterministic global-stability result avoids the need to invoke Gibbs sampling or other forms of Markov chain Monte Carlo and their often extensive burn-in runs before they achieve stochastic equilibrium. So rapid ABAM convergence may explain the observed "surprising empirical result" that a trivial Markov chain of just one step "often gives good results" (Bengio, 2009).

We can easily compute the pdf $p(\mathbf{a^h}|\mathbf{a^v}, \Theta^n)$ for the BAM or RBM because there are no connections between any two hidden neurons or between any two visible neurons in these simple BAM models (unlike the more general case of BAM fields of winner-take-all or other competitive neurons (Kosko, 1991)). This pdf gives the expectation $\mathbb{E}_{\mathbf{a^h}|\mathbf{a^v},\Theta^n}\{a_i^v a_j^h\}$. But we cannot so easily compute the joint pdf $p(\mathbf{a^v}, \mathbf{a^h}|\Theta^n)$ because of the partition function $Z(\Theta)$ in (167). Contrastive divergence (CD) (Bengio, 2009; Hinton et al., 2006) approximates $Z(\Theta)$ through activations that derive from a forward and a backward pass in the BAM or RBM.

The next theorem shows that the contrastive-divergence learning law (172) is also special case of the GEM algorithm (93). This result holds because of the Gibbs ratio form of the two-layer-network density function $p(\mathbf{a^v}, \mathbf{a^h}|\Theta)$ in (166).

Theorem 8 (*Contrastive-Divergence Learning in a BAM or RBM is Generalized EM*). *The contrastive-divergence update equation (172) for the differentiable Gibbs likelihood function $p(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)$ in (166) at epoch n*

$$w_{ij}^{n+1} = w_{ij}^{n} + \eta \frac{\partial \ln p(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)}{\partial w_{ij}}\Big|_{\Theta=\Theta^n} \tag{173}$$

equals the GEM update equation at epoch n

$$w_{ij}^{n+1} = w_{ij}^{n} + \eta \frac{\partial Q\left(\Theta|\Theta^n\right)}{\partial w_{ij}}\Big|_{\Theta=\Theta^n} . \tag{174}$$

The next section shows that NEM noise can speed up the ML estimation involved in pre-training BAMs or RBMs.

9. Noise-boosting contrastive divergence in BAMs and RBMs

Theorem 8 lets us inject NEM noise $\mathbf{n}$ into the input activations $\mathbf{a}^{\mathbf{v}}$. Theorem 2 implies that the BAM or RBM enjoys a NEM additive-noise benefit if it satisfies the NEM inequality

$$\mathbb{E}_{\mathbf{a}^{\mathbf{v}},\mathbf{a}^{\mathbf{h}},\mathbf{n}|\Theta^*}\left\{\ln \frac{p(\mathbf{a}^{\mathbf{v}}+\mathbf{n}, \mathbf{a}^{\mathbf{h}}|\Theta^n)}{p(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta^n)}\right\} \geqslant 0 . \tag{175}$$

The noisy complete data likelihood is

$$p(\mathbf{a}^{\mathbf{v}}+\mathbf{n}, \mathbf{a}^{\mathbf{h}}|\Theta^n) = \frac{\exp(-E(\mathbf{a}^{\mathbf{v}}+\mathbf{n}, \mathbf{a}^{\mathbf{h}}|\Theta^n))}{Z_{\mathbf{n}}(\Theta^n)} \tag{176}$$

where $Z_{\mathbf{n}}(\Theta)$ is the noisy partition function

$$Z_{\mathbf{n}}(\Theta) = \sum_{\mathbf{a}^{\mathbf{v}}} \sum_{\mathbf{a}^{\mathbf{h}}} \exp(-E(\mathbf{a}^{\mathbf{v}}+\mathbf{n}, \mathbf{a}^{\mathbf{h}}|\Theta)) \tag{177}$$

from (166).

So a NEM noise benefit holds at epoch n if

$$\mathbb{E}_{\mathbf{a}^{\mathbf{v}},\mathbf{a}^{\mathbf{h}},\mathbf{n}|\Theta^*}\left\{\ln \frac{\exp(-E(\mathbf{a}^{\mathbf{v}}+\mathbf{n}, \mathbf{a}^{\mathbf{h}}|\Theta^n))}{\exp(-E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta^n))}\right\} \geqslant \mathbb{E}_{\mathbf{n}|\Theta^*}\left\{\ln \frac{Z_{\mathbf{n}}(\Theta^n)}{Z(\Theta^n)}\right\} \tag{178}$$

because neither the partition function $Z(\Theta)$ nor its noisy version $Z_{\mathbf{n}}(\Theta)$ depend on the input or hidden activations. This gives the key BAM/RBM noise-benefit inequality:

$$\mathbb{E}_{\mathbf{a}^{\mathbf{v}},\mathbf{a}^{\mathbf{h}},\mathbf{n}|\Theta^*}\left\{E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta^n) - E(\mathbf{a}^{\mathbf{v}}+\mathbf{n}, \mathbf{a}^{\mathbf{h}}|\Theta^n)\right\} \geqslant \mathbb{E}_{\mathbf{n}|\Theta^*}\left[\ln Z_{\mathbf{n}}(\Theta^n)\right] - \ln Z(\Theta^n) . \tag{179}$$

A practical heuristic takes the lower bound in (179) as zero. This gives a simple inequality that the NEM noise $\mathbf{n}$ must satisfy on average:

$$E(\mathbf{a}^{\mathbf{v}}+\mathbf{n}, \mathbf{a}^{\mathbf{h}}|\Theta^n) \leqslant E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta^n). \tag{180}$$

The BAM/RBM noise-benefit condition (179) holds for arbitrary probabilistic neurons if the network probability has the Gibbs ratio structure (166). The next theorem states that the important special case of a logistic–logistic (Bernoulli–Bernoulli) BAM or RBM defines a separating NEM hyperplane in noise space. A simple heuristic also takes its lower bound as zero.

Theorem 9 (*Logistic–Logistic Hyperplane Noise Benefit*). *The NEM positivity condition holds for a Bernoulli–Bernoulli (logistic–logistic) BAM or RBM at iteration n if*

$$\mathbb{E}_{\mathbf{a}^{\mathbf{v}},\mathbf{a}^{\mathbf{h}},\mathbf{n}|\Theta^*}\left\{\mathbf{n}^T(\mathbf{W}\mathbf{a}^{\mathbf{h}}+\mathbf{b})\right\} \geqslant \mathbb{E}_{\mathbf{n}|\Theta^*}\left[\ln Z_{\mathbf{n}}(\Theta^n)\right] - \ln Z(\Theta^n) . \tag{181}$$

Fig. 6 shows that injecting NEM noise into a BAM in accord with Theorem 9 reduced the MNIST training-set squared error by 16% compared with noiseless training. Fig. 7 shows that injecting blind noise into the BAM produced no benefit.

The same argument holds for the Gaussian-logistic energy function in (169) but gives a *hyperspherical* NEM separation condition for a Gaussian–Bernoulli BAM or RBM.

Theorem 10 (*Gaussian-Logistic Spherical Noise Benefit*). *The NEM positivity condition holds for training a Gaussian–Bernoulli BAM or RBM at iteration n if*

$$\mathbb{E}_{\mathbf{a}^{\mathbf{v}},\mathbf{a}^{\mathbf{h}},\mathbf{n}|\Theta^*}\left\{\frac{1}{2}\|\mathbf{n}\|^2 - \mathbf{n}^T(\mathbf{W}\mathbf{a}^{\mathbf{h}}+\mathbf{b}-\mathbf{x})\right\} \leqslant \ln Z(\Theta^n) - \mathbb{E}_{\mathbf{n}|\Theta^*}\left[\ln Z_{\mathbf{n}}(\Theta^n)\right] . \tag{182}$$

The NEM inequality in (182) bisects the noise space. The bisecting surface itself is a hypersphere. This hyperspherical NEM sufficient condition resembles that of injecting noise into the output layer of a regression network as in Theorem 4.

10. Simulation results

The classifier simulations used 1000 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. The classifier networks had 5 layers. There were 40 logistic neurons in each of the three hidden layers. There were 10 softmax neurons in the output layer for classifying the 10 categories of handwritten digits. We modified the MATLAB code in (Hinton) to inject noise during EM-backpropagation training of a neural network.

The simulations used 10 Monte Carlo samples for approximating the Q-function in the 10-class classification network. Fig. 2 shows the NEM noise benefit for cross-entropy training of a feedforward neural classifier. The NEM version produced an 18% median decrease in cross entropy per iteration compared with noiseless BP training. Fig. 3 shows that adding blind noise instead of NEM noise only gave a miniscule improvement of 1.7% in cross entropy over the noiseless EM–BP algorithm. Fig. 4 shows that NEM noise injection gave a 15% median improvement in the per-iteration classification error rate for the training set and a 10% improvement for the test set at the optimal noise variance of 0.42. Fig. 5 shows that this noise benefit disappears upon using blind noise in place of NEM noise.

Fig. 10 shows the effects of NEM noise injection in the hidden layers as well as the output layers of a regression network and an MNIST-trained classifier network. The regression network approximated $f(x) = \sin x$ with 18,000 random training samples from the domain $[0, 2\pi]$. The three-layer network used just one identity neuron in the output layer and 10 logistic neurons in the hidden layer. Adding NEM noise to the output and hidden neurons substantially reduced the average squared error compared with adding NEM noise to just the output neuron alone or adding no noise at all. The NEM noise for the output identity neuron used the hyperspherical NEM condition from Theorem 4. The NEM noise for the hidden logistic neurons used the hyperplane NEM condition from Theorem 5. This total NEM noise injection led to almost complete convergence after just 4 iterations.

Fig. 10 also shows the effects of NEM noise injection both in the output layer and in the hidden layers of a 3-hidden-layer classifier network with 40 hidden logistic neurons each. NEM noise gave a 60.44% relative reduction in the per-iteration training-set cross-entropy compared with noiseless BP. It gave a 54.39% relative reduction in the per-iteration test-set cross-entropy for NEM compared with noiseless BP.

Fig. 11 shows that logistic neurons with bipolar values in $[-1, 1]$ speeded up BAM convergence by more than an order of magnitude over ordinary binary logistic neurons with values in $[0, 1]$ when training on the MNIST test images. Each BAM used 784 visible logistic neurons and 40 hidden logistic neurons. The bipolar BAM converged in about 25 iterations. The binary BAM converged in about 500 iterations.

11. Conclusions

The backpropagation algorithm is a special case of the generalized EM algorithm. That should not be surprising because backpropagation is a form of iterative maximum-likelihood estimation and because EM generalizes iterative maximum likelihood to the case of hidden variables. Proper noise injection speeds average backpropagation convergence because it speeds average EM convergence. This leads to several sufficient conditions that guarantee a BP speed-up for classification and regression networks as well as for logistic networks. These noise benefits still hold for regularized networks.

Similar sufficient conditions hold for a noise benefit in pretraining neural networks based on the NEM theorem. Basic contrastive-divergence learning is also a special case of generalized EM if the probability density of the two-layer network has the form of a Gibbs density based on the network energy. The convergence involved between two stacked layers is the global stability that the (adaptive) BAM convergence theorem ensures. This holds if both layers use sigmoidal neurons or if they both use logistic neurons. It still holds if one of the layers uses Gaussian or other nonmonotonic neurons so long as there are enough logistic neurons in the other layer. Correlation encoding properties of BAMs show that bipolar neurons give better recall performance on average than do binary neurons.

In sum: The basic gradient identity from Theorem 1

$$\nabla_\Theta \ln p(\mathbf{y}|\mathbf{x}, \Theta^n) = \nabla_\Theta Q(\Theta^n|\Theta^n) \tag{183}$$

applies to any iterative maximum-likelihood scheme such as convolutional and recurrent classification or regression (Adigun & Kosko, 2017; Audhkhasi et al., 2016). Then a corresponding NEM noise benefit will also apply.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix. Proofs of theorems

This appendix gives the complete proofs of all theorems except the quoted Theorem 2 from Osoba et al. (2011b) and Osoba et al. (2013).

Theorem 1 (*Backpropagation as the GEM Algorithm*)**.** *The backpropagation update equation for a differentiable likelihood function* $p(\mathbf{y}|\mathbf{x}, \Theta)$ *at epoch* n

$$\Theta^{n+1} = \Theta^n + \eta \nabla_\Theta \ln p(\mathbf{y}|\mathbf{x}, \Theta)\Big|_{\Theta=\Theta^n} \tag{95}$$

equals the GEM update equation at epoch n

$$\Theta^{n+1} = \Theta^n + \eta \nabla_\Theta Q(\Theta|\Theta^n)\Big|_{\Theta=\Theta^n} \tag{96}$$

where GEM uses the differentiable Q-function

$$Q(\Theta|\Theta^n) = \mathbb{E}_{\mathbf{h}|\mathbf{y},\mathbf{x},\Theta^n}\left\{\ln p(\mathbf{y}, \mathbf{h}|\mathbf{x}, \Theta)\right\} . \tag{97}$$

Proof. The proof rests on the above EM equality (77) for the network log-likelihood:

$$\ln p(\mathbf{y}|\mathbf{x}, \Theta) = Q(\Theta|\Theta^n) - \mathbb{E}_{\mathbf{h}|\mathbf{y},\mathbf{X},\Theta^n}\{\ln p(\mathbf{h}, \mathbf{y}|\mathbf{X}, \Theta)\} \tag{A.1}$$

$$= Q(\Theta|\Theta^n) + H(\Theta|\Theta^n) \tag{A.2}$$

for differentiable cross entropy $H(\Theta|\Theta^n)$ from (78).

Taking gradients with respect to the network parameter vector Θ gives

$$\nabla_\Theta \ln p(\mathbf{y}|\mathbf{x}, \Theta) = \nabla_\Theta Q(\Theta|\Theta^n) + \nabla_\Theta H(\Theta|\Theta^n) . \tag{A.3}$$

Then the theorem follows if we can show that the null gradient $\nabla_\Theta H(\Theta|\Theta^n) = \mathbf{0}$ holds when $\Theta = \Theta^n$. But the entropy inequality (82) states that

$$H(\Theta|\Theta^n) \geqslant H(\Theta^n|\Theta^n) \tag{A.4}$$

for all Θ. Thus Θ^n *minimizes* $H(\Theta|\Theta^n)$. So

$$\nabla_\Theta H(\Theta|\Theta^n) = \mathbf{0} \tag{A.5}$$

holds at $\Theta = \Theta^n$ from Fermat's Theorem for gradients.

Putting (A.5) in (A.3) gives the desired gradient equality at $\Theta = \Theta^n$:

$$\nabla_\Theta \ln p(\mathbf{y}|\mathbf{x}, \Theta)\Big|_{\Theta=\Theta^n} = \nabla_\Theta Q(\Theta|\Theta^n)\Big|_{\Theta=\Theta^n} . \tag{A.6}$$

So the BP and GEM update equations are identical at each iteration n. ■

Theorem 3 (*Hyperplane Noise Benefit for Injecting Noise in a Classifier Network's Output Layer*)**.** *The NEM positivity condition (114) holds for maximum-likelihood training of a classifier neural network with output Gibbs or softmax activations if the following average hyperplane condition holds at iteration* n*:*

$$\mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\left\{\mathbf{n}^T \ln \mathbf{a}^t\right\} \geqslant 0 . \tag{119}$$

The NEM condition (114) also holds for injecting noise in output logistic *neurons if*

$$\mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\{\mathbf{n}^T \ln \mathbf{a}^t\} \geqslant \mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\{\mathbf{n}^T \ln(\mathbf{1} - \mathbf{a}^t)\} . \tag{120}$$

Proof. We add the noise vector $\mathbf{n}$ to the output target 1-in-K encoding vector $\mathbf{t}$. Then expanding the EM-based *complete* likelihood ratio in the NEM sufficient condition (114) gives the likelihood ratio as a simple product of exponentiated output activations because the output neurons are conditionally independent:

$$\frac{p(\mathbf{t}+\mathbf{n}, \mathbf{h}|\mathbf{x}, \Theta)}{p(\mathbf{t}, \mathbf{h}|\mathbf{x}, \Theta)} = \frac{p(\mathbf{t}+\mathbf{n}, \mathbf{h}|\mathbf{x}, \Theta)p(\mathbf{h}|\mathbf{x}, \Theta)}{p(\mathbf{h}|\mathbf{x}, \Theta)p(\mathbf{t}, \mathbf{h}|\mathbf{x}, \Theta)} \tag{A.7}$$

$$= \frac{p(\mathbf{t}+\mathbf{n}|\mathbf{h}, \mathbf{x}, \Theta)}{p(\mathbf{t}|\mathbf{h}, \mathbf{x}, \Theta)} \tag{A.8}$$

$$= \frac{\prod_{k=1}^K (a_k^t)^{t_k+n_k}}{\prod_{k=1}^K (a_k^t)^{t_k}} \tag{A.9}$$

$$= \prod_{k=1}^K \frac{(a_k^t)^{t_k+n_k}}{(a_k^t)^{t_k}} \tag{A.10}$$

$$= \prod_{k=1}^K (a_k^t)^{n_k} . \tag{A.11}$$

So the NEM positivity condition (114) becomes

$$\mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\left\{\ln \prod_{k=1}^K (a_k^t)^{n_k}\right\} \geqslant 0 \tag{A.12}$$

or

$$\mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\left\{\sum_{k=1}^K n_k \ln a_k^t\right\} \geqslant 0 . \tag{A.13}$$

The vector version has the inner-product form

$$\mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\{\mathbf{n}^T \ln \mathbf{a}^t\} \geqslant 0 \tag{A.14}$$

if $\ln \mathbf{a}^t$ is the vector of the output neuron log-activations.

The same argument gives a related NEM-hyperplane result for output logistic neurons with noise-injected complete likelihood $p_{log}(\mathbf{t}+\mathbf{n}, \mathbf{h}|\mathbf{x}, \Theta)$ using (60):

$$\frac{p_{log}(\mathbf{t}+\mathbf{n}, \mathbf{h}|\mathbf{x}, \Theta)}{p_{log}(\mathbf{t}, \mathbf{h}|\mathbf{x}, \Theta)} = \prod_{k=1}^{K} \frac{(a_k^t)^{t_k+n_k}(1-a_k^t)^{1-t_k-n_k}}{(a_k^t)^{t_k}(1-a_k^t)^{1-t_k}} \tag{A.15}$$

$$= \prod_{k=1}^{K} (a_k^t)^{n_k}(1-a_k^t)^{-n_k} \ . \tag{A.16}$$

Then taking logarithms and NEM expectations gives the more complex hyperplane inequality

$$\mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\{\mathbf{n}^T \ln \mathbf{a}^t\} \geqslant \mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\{\mathbf{n}^T \ln (\mathbf{1}-\mathbf{a})^t\} \ . \tag{A.17}$$

■

Theorem 4 (*Regression Hypersphere Noise Benefit*)**.** *The NEM positivity condition (114) holds at iteration n for maximum-likelihood training of a regression neural network with Gaussian target vector* $\mathbf{t} \sim \mathcal{N}(\mathbf{t}|\mathbf{a}^t, \mathbf{I})$ *if*

$$\mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|,\mathbf{x},\Theta^*}\left\{\|\mathbf{n}-\mathbf{a}^t+\mathbf{t}\|^2 - \|\mathbf{a}^t-\mathbf{t}\|^2\right\} \leqslant 0 \tag{122}$$

where $\|.\|$ *is the Euclidean vector norm.*

Proof. Add the noise vector $\mathbf{n}$ to the K output neurons $\mathbf{t}$. So the noise $\mathbf{n}$ enters the regression likelihood as $p_{reg}(\mathbf{t}+\mathbf{n}|\mathbf{h}, \mathbf{x}, \Theta)$ from (48). Then the corresponding complete likelihood ratio in the NEM sufficient condition (114) becomes

$$\frac{p_{reg}(\mathbf{t}+\mathbf{n}, \mathbf{h}|\mathbf{x}, \Theta)}{p_{reg}(\mathbf{t}, \mathbf{h}|\mathbf{x}, \Theta)} = \frac{p_{reg}(\mathbf{t}+\mathbf{n}, \mathbf{h}|\mathbf{x}, \Theta)p_{reg}(\mathbf{h}|\mathbf{x}, \Theta)}{p_{reg}(\mathbf{h}|\mathbf{x}, \Theta)p_{reg}(\mathbf{t}, \mathbf{h}|\mathbf{x}, \Theta)} \tag{A.18}$$

$$= \frac{p_{reg}(\mathbf{t}+\mathbf{n}|\mathbf{h}, \mathbf{x}, \Theta)}{p_{reg}(\mathbf{t}|\mathbf{h}, \mathbf{x}, \Theta)} \tag{A.19}$$

$$= \frac{\mathcal{N}(\mathbf{t}+\mathbf{n}|\mathbf{a}^t, \mathbf{I})}{\mathcal{N}(\mathbf{t}|\mathbf{a}^t, \mathbf{I})} \tag{A.20}$$

from (51)

$$= \exp\left(\frac{1}{2}\left[\|\mathbf{t}-\mathbf{a}^t\|^2 - \|\mathbf{t}+\mathbf{n}-\mathbf{a}^t\|^2\right]\right) \tag{A.21}$$

from (49) for Euclidean norm $\|\mathbf{z}\|^2 = z_1^2 + \cdots + z_d^2$. Take the logarithm and expectation to get the spherical NEM condition

$$\mathbb{E}_{\mathbf{t},\mathbf{h},\mathbf{n}|,\mathbf{x},\Theta^*}\left\{\|\mathbf{n}-\mathbf{a}^t+\mathbf{t}\|^2 - \|\mathbf{a}^t-\mathbf{t}\|^2\right\} \leqslant 0 \ . \quad ■ \tag{A.22}$$

Theorem 5 (*NEM Noise in Hidden Logistic Neurons*)**.** *NEM noise* $\mathbf{n}$ *boosts a given hidden layer of logistic neurons if the injected noise satisfies the NEM likelihood inequality*

$$\mathbb{E}_{\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\{\mathbf{n}^T \ln \mathbf{a}^h\} \geqslant \mathbb{E}_{\mathbf{h},\mathbf{n}|\mathbf{x},\Theta^*}\{\mathbf{n}^T \ln (\mathbf{1}-\mathbf{a})^h\} \ . \tag{134}$$

for the hidden-layer activation vector $\mathbf{a}^h$ *with the logistic layer-likelihood structure (60)–(62).*

Proof. The general likelihood factorization (133) gives the total network log-likelihood as the respective sum of layer log-likelihoods at iteration n:

$$L(\mathbf{x}) = L(\mathbf{y}|\mathbf{x}) + L(\mathbf{h}_k|\mathbf{x}) + \cdots + L(\mathbf{h}_1|\mathbf{x}) \tag{A.23}$$

where $L(\mathbf{h}_k|\mathbf{x}) = \ln p(\mathbf{h}_k|\mathbf{h}_{k-1}, \ldots, \mathbf{h}_1, \mathbf{x}, \Theta^n)$. This additive structure allows NEM-noise injection at all layers or at any subset of layers at the nth training iteration.

We assume for simplicity that the hidden layer in question is the kth hidden layer. It has likelihood function $p(\mathbf{h}_k|\mathbf{h}_{k-1}, \ldots, \mathbf{h}_1, \mathbf{x}, \Theta^n)$ that we write in abbreviated form $p(\mathbf{h}_k|\mathbf{h}, \mathbf{x}, \Theta)$ where $\mathbf{h}$ describes all lower hidden layers $\mathbf{h}_{k-1}, \ldots, \mathbf{h}_1$. The NEM structure still holds for the likelihood ratio because

$$\frac{p(\mathbf{h}_k+\mathbf{n}|\mathbf{h}, \mathbf{x}, \Theta)}{p(\mathbf{h}_k|\mathbf{h}, \mathbf{x}, \Theta)} = \frac{p(\mathbf{h}_k+\mathbf{n}, \mathbf{h}|\mathbf{x}, \Theta)p(\mathbf{h}|\mathbf{x}, \Theta)}{p(\mathbf{h}|\mathbf{x}, \Theta)p(\mathbf{h}_k, \mathbf{h}|\mathbf{x}, \Theta)} \tag{A.24}$$

$$= \frac{p(\mathbf{h}_k+\mathbf{n}, \mathbf{h}|\mathbf{x}, \Theta)}{p(\mathbf{h}_k, \mathbf{h}|\mathbf{x}, \Theta)} . \tag{A.25}$$

Then the logistic likelihood (60) gives the noise-injected complete likelihood ratio as

$$\frac{p(\mathbf{h}_k+\mathbf{n}, \mathbf{h}|\mathbf{x}, \Theta)}{p(\mathbf{h}_k, \mathbf{h}|\mathbf{x}, \Theta)} = \prod_{j=1}^{J} \frac{(a_j^{h_k})^{h_{k_j}+n_j}(1-a_j^{h_k})^{1-h_{k_j}-n_j}}{(a_j^{h_k})^{h_{k_j}}(1-a_j^{h_k})^{1-h_{k_j}}} \tag{A.26}$$

$$= \prod_{j=1}^{J} (a_j^{h_k})^{n_j}(1-a_j^{h_k})^{-n_j} \ . \tag{A.27}$$

The result now follows from the basic NEM Theorem by taking logarithms and then taking expectations with respect to the likelihood $p(\mathbf{h}, \mathbf{n}|\mathbf{x}, \Theta^*)$. ■

Lemma 1. *Let* $x \in (0, 1]$ *and* $y \in (0, 1]$*. Then*

$$I\left(x \geqslant y\right) \geqslant \ln\left(\frac{x}{y}\right) \tag{137}$$

if $y \geqslant x/e$*.*

Proof. Suppose first that $x \in (0, y)$. Then $x < y$. So the indicator event does not occur: $I\left(x \geqslant y\right) = 0$. But $\ln(\frac{x}{y}) \leqslant 0$ because $x < y$ and both x and y are positive. So $I\left(x \geqslant y\right) \geqslant \ln(\frac{x}{y})$ holds in this case. Suppose next that $x \in [y, 1]$ for $y > 0$. Then $x \geqslant y$ holds. So $I\left(x \geqslant y\right) = 1$ holds. But $\ln(\frac{x}{y}) \leqslant 1$ holds just in case $y \geqslant x/e$. Then $I\left(x \geqslant y\right) \geqslant \ln(\frac{x}{y})$ holds in this case as well. ■

Theorem 6 (*Classification Accuracy-Likelihood Bound*)**.** *The classification accuracy* $\mathbf{A}_{class}$ *in (149) of a softmax-output neural network exceeds the log-likelihood* $L(\Theta)$ *in (22):*

$$\mathbf{A}_{class} \geqslant L \ . \tag{153}$$

A network with K logistic output neurons has the bound

$$\mathbf{A} \geqslant L_{log} + K \ln 2 \tag{154}$$

for the logistic log-likelihood L_{log} *in (62).*

Proof. The K softmax neurons obey $0 < \max_{1 \leqslant j \leqslant K} a_j^t \leqslant 1$. So $\ln \max_{1 \leqslant j \leqslant K} a_j^t \leqslant 0$. Combine this inequality with the inequality in (152):

$$\mathbf{A}_{class} = \sum_{k=1}^{K} t_k \ I\left(a_k^t = \max_{1 \leqslant j \leqslant K} a_j^t\right) \tag{A.28}$$

$$\geqslant \sum_{k=1}^{K} t_k \ln\left(\frac{a_k^t}{\max_{1 \leqslant j \leqslant K} a_j^t}\right) \tag{A.29}$$

$$= \sum_{k=1}^{K} t_k \ln a_k^t - \sum_{k=1}^{K} t_k \ln \max_{1 \leqslant j \leqslant K} a_j^t \tag{A.30}$$

$$= L(\Theta) - (\ln \max_{1 \leqslant j \leqslant K} a_j^t) \sum_{k=1}^{K} t_k \tag{A.31}$$

$$= L(\Theta) - \ln \max_{1 \leqslant j \leqslant K} a_j^t \tag{A.32}$$

$$\geqslant L(\Theta) \tag{A.33}$$

from (22) and since $\{t_k\}$ is a K-length probability distribution.

The logistic bound follows similarly from (140) and (142):

$$\mathbf{A} = \sum_{k=1}^{K} t_k \, I\left(a_k^t \geqslant \frac{1}{2}\right) + \sum_{k=1}^{K}(1-t_k)\, I\left(a_k^t < \frac{1}{2}\right) \tag{A.34}$$

$$\geqslant \sum_{k=1}^{K} t_k \ln 2a_k^t + \sum_{k=1}^{K}(1-t_k)\ln(2(1-a^t)) \tag{A.35}$$

$$= \sum_{k=1}^{K} t_k \ln a_k^t + \ln 2 + \sum_{k=1}^{K}(1-t_k)\ln(1-a^t) + (K-1)\ln 2 \tag{A.36}$$

$$= \sum_{k=1}^{K} t_k \ln a_k^t + \sum_{k=1}^{K}(1-t_k)\ln(1-a^t) + K\ln 2 \tag{A.37}$$

$$= L_{log} + K\ln 2 \tag{A.38}$$

from (62). ■

Theorem 7 (*Discrete BAM Theorem*)**.** *Every connection matrix W is bidirectionally stable for visible and hidden neurons with sufficiently steep sigmoid activations.*

Proof. Let W be any $I \times J$ matrix that connects the visible field of I sigmoidal neurons with the hidden field of J sigmoidal neurons. The jth hidden neuron receives the W-filtered inner product

$$o_j^h = \sum_{i=1}^{I} w_{ij} a_i^v(o_i^v) \tag{A.39}$$

from the I visible neurons if we ignore external inputs. The ith visible neuron likewise receives the W^T-filtered inner product

$$o_i^v = \sum_{j=1}^{J} w_{ij} a_j^h(o_j^h) \tag{A.40}$$

from the J visible neurons. Define the quadratic energy function E as

$$E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta) = -\sum_{i=1}^{I}\sum_{j=1}^{J} w_{ij} a_i^v a_j^h \tag{A.41}$$

Then E is bounded below: $E \geqslant -\sum_{i=1}^{I}\sum_{j=1}^{J}|w_{ij}|$. We will show that E is a global Lyapunov function for the two-layer network by showing that a state change in either field can only decrease E and thus that $\Delta E < 0$ along system trajectories. The state changes also have a minimal step size. So E stops decreasing after a finite number of steps.

Suppose a nonempty subset of the I visible neurons changes state from time increment t to $t+1$. There are $2^I - 1$ such subsets. Suppose the ith neuron belongs to this subset. Then $\Delta a_i^v \neq 0$. So either $\Delta a_i^v = 1 - 0 = 1$ or $\Delta a_i^v = 0 - 1 = -1$ since the sigmoidal activation a_i^v is sufficiently steep to approximate a binary threshold. We assume the threshold is zero but it can any real number. The first case $\Delta a_i^v = 1$ holds if and only if the inner product o_i^v in (A.40) is positive at $t+1$ after being negative at t: $o_i^v > 0$. Then $\Delta a_i^v \sum_{j=1}^{J} w_{ij} a_j^h > 0$. The second case $\Delta a_i^v = -1$ occurs just in case $o_i^v < 0$ at $t+1$ after being positive at t. Then again $\Delta a_i^v \sum_{j=1}^{J} w_{ij} a_j^h > 0$.

So the total change ΔE due to these updating neurons during the backward pass obeys

$$\Delta E = E(t+1) - E(t) \tag{A.42}$$

$$= -\sum_{i=1}^{I} \Delta a_i^v \sum_{j=1}^{J} w_{ij} a_j^h \tag{A.43}$$

$$< 0 \tag{A.44}$$

from (A.41). A symmetric result holds for hidden neurons that change state during the forward pass: $\Delta E = -\sum_{j=1}^{J} \Delta a_j^h \sum_{i=1}^{I} w_{ij} a_i^v < 0$ for any of the $2^J - 1$ subsets of such hidden neurons. So $\Delta E < 0$ for a state change in either field. So every matrix W is bidirectionally globally stable. ■

Theorem 8 (*Contrastive-Divergence Learning in a BAM or RBM is Generalized EM*)**.** *The contrastive-divergence update equation (172) for the differentiable Gibbs likelihood function $p(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)$ in (166) at epoch n*

$$w_{ij}^{n+1} = w_{ij}^n + \eta \frac{\partial \ln p(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)}{\partial w_{ij}}\Big|_{\Theta=\Theta^n} \tag{173}$$

equals the GEM update equation at epoch n

$$w_{ij}^{n+1} = w_{ij}^n + \eta \frac{\partial Q\,(\Theta|\Theta^n)}{\partial w_{ij}}\Big|_{\Theta=\Theta^n}. \tag{174}$$

Proof. The EM surrogate likelihood Q-function $Q(\Theta|\Theta^n)$ of the BAM or RBM network takes the expectation of the joint log-likelihood $\ln p(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)$ with respect to the hidden posterior pdf $p(\mathbf{a}^{\mathbf{h}}|\mathbf{a}^{\mathbf{v}}, \Theta^n)$:

$$Q(\Theta|\Theta^n) = \mathbb{E}_{\mathbf{a}^{\mathbf{h}}|\mathbf{a}^{\mathbf{v}},\Theta^n}\{\ln p(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)\} \tag{A.45}$$

$$= \mathbb{E}_{\mathbf{a}^{\mathbf{h}}|\mathbf{a}^{\mathbf{v}},\Theta^n}\{-E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta) - \ln Z(\Theta)\}. \tag{A.46}$$

Then taking the derivative with respect to w_{ij} and using the Hebbian derivative result (170) gives

$$\frac{\partial Q\,(\Theta|\Theta^n)}{\partial w_{ij}} = \frac{\partial \mathbb{E}_{\mathbf{a}^{\mathbf{h}}|\mathbf{a}^{\mathbf{v}},\Theta^n}\{-E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta) - \ln Z(\Theta)\}}{\partial w_{ij}} \tag{A.47}$$

$$= \mathbb{E}_{\mathbf{a}^{\mathbf{h}}|\mathbf{a}^{\mathbf{v}},\Theta^n}\left\{-\frac{\partial E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)}{\partial w_{ij}} - \frac{\partial \ln Z(\Theta)}{\partial w_{ij}}\right\} \tag{A.48}$$

$$= \mathbb{E}_{\mathbf{a}^{\mathbf{h}}|\mathbf{a}^{\mathbf{v}},\Theta^n}\left\{a_i^v a_j^h - \frac{1}{Z(\Theta)}\frac{\partial Z(\Theta)}{\partial w_{ij}}\right\}. \tag{A.49}$$

The partition-function term expands with (170) as

$$\frac{1}{Z(\Theta)}\frac{\partial Z(\Theta)}{\partial w_{ij}} = \frac{1}{Z(\Theta)}\frac{\partial\left\{\sum_{\mathbf{a}^{\mathbf{v}}}\sum_{\mathbf{a}^{\mathbf{h}}}\exp(-E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta))\right\}}{\partial w_{ij}} \tag{A.50}$$

$$= \frac{1}{Z(\Theta)}\sum_{\mathbf{a}^{\mathbf{v}}}\sum_{\mathbf{a}^{\mathbf{h}}}\frac{\partial \exp(-E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta))}{\partial w_{ij}} \tag{A.51}$$

$$= \frac{1}{Z(\Theta)}\sum_{\mathbf{a}^{\mathbf{v}}}\sum_{\mathbf{a}^{\mathbf{h}}} -\exp(-E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta))\frac{\partial E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)}{\partial w_{ij}} \tag{A.52}$$

$$= \frac{1}{Z(\Theta)}\sum_{\mathbf{a}^{\mathbf{v}}}\sum_{\mathbf{a}^{\mathbf{h}}}\exp(-E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta))a_i^v a_j^h \tag{A.53}$$

$$= \sum_{\mathbf{a}^{\mathbf{v}}}\sum_{\mathbf{a}^{\mathbf{h}}}\frac{\exp(-E(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta))}{Z(\Theta)}a_i^v a_j^h \tag{A.54}$$

$$= \sum_{\mathbf{a}^{\mathbf{v}}}\sum_{\mathbf{a}^{\mathbf{h}}} p(\mathbf{a}^{\mathbf{v}}, \mathbf{a}^{\mathbf{h}}|\Theta)\, a_i^v a_j^h \tag{A.55}$$

$$= \mathbb{E}_{\mathbf{a}^{\mathbf{v}},\mathbf{a}^{\mathbf{h}}|\Theta}\{a_i^v(\tilde{x}_i)a_j^h(\tilde{h}_j)\}. \tag{A.56}$$

So the partial derivative of the Q-function becomes

$$\frac{\partial Q\,(\Theta|\Theta^n)}{\partial w_{ij}} = \mathbb{E}_{\mathbf{a}^{\mathbf{h}}|\mathbf{a}^{\mathbf{v}},\Theta^n}\left\{a_i^v a_j^h - \mathbb{E}_{\mathbf{a}^{\mathbf{v}},\mathbf{a}^{\mathbf{h}}|\Theta}\{a_i^v a_j^h\}\right\} \tag{A.57}$$

$$= \mathbb{E}_{\mathbf{a}^{\mathbf{h}}|\mathbf{a}^{\mathbf{v}},\Theta^n}\{a_i^v a_j^h\} - \mathbb{E}_{\mathbf{a}^{\mathbf{v}},\mathbf{a}^{\mathbf{h}}|\Theta}\{a_i^v a_j^h\} \tag{A.58}$$

since the expectation of a constant equals the constant. This learning term gives the GEM gradient-ascent equation:

$$w_{ij}^{n+1} = w_{ij}^n + \eta \frac{\partial Q\left(\Theta|\Theta^n\right)}{\partial w_{ij}}\Big|_{\Theta=\Theta^n} \tag{A.59}$$

$$= w_{ij}^n + \eta\left(\mathbb{E}_{\mathbf{a^h}|\mathbf{a^v},\Theta^n}\{a_i^v a_j^h\} - \mathbb{E}_{\mathbf{a^v},\mathbf{a^h}|\Theta^n}\{a_i^v a_j^h\}\right) \tag{A.60}$$

$$= w_{ij}^n + \eta \frac{\partial \ln p(\mathbf{a^v}, \mathbf{a^h}|\Theta)}{\partial w_{ij}}\Big|_{\Theta=\Theta^n} \tag{A.61}$$

from (172). So the two update equations are identical. ■

Theorem 9 (*Logistic–Logistic Hyperplane Noise Benefit*)**.** *The NEM positivity condition holds for a Bernoulli–Bernoulli (logistic–logistic) BAM or RBM at iteration n if*

$$\mathbb{E}_{\mathbf{a^v},\mathbf{a^h},\mathbf{n}|\Theta^*}\left\{\mathbf{n}^T(\mathbf{W}\mathbf{a^h} + \mathbf{b})\right\} \geqslant \mathbb{E}_{\mathbf{n}|\Theta^*}\left[\ln Z_{\mathbf{n}}(\Theta^n)\right] - \ln Z(\Theta^n)\,. \tag{181}$$

Proof. The Bernoulli–Bernoulli energy $E(\mathbf{a^v}, \mathbf{a^h}|\Theta)$ in (168) gives the energy difference

$$E(\mathbf{a^v}, \mathbf{a^h}|\Theta^n) - E(\mathbf{a^v} + \mathbf{n}, \mathbf{a^h}|\Theta^n)$$
$$= \sum_{i=1}^{I}\sum_{j=1}^{J} w_{ij} n_i a_j^h(\tilde{h}_j) + \sum_{i=1}^{I} b_i n_i\,. \tag{A.62}$$

Then putting (A.62) into (179) gives the NEM noise-benefit condition for this logistic–logistic BAM/RBM:

$$\mathbb{E}_{\mathbf{x},\mathbf{h},\mathbf{n}|\Theta^*}\left\{\sum_{i=1}^{I}\sum_{j=1}^{J} w_{ij} n_i a_j^h(\tilde{h}_j) + \sum_{i=1}^{I} b_i n_i\right\} \geqslant \mathbb{E}_{\mathbf{a^v},\mathbf{h},\mathbf{n}|\Theta^*} \ln \frac{Z_{\mathbf{n}}(\Theta^n)}{Z(\Theta^n)}\,.$$

The term in brackets has the matrix–vector form

$$\sum_{i=1}^{I}\sum_{j=1}^{J} w_{ij} n_i a_j^h + \sum_{i=1}^{I} n_i b_i = \mathbf{n}^T(\mathbf{W}\mathbf{a^h} + \mathbf{b})\,. \tag{A.63}$$

So the NEM condition becomes a hyperplane inequality:

$$\mathbb{E}_{\mathbf{a^v},\mathbf{a^h},\mathbf{n}|\Theta^*}\left\{\mathbf{n}^T(\mathbf{W}\mathbf{a^h} + \mathbf{b})\right\} \geqslant \mathbb{E}_{\mathbf{n}|\Theta^*}\left[\ln Z_{\mathbf{n}}(\Theta^n)\right]$$
$$- \ln Z(\Theta^n)\,. \quad ■ \tag{A.64}$$

Theorem 10 (*Gaussian-Logistic Spherical Noise Benefit*)**.** *The NEM positivity condition holds for training a Gaussian–Bernoulli BAM or RBM at iteration n if*

$$\mathbb{E}_{\mathbf{a^v},\mathbf{a^h},\mathbf{n}|\Theta^*}\left\{\frac{1}{2}\|\mathbf{n}\|^2 - \mathbf{n}^T(\mathbf{W}\mathbf{a^h} + \mathbf{b} - \mathbf{x})\right\} \tag{182}$$
$$\leqslant \ln Z(\Theta^n) - \mathbb{E}_{\mathbf{n}|\Theta^*}\left[\ln Z_{\mathbf{n}}(\Theta^n)\right]\,.$$

Proof. Putting the Gaussian-logistic energy function in (169) into (179) gives the noise-benefit condition for a Gaussian(visible)–Bernoulli(hidden) BAM or RBF:

$$\mathbb{E}_{\mathbf{a^v},\mathbf{a^h},\mathbf{n}|\Theta^*}\left\{\sum_{i=1}^{I}\sum_{j=1}^{J} w_{ij} n_i a_j^h + \sum_{i=1}^{I} n_i b_i - \frac{1}{2}\sum_{i=1}^{I} n_i^2\right. \tag{A.65}$$
$$\left. - \sum_{i=1}^{I} n_i a_i^v\right\} \geqslant \mathbb{E}_{\mathbf{n}|\Theta^*}\left[\ln Z_{\mathbf{n}}(\Theta^n)\right] - \ln Z(\Theta^n)\,.$$

The term in brackets has the vector–matrix form

$$\sum_{i=1}^{I}\sum_{j=1}^{J} w_{ij} n_i a_j^h + \sum_{i=1}^{I} n_i b_i - \frac{1}{2}\sum_{i=1}^{I} n_i^2 - \sum_{i=1}^{I} n_i a_i^v$$
$$= \mathbf{n}^T(\mathbf{W}\mathbf{a^h} + \mathbf{b} - \mathbf{a^v}) - \frac{1}{2}\|\mathbf{n}\|^2\,. \tag{A.66}$$

So taking expectations of both sides of (A.65) gives the noise-benefit sufficient condition as the quadratic condition (182). ■

References

Adigun, O., & Kosko, B. (2016). Bidirectional representation and backpropagation learning. In *International joint conference on advances in big data analytics* (pp. 3–9).

Adigun, O., & Kosko, B. (2017). Using noise to speed up video classification with recurrent backpropagation. In *Neural networks (IJCNN), 2017 international joint conference on* (pp. 108–115). IEEE.

Adigun, O., & Kosko, B. (2018). Training generative adversarial networks with bidirectional backpropagation. In *2018 17th IEEE international conference on machine learning and applications (ICMLA)* (pp. 1178–1185). IEEE.

Adigun, O., & Kosko, B. (2019a). Bidirectional backpropagation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(5), 1982 - 1994, May 2020.

Adigun, O., & Kosko, B. (2019b). Noise-boosted bidirectional backpropagation and adversarial learning. *Neural Networks*, *120*, 9–31.

Amari, S.-I. (1995). Information geometry of the EM and em algorithms for neural networks. *Neural Networks*, *8*(9), 1379–1408.

An, G. (1996). The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation*, *8*(3), 643–674.

Audhkhasi, K., Osoba, O., & Kosko, B. (2016). Noise-enhanced convolutional neural networks. *Neural Networks*, *78*, 15–23.

Azamimi, A., Uwate, Y., & Nishio, Y. (2008). An analysis of chaotic noise injected to backpropagation algorithm in feedforward neural network. In *Proceedings of IWVCC08* (pp. 70–73). Citeseer.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, *39*(3), 930–945.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127.

Bishop, C. M. (1995). Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, *7*(1), 108–116.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Bulsara, A., Boss, R., & Jacobs, E. (1989). Noise effects in an electronic model of a single neuron. *Biological Cybernetics*, *61*(3), 211–222.

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991). Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, *4*(6), 759–771.

Ciresan, D., Meier, U., Gambardella, L., & Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, *22*(12), 3207–3220.

Cohen, M. A., & Grossberg, S. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, (5), 815–826.

Cook, G. D., & Robinson, A. J. (1995). Training MLPs via the expectation maximization algorithm. In *Proc. artificial neural networks*. IET.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303–314.

Dahl, G., Ranzato, M., Mohamed, A., & Hinton, G. (2010). Phone recognition with the mean-covariance restricted Boltzmann machine. In *Proc. NIPS, Vol. 23* (pp. 469–477).

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, 1–38.

Deselaers, T., Hasan, S., Bender, O., & Ney, H. (2009). A deep learning approach to machine transliteration. In *Proceedings of the fourth workshop on statistical machine translation* (pp. 233–241). Association for Computational Linguistics.

Efron, B., & Hastie, T. (2016). *Computer age statistical inference, Vol. 5*. Cambridge University Press.

Franzke, B., & Kosko, B. (2011). Noise can speed convergence in Markov Chains. *Physical Review E*, *84*(4), 041112.

Franzke, B., & Kosko, B. (2015). Using noise to speed up Markov Chain Monte Carlo estimation. *Procedia Computer Science*, *53*, 113–120.

Gammaitoni, L., Hänggi, P., Jung, P., & Marchesoni, F. (1998). Stochastic resonance. *Reviews of Modern Physics*, *70*(1), 223.

Girosi, F., Jones, M. B., & Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural Computation*, *7*(2), 219–269.

Grossberg, S. (1988). Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural Networks*, *1*(1), 17–61.

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., et al. (2016). Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, *316*(22), 2402–2410.

Guo, Y., Zhou, D., Nie, R., Ruan, X., & Li, W. (2019). Deepanf: A deep attentive neural framework with distributed representation for chromatin accessibility prediction. *Neurocomputing*.

Gutmann, M. U., & Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research (JMLR)*, *13*(1), 307–361.

Hamel, P., & Eck, D. (2010). Learning features from music audio with deep belief networks. In *Proc. ISMIR*.

Hayakawa, Y., Marumoto, A., & Sawada, Y. (1995). Effects of the chaotic noise on the performance of a neural network model for optimization problems. *Physical Review E*, *51*(4), 2693–2696.

Haykin, S. (1998). *Neural networks: A comprehensive foundation*. Prentice Hall.

Hinton, G. (2018). Deep learning: a technology with the potential to transform health care. *Journal of the American Medical Association*, *320*(11), 1101–1102.

Hinton, G. (2020). Training a deep autoencoder or a classifier on MNIST digits. http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html. [Online; accessed February-2020].

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-R., Jaitly, N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*.

Hinton, G., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Hogg, R. V., McKean, J., & Craig, A. T. (2013). *Introduction to mathematical statistics*. Pearson.

Holmstrom, L., & Koistinen, P. (1992). Using additive noise in back-propagation training. *IEEE Transactions on Neural Networks*, *3*(1), 24–38.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Hou, R., Zhou, D., Nie, R., Liu, D., & Ruan, X. (2019). Brain CT and MRI medical image fusion using convolutional neural networks and a dual-channel spiking cortical model. *Medical & Biological Engineering & Computing*, *57*(4), 887–900.

Hu, X., Cammann, H., Meyer, H.-A., Miller, K., Jung, K., & Stephan, C. (2013). Artificial neural networks and prostate cancertool for diagnosis and management. *Nature Reviews Urology*.

Intrator, O., & Intrator, N. (2001). Interpreting neural-network results: a simulation study. *Computational Statistics & Data Analysis*, *37*(3), 373–393.

Jang, J.-S. R., & Sun, C.-T. (1993). Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Transactions on Neural Networks*, *4*(1), 156–159.

Jordan, M., & Mitchell, T. (2015). Machine learning: trends, perspectives, and prospects. *Science*, *349*, 255–260.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kosko, B. (1987). Adaptive bidirectional associative memories. *Applied Optics*, *26*(23), 4947–4960.

Kosko, B. (1988). Bidirectional associative memories. *IEEE Transactions on Systems, Man and Cybernetics*, *18*(1), 49–60.

Kosko, B. (1990). Unsupervised learning in noise. *IEEE Transactions on Neural Networks*, *1*(1), 44–57.

Kosko, B. (1991). *Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence*. Prentice Hall.

Kosko, B. (1994). Fuzzy systems as universal approximators. *IEEE Transactions on Computers*, *43*(11), 1329–1333.

Kosko, B. (1996). *Fuzzy engineering*. Prentice Hall.

Kosko, B. (2006). *Noise*. Viking.

Kosko, B. (2018). Additive fuzzy systems: From generalized mixtures to rule continua. *International Journal of Intelligent Systems*, *33*(8), 1573–1623.

Kosko, B., Lee, I., Mitaim, S., Patel, A., & Wilde, M. M. (2009). Applications of forbidden interval theorems in stochastic resonance. In *Applications of nonlinear dynamics* (pp. 71–89). Springer.

Kosko, B., & Mitaim, S. (2003). Stochastic resonance in noisy threshold neurons. *Neural Networks*, *16*(5), 755–761.

Kosko, B., & Mitaim, S. (2004). Robust stochastic resonance for simple threshold neurons. *Physical Review E*, *70*(3), 031911.

Kung, S. Y. (2014). *Kernel methods and machine learning*. Cambridge University Press.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444.

Lee, I., Liu, X., Zhou, C., & Kosko, B. (2006). Noise-enhanced detection of subthreshold signals with carbon nanotubes. *IEEE Transactions on Nanotechnology*, *5*(6), 613–627.

Matsuoka, K. (1992). Noise injection into inputs in back-propagation learning. *IEEE Transactions on Systems, Man and Cybernetics*, *22*(3), 436–440.

McDonnell, M., Stocks, N., Pearce, C., & Abbott, D. (2008). *Stochastic resonance: from suprathreshold stochastic resonance to stochastic signal quantization*. Cambridge University Press.

McLachlan, G. J., & Krishnan, T. (2007). *The EM algorithm and extensions, Vol. 382*. Wiley-Interscience.

McLachlan, G., & Peel, D. (2000). *Finite mixture models*. Wiley-Interscience.

Minsky, M. (1961). Steps toward artificial intelligence. *Proceedings of the IRE*, *49*(1), 8–30.

Mitaim, S., & Kosko, B. (1998). Adaptive stochastic resonance. *Proceedings of the IEEE*, *86*(11), 2152–2183.

Mitaim, S., & Kosko, B. (2004). Adaptive stochastic resonance in noisy neurons based on mutual information. *IEEE Transactions on Neural Networks*, *15*(6), 1526–1540.

Mitaim, S., & Kosko, B. (2014). Noise-benefit forbidden-interval theorems for threshold signal detectors based on cross correlations. *Physical Review E*, *90*(5), 052124.

Mnih, A., & Kavukcuoglu, K. (2013). Learning word embeddings efficiently with noise-contrastive estimation. In *Proc. Advances in neural information processing systems* (pp. 2265–2273).

Mohamed, A., Dahl, G., & Hinton, G. (2009). Deep belief networks for phone recognition. In *Proc. NIPS workshop on deep learning for speech recognition and related applications*.

Mohamed, A., Dahl, G., & Hinton, G. (2012). Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, *20*(1), 14–22.

Mohamed, A., Sainath, T., Dahl, G., Ramabhadran, B., Hinton, G., & Picheny, M. (2011). Deep belief networks using discriminative features for phone recognition. In *Acoustics, speech and signal processing (ICASSP), 2011 IEEE international conference on* (pp. 5060–5063). IEEE.

Mohamed, A., Yu, D., & Deng, L. (2010). Investigation of full-sequence training of deep belief networks for speech recognition. In *Proc. Interspeech* (pp. 2846–2849). Citeseer.

Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, *13*(6), 47–60.

Nair, V., & Hinton, G. (2009). 3D object recognition with deep belief nets. *Advances in Neural Information Processing Systems*, *22*, 1339–1347.

Ng, S.-K., & McLachlan, G. J. (2004). Using the EM algorithm to train neural networks: misconceptions and a new algorithm for multiclass classification. *IEEE Transactions on Neural Networks*, *15*(3), 738–749.

Oakes, D. (1999). Direct calculation of the information matrix via the EM. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, *61*(2), 479–482.

Osoba, O., & Kosko, B. (2013). Noise-enhanced clustering and competitive learning algorithms. *Neural Networks*.

Osoba, O., & Kosko, B. (2016a). The noisy expectation–maximization algorithm for multiplicative noise injection. *Fluctuation and Noise Letters*, 1350012.

Osoba, O., & Kosko, B. (2016b). The noisy expectation-maximization algorithm for multiplicative noise injection. *Fluctuation and Noise Letters*, 1650007.

Osoba, O., Mitaim, S., & Kosko, B. (2011a). Bayesian inference with adaptive fuzzy priors and likelihoods. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, *41*(5), 1183–1197.

Osoba, O., Mitaim, S., & Kosko, B. (2011b). Noise benefits in the expectation-maximization algorithm: NEM theorems and models. In *The international joint conference on neural networks (IJCNN)* (pp. 3178–3183). IEEE.

Osoba, O., Mitaim, S., & Kosko, B. (2013). The noisy expectation–maximization algorithm. *Fluctuation and Noise Letters*, *12*(03), 1350012.

Patel, A., & Kosko, B. (2008). Stochastic resonance in continuous and spiking neurons with levy noise. *IEEE Transactions on Neural Networks*, *19*(12), 1993–2008.

Patel, A., & Kosko, B. (2009). Error-probability noise benefits in threshold neural signal detection. *Neural Networks*, *22*(5), 697–706.

Patel, A., & Kosko, B. (2010). Optimal mean-square noise benefits in quantizer-array linear estimation. *IEEE Signal Processing Letters*, *17*(12), 1005–1009.

Patel, A., & Kosko, B. (2011). Noise benefits in quantizer-array correlation detection and watermark decoding. *IEEE Transactions on Signal Processing*, *59*(2), 488–505.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, *29*(9), 2352–2449.

Reed, R., Marks, R., & Oh, S. (1995). Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter. *IEEE Transactions on Neural Networks*, *6*(3), 529–538.

Reed, R., Oh, S., & Marks, R. (1992). Regularization using jittered training data. In *Neural networks, 1992. IJCNN., International joint conference on, Vol. 3* (pp. 147–152). IEEE.

Ripley, B. D. (1994). Neural networks and related methods for classification. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, 409–456.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323–533.

Sainath, T., Kingsbury, B., Ramabhadran, B., Fousek, P., Novak, P., & Mohamed, A. (2011). Making deep belief networks effective for large vocabulary continuous speech recognition. In *Proc. ASRU* (pp. 30–35). IEEE.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.

Seide, F., Li, G., & Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. In *Proc. Interspeech* (pp. 437–440).

Smolensky, P. (1986). *Information processing in dynamical systems: Foundations of harmony theory*. Boulder: Department of Computer Science, University of Colorado.

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

Susskind, J., Hinton, G., Movellan, J., & Anderson, A. (2008). Generating facial expressions with deep belief nets. *Affective Computing, Emotion Modelling, Synthesis and Recognition*, 421–440.

Teicher, H. (1963). Identifiability of finite mixtures. *The Annals of Mathematical Statistics*, *34*(4), 1265–1269.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, 267–288.

Tucker, H. G. (2013). *A graduate course in probability*. Courier Corporation.

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

Werbos, P. J. (1974). *Applied mathematics, Beyond regression: New tools for prediction and analysis in the behavioral sciences* (Doctoral Dissertation), MA: Harvard University.

Widrow, B., & McCool, J. M. (1976). A comparison of adaptive algorithms based on the methods of steepest descent and random search. *IEEE Transactions on Antennas and Propagation*, *24*(5), 615–637.

Wilde, M., & Kosko, B. (2009). Quantum forbidden-interval theorems for stochastic resonance. *Journal of Physical A: Mathematical Theory*, *42*(46).

Xu, R., & Wunsch, D. (2008). *Clustering, Vol. 10*. John Wiley & Sons.