Learning Non-deterministic Representations with Energy-based Ensembles

Maruan Al-Shedivat[*]
Computer, Electrical and Mathematical Sciences and Engineering Division
King Abdullah University of Science and Technology (KAUST)
Thuwal 23955-6900, Saudi Arabia
maruan.shedivat@kaust.edu.sa

Emre Neftci and Gert Cauwenberghs
Institute for Neural Computation
University of California, San Diego
La Jolla, CA 92093, USA
{nemre,gert}@ucsd.edu

Abstract

The goal of a generative model is to capture the distribution underlying the data, typically through latent variables. After training, these variables are often used as a new representation, more effective than the original features in a variety of learning tasks. However, the representations constructed by contemporary generative models are usually point-wise deterministic mappings from the original feature space. Thus, even with representations robust to class-specific transformations, statistically driven models trained on them would not be able to generalize when the labeled data is scarce. Inspired by the stochasticity of the synaptic connections in the brain, we introduce Energy-based Stochastic Ensembles. These ensembles can learn non-deterministic representations, i.e., mappings from the feature space to a family of distributions in the latent space. These mappings are encoded in a distribution over a (possibly infinite) collection of models. By conditionally sampling models from the ensemble, we obtain multiple representations for every input example and effectively augment the data. We propose an algorithm similar to contrastive divergence for training restricted Boltzmann stochastic ensembles. Finally, we demonstrate the concept of the stochastic representations on a synthetic dataset as well as test them in the one-shot learning scenario on MNIST.

1 Introduction

Learning data representations is a powerful technique that has been widely adopted in many fields of artificial intelligence (Bengio, 2009). Its main goal is usually to learn transformations of the data that disentangle different classes in the new space, that are robust to the noise in the input and tolerant to the variations along the class-specific manifolds (DiCarlo & Cox, 2007). A widely used technique of constructing representations is based on using probabilistic generative models of the data. Latent variables in such models can capture high-order correlations between the samples and are successful as new representations in a variety of tasks (Bengio, 2009). However, even high quality representations do not solve the problem of generalization: statistically derived discriminative models require a sufficient number of labeled training examples in order to exhibit good performance when tested.

The standard way of ameliorating the problem of overfitting due to the limited training data is based on enforcing a regularization (Bishop, 2006). Maaten et al. (2013) recently demonstrated that the artificial data augmentation via feature corruption effectively plays the role of a data-adaptive regularization. Wager et al. (2013) also showed that the dropout techniques applied to generalized linear models result into an adaptive regularization. In other words, these approaches confirm that having more (even corrupted) training data is equivalent to regularizing the model.

---

[*]Corresponding author: maruan.alshedivat.com.

On the other hand, instead of corrupting features one can try perform regularization by perturbing the learning model itself. Parameter perturbation leads to the notion of (possibly infinite) collection of models that we call *stochastic ensemble*. The Dropout (Hinton et al., 2012) and DropConnect (Wan et al., 2013) techniques are successful examples of using a particular form of stochastic ensemble in the context of feedforward neural networks. A unified framework for a collection of perturbed models (which also encompasses the data corruption methods) was recently introduced by Bachman et al. (2014): An arbitrary noise process was used to perturb a parametric parent model to generate an ensemble of child models. The training was performed by minimizing the marginalized loss function, and yielded an effectively regularized parent model that was robust to the introduced noise.

The above mentioned approaches use *fixed* corruption processes to regularize the original model and improve its generalization. Injection of arbitrary noise improves the robustness of the model to the corruption process (Maaten et al., 2013), but it does not necessarily capture the information about the generative process behind the actual data. In order to capture the variance of the data manifold, we propose to learn the noise that perturbs the model. Our work is inspired by the adaptive stochasticity of the synaptic connections in the brain. According to the experimental data, synapses between cortical neurons are highly unreliable (Branco & Staras, 2009). Moreover, this type of stochasticity is adaptive and adjusted by the learning (Stevens & Wang, 1994).

In this paper, we introduce energy-based stochastic ensembles (EBSEs) which can be trained in unsupervised fashion to fit a data distribution. These ensembles result from energy-based models (EBMs) with stochastic parameters. The EBSE learning procedure optimizes the log-likelihood of the ensemble by tuning a parametric distribution over the models. This distribution is first arbitrarily initialized, and then tuned by an expectation-maximization (EM) like procedure: In the E-step, we perform sampling to estimate the necessary expectations, while in the M-step we maximize the log-likelihood with respect to the ensemble parameters. We further develop an algorithm for learning restricted Boltzmann stochastic ensembles (RBSE) similar to contrastive divergence.

Using a pre-trained ensemble, we further introduce the notion of non-deterministic representations: instead of constructing point-wise mappings from the original feature space to a new latent space, we propose using stochastic mappings (Figure 1a). The stochasticity of the representations is based on the distribution stored in the ensemble. For every input object we can sample multiple models, and hence perform non-deterministic transformations to obtain a set of different representations. The ensemble is tuned to capture the variance of the data, hence the stochastic representations are likely to better capture the data manifold. We demonstrate these concepts visually by performing experiments on a two-dimensional synthetic dataset. Finally, we train a classifier on the representations of the MNIST hand-written digits generated by an RBSE and observe that the generalization capability in the one-shot learning scenario improves.

2 ENERGY-BASED STOCHASTIC ENSEMBLES

The distribution of the binary data vectors $\boldsymbol{v} \in \{0, 1\}^D$ can be encoded with the following energy-based model (EBM) that has some binary hidden variables $\boldsymbol{h} \in \{0, 1\}^K$:

$$P(\boldsymbol{v}, \boldsymbol{h}; \theta) = \frac{e^{-E(\boldsymbol{v}, \boldsymbol{h}; \theta)}}{Z(\theta)}, \tag{1}$$

where θ denotes the model parameters, $E(\boldsymbol{v}, \boldsymbol{h}; \theta)$ is a parametric scalar energy function (LeCun et al., 2006), and $Z(\theta)$ is the normalizing coefficient (partition function). If we impose a distribution over the model parameters (i.e., perturb θ according to some noise generation process), we obtain an energy-based stochastic ensemble (EBSE). In order to optimize the distribution over the models in the ensemble, we parametrize the noise generation process with α:

$$P(\boldsymbol{v}, \boldsymbol{h}, \theta; \alpha) = P(\boldsymbol{v}, \boldsymbol{h} \mid \theta)P(\theta; \alpha) = \frac{e^{-E(\boldsymbol{v}, \boldsymbol{h}, \theta)}}{Z(\theta)}P(\theta; \alpha) = \frac{e^{-E(\boldsymbol{v}, \boldsymbol{h}, \theta) - \phi(\theta; \alpha)}}{\zeta(\alpha)}, \tag{2}$$

where $\phi(\theta; \alpha)$ is an additional energy potential, and $\zeta(\alpha)$ is a new partition function. EBSE can be trained by maximizing the data log-likelihood $\log P(\boldsymbol{V}; \alpha)$ with respect to parameters α, where $\boldsymbol{V}$ denotes the entire dataset.

The introduced form of the model (2) allows to think of $P(\theta; \alpha)$ as a prior. Hence, we can relate the optimization of EBSE with the Bayesian inference framework for a standard EBM by taking

(a) Point-wise vs. stochastic representations   (b) RBM   (c) RBSE
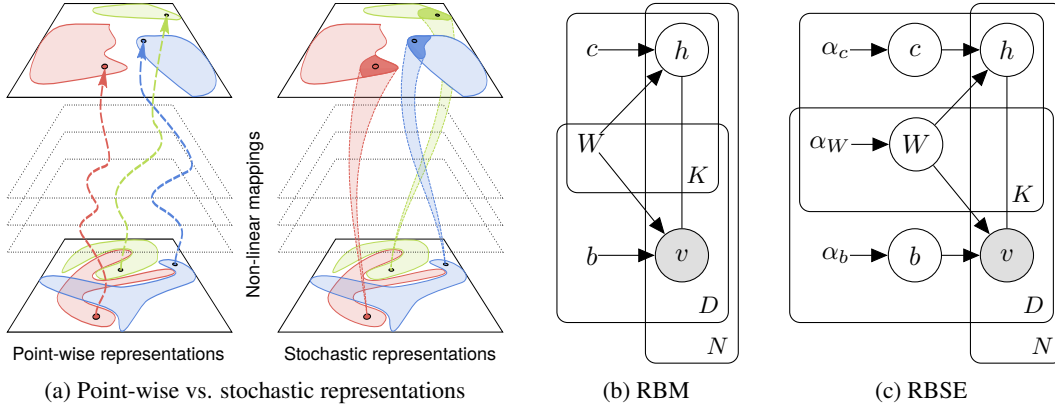
Figure 1: (a) The difference between point-wise and stochastic representations. While both mappings disentangle the manifolds of different classes, stochasticity additionally captures the data manifold in the representation space. (b) RBM and (c) RBSE graphical diagrams. The shaded nodes are the visible variable, the black nodes are the latent variables, literals outside of nodes are constants (model parameters). Plates denote variable scopes with sizes indicated at the bottom right corners.

expectation over the posterior $P(\theta \mid \boldsymbol{V}; \alpha)$:

$$
\begin{aligned}
\log P(\boldsymbol{V}; \alpha) &= \int P(\theta \mid \boldsymbol{V}; \alpha) \log P(\boldsymbol{V} \mid \theta) d\theta - D_{KL}\left[P(\theta \mid \boldsymbol{V}; \alpha) \| P(\theta; \alpha)\right] \\
&= \underbrace{\mathbb{E}\left[\log P(\boldsymbol{V} \mid \theta)\right]_{P(\theta | \boldsymbol{V}; \alpha)}}_{\text{Expected EBM log-likelihood}} - \underbrace{D_{KL}\left[P(\theta \mid \boldsymbol{V}; \alpha) \| P(\theta; \alpha)\right]}_{\text{KL divergence of posterior and prior}}.
\end{aligned}
\tag{3}
$$

Since the KL divergence is non-negative, by optimizing the log-likelihood of EBSE, we effectively maximize a lower bound on the EBM log-likelihood averaged over the posterior $P(\theta \mid \boldsymbol{V}; \alpha)$. This relates our approach to the feature corruption method (Maaten et al., 2013, Eq. 2) which optimizes the expected loss directly, but over a simple posterior feature corruption distribution $P(\tilde{\boldsymbol{v}} \mid \boldsymbol{v})$. Eventually, once we trained the stochastic ensemble of energy-based generative models, we get new parameters $\hat{\alpha}$ that make $P(\theta; \hat{\alpha})$ better tuned to capture the variance of the data.

Below, we derive the gradient of the EBSE log-likelihood (3) for the general energy case. Then, we focus on the restricted Boltzmann stochastic ensembles (RBSE), analyze their structure, and propose an efficient learning algorithm.

2.1 LOG-LIKELIHOOD OPTIMIZATION

The gradient of the log-likelihood function (3) can be written in the following way:

$$
\frac{\partial \log P(\boldsymbol{v}; \alpha)}{\partial \alpha} = -\frac{\partial \mathscr{F}(\boldsymbol{v}; \alpha)}{\partial \alpha} - \frac{1}{\zeta(\alpha)} \frac{\partial \zeta(\alpha)}{\partial \alpha},
\tag{4}
$$

where $\mathscr{F}(\boldsymbol{v}; \alpha) = -\log\left(\int e^{-E(\boldsymbol{v},\boldsymbol{h},\theta)-\phi(\theta;\alpha)} d\boldsymbol{h} d\theta\right)$ is called *free energy*. It is easy to show that the gradients of the free energy and the partition function have the following form:

$$
\frac{\partial \mathscr{F}(\boldsymbol{v}; \alpha)}{\partial \alpha} = \int P(\theta \mid \boldsymbol{v}; \alpha)\left(\frac{\partial \phi(\theta; \alpha)}{\partial \alpha}\right) d\theta, \quad \frac{1}{\zeta(\alpha)} \frac{\partial \zeta(\alpha)}{\partial \alpha} = -\int P(\theta; \alpha)\left(\frac{\partial \phi(\theta; \alpha)}{\partial \alpha}\right) d\theta.
\tag{5}
$$

The final expression for the gradient has a contrastive divergence like form (Hinton, 2002):

$$
\frac{\partial \log P(\boldsymbol{v}; \alpha)}{\partial \alpha} = \mathbb{E}\left[\frac{\partial \phi(\theta; \alpha)}{\partial \alpha}\right]_{P(\theta; \alpha)} - \mathbb{E}\left[\frac{\partial \phi(\theta; \alpha)}{\partial \alpha}\right]_{P(\theta | \boldsymbol{v}; \alpha)},
\tag{6}
$$

where the expectations $\mathbb{E}[\,\cdot\,]_{P(\theta; \alpha)}$ and $\mathbb{E}[\,\cdot\,]_{P(\theta | \boldsymbol{v}; \alpha)}$ are taken over the marginal and conditional distributions over the models, respectively. Based on the properties of the $P(\theta; \alpha)$ distribution, these expectations can be either fully estimated by Monte Carlo sampling, or partly computed analytically.

Algorithm 1 Expectation-maximization k-step contrastive divergence for RBSE

Input: S—training (mini-)batch; learning rate λ; se-RBM(v, h, W, b, c)
Output: gradient approximation $\Delta\alpha$ [depends on the parametrization of $\phi(W, b, c; \alpha)$]
 1: initialize $\Delta\alpha = 0$
 2: **for all** the $v \in S$ **do**
 3:     $v^{(0)} \leftarrow v$
 4:     $h^{(0)} \leftarrow$ persistent state (or sample)
 5:     sample $W^{(0)}, b^{(0)}, c^{(0)} \sim P(W, b, c \mid v^{(0)}, h^{(0)})$
 6:     **for** $t = 0, \ldots, k$ **do**   ▷ CD-k for sampling from $P(v, h)$
 7:         sample $W^{(t)}, b^{(t)}, c^{(t)} \sim P(W, b, c \mid v^{(t)}, h^{(t)})$
 8:         sample $v^{(t)} \sim P(v \mid h^{(t)}, W^{(t)}, b^{(t)}, c^{(t)})$
 9:         sample $h^{(t)} \sim P(h \mid v^{(t)}, W^{(t)}, b^{(t)}, c^{(t)})$
10:     $\Delta\alpha_m = \mathbb{E}\left[\frac{\partial\phi}{\partial\alpha} \mid v^{(k)}, h^{(k)}\right]$
11:     **for** $t = 0, \ldots, k$ **do**   ▷ CD-k for sampling from $P(h \mid v)$
12:         sample $W^{(t)}, b^{(t)}, c^{(t)} \sim P(W, b, c \mid v, h^{(t)})$
13:         sample $h'^{(t)} \sim P(h' \mid v, W^{(t)}, b^{(t)}, c^{(t)})$
14:     $\Delta\alpha_c = \mathbb{E}\left[\frac{\partial\phi}{\partial\alpha} \mid v, h'^{(k)}\right]$
15:     $\Delta\alpha \leftarrow \Delta\alpha + \lambda\left(\Delta\alpha_m - \Delta\alpha_c\right)$ ▷ Estimate SGD step
16: $\Delta\alpha = \Delta\alpha / \mathrm{cord}(S)$

Since the expectations depend on the parameter α, EBSE training is reminiscent of expectation-maximization (EM): After initializing α, in the E-step of the algorithm, we estimate $\mathbb{E}[\,\cdot\,]_{P(\theta;\alpha)}$ and $\mathbb{E}[\,\cdot\,]_{P(\theta|v;\alpha)}$ using the current value of α. In the M-step, we maximize the log-likelihood by using gradient-based optimization.

2.2 MODEL STRUCTURE

We further consider a specific energy-based stochastic ensemble composed of restricted Boltzmann machines (RBM). The energy function of the RBM is linear in each of the variables v, h, and θ:

$$E(v, h, \theta) = -(v^T W h + b^T v + c^T h) = -(\sum_{i,j} W_{ij} v_i h_j + \sum_i b_i v_i + \sum_j c_j h_j), \qquad (7)$$

where parameters θ are represented by a tuple (W, b, c), and $W \in \mathbb{R}^{D \times K}, b \in \mathbb{R}^D, c \in \mathbb{R}^K$.

RBM is an undirected graphical model that can be represented by two layers of interconnected probabilistic units (Figure 1b). These units can be seen as neurons with a probabilistic sigmoid activation function, and the graphical model can be represented by a two-layer neural network. In this case, the parameters W play the role of connection strengths between the neurons. By imposing a noise distribution on the parameters, connection strengths become stochastic. From a graphical model perspective, this is equivalent to introducing new latent variables (Figure 1c). Notice that the model becomes mixed directed and undirected—an ensemble of undirected RBMs generated by a simple Bayesian network composed of (W, b, c) random hidden variables.

In order to fully define the model, we need to choose a specific form of the marginal distribution $P(\theta; \alpha)$ that generates the ensemble. First, to make the expectations computationally feasible, we suppose that W, b, c are all marginally independent:

$$P(\theta; \alpha) = \prod_{ij} P(W_{ij}; \alpha_{W_{ij}}) \prod_i P(b_i; \alpha_{b_i}) \prod_j P(c_j; \alpha_{c_j}). \qquad (8)$$

Then, we consider two cases: (a) Bernoulli distributed parameters. In this case W_{ij} can be either zero with probability $1 - p_{ij}$, or equal to some $\bar{W}_{ij}$ with probability p_{ij}. The tunable parameters are $\alpha_{ij} = \{p_{ij}, \bar{W}_{ij}\}$. This case is similar to DropConnect (Wan et al., 2013) technique but with adaptive distributions over the connections between visible and hidden layers in RBM. (b) In the second case, parameters are normally distributed, i.e., $W_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij})$, and $\alpha_{W_{ij}} = \{\mu_{ij}, \sigma_{ij}\}$. In both cases, b and c are parametrized similarly as W. The number of parameters in the RBSE is twice the original number for the RBM.

This structure of the model (Figure 1c) implies the following set of independencies:

$$v_i \perp\!\!\!\perp v_j \mid \boldsymbol{h}, \quad h_i \perp\!\!\!\perp h_j \mid \boldsymbol{v}, \quad W_{ij} \perp\!\!\!\perp W_{lk}, \quad W_{ij} \not\perp\!\!\!\perp W_{ik} \mid v_i, \quad W_{ij} \not\perp\!\!\!\perp W_{lj} \mid h_i. \tag{9}$$

These expressions show the marginal independencies we purposefully incorporated into $P(\theta; \alpha)$ and conditional dependences between the components of $\boldsymbol{W}$. Due to these dependencies between stochastic connections though visible and hidden units, the model is able to capture the data variance in the distribution over the stochastic parameters θ. Importantly, even though the components of $\boldsymbol{W}$ are dependent on each other given $\boldsymbol{v}$ and $\boldsymbol{h}$, we are still able to factorize the conditional distribution $P(\theta \mid \boldsymbol{v}, \boldsymbol{h})$ using the renormalization procedure (see the supplementary material for details).

2.3 TRAINING

We propose the expectation-maximization k-step contrastive divergence algorithm for training RBSE (summarized in Algorithm 1) with two different types of stochastic connections—Bernoulli and Gaussian—between the visible and the hidden layers. To carry out the E-step, we need to compute the expectations in (6). We use Monte Carlo estimates of these expectations:

$$\mathbb{E}[\,\cdot\,]_{P(\theta;\alpha)} = \int d\boldsymbol{v} d\boldsymbol{h}\, P(\boldsymbol{v}, \boldsymbol{h}; \alpha) \int d\theta\, P(\theta \mid \boldsymbol{v}, \boldsymbol{h}; \alpha)[\,\cdot\,] \approx \frac{1}{M} \sum_{\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{h}} \sim P(\boldsymbol{v}, \boldsymbol{h}; \alpha)} \int d\theta\, P(\theta \mid \tilde{\boldsymbol{v}}, \tilde{\boldsymbol{h}}; \alpha)[\,\cdot\,],$$

$$\mathbb{E}[\,\cdot\,]_{P(\theta \mid \boldsymbol{v}; \alpha)} \approx \frac{1}{M} \sum_{\hat{\boldsymbol{h}} \sim P(\boldsymbol{h} \mid \boldsymbol{v}; \alpha)} \int d\theta\, P(\theta \mid \boldsymbol{v}, \hat{\boldsymbol{h}}; \alpha)[\,\cdot\,],$$

$$\tag{10}$$

where M is the number of samples used to approximate a distribution. The states $(\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{h}})$ should be sampled from the marginal model distribution $P(\boldsymbol{v}, \boldsymbol{h}; \alpha)$, and $\hat{\boldsymbol{h}}$ sampled from the marginal model conditional distribution $P(\boldsymbol{h} \mid \boldsymbol{v}; \alpha)$. This can be achieved by running a Gibbs sampling procedure as in standard contrastive divergence algorithm (see Algorithm 1).

Lastly, we need to compute the expectation over the posterior $P(\theta \mid \boldsymbol{v}, \boldsymbol{h}; \alpha)$ given a visible and a hidden state. In both Bernoulli and Gaussian cases, due to the structure of the ensemble distribution $P(\theta; \alpha)$ discussed in section 2.2, these expectations can be computed analytically. As an example, we get the following expressions for the gradient update of the Gaussian RBSE (for the Bernoulli case, the notation and other details see the supplementary material):

$$\frac{\partial \log P(\boldsymbol{v}; \alpha)}{\partial \bar{W}_{ij}} = \langle v_i h_i \rangle_{\text{data}} - \langle v_i h_i \rangle_{\text{recon}}, \quad \frac{\partial \log P(\boldsymbol{v}; \alpha)}{\partial \bar{\sigma}_{ij}} = \left\langle v_i^2 h_j^2 \sigma_{ij} \right\rangle_{\text{data}} - \left\langle v_i^2 h_j^2 \sigma_{ij} \right\rangle_{\text{recon}}.$$

For the models where the expectations over the ensemble are not analytically tractable, we can estimate them using Monte Carlo Markov chain sampling as well.

The main bottleneck of the algorithm is the number of samplings per gradient update. Since all the connections between the visible and the hidden units are probabilistic, the random variables need to be sampled a quadratic number of times compared to learning an RBM. However, due to the independencies introduced in Eq. (9), all the variables W_{ij}, b_i, and c_j can be sampled in parallel.

3 EXPERIMENTS

We introduce the concept of stochastic representations by considering a semi-supervised learning scenario where a large amount of data is available but with only a few labeled examples per class. RBSE is a generative model, and hence it can be trained in an unsupervised fashion on the entire dataset. Once the ensemble distribution $P(\theta; \alpha)$ is tuned to the data, for every labeled training example, we can conditionally sample models $\theta \sim P(\theta \mid \boldsymbol{v})$ and then use each model to generate a representation based on the activation of the hidden units $\boldsymbol{h}$. In other words, the generative stochastic ensembles can be used to store the information about the entire dataset and effectively augment the number of labeled examples by generating multiple stochastic representations. This is analogous to the concept of corrupted features (Maaten et al., 2013; Wager et al., 2013). The main difference is that RBSEs can learn from the unlabeled part of the dataset how to corrupt the data properly.

To test the concept, we implemented Bernoulli and a Gaussian RBSE using the Theano library (Bergstra et al., 2010). We trained our generative models on two-dimensional synthetic datasets and on MNIST digits. The following sections provide the details on each of the experiments.

(a) RBM performs deterministic mapping.

(b) RBSE captures the training data variance.

(c) Isolated points are attracted by the training data.
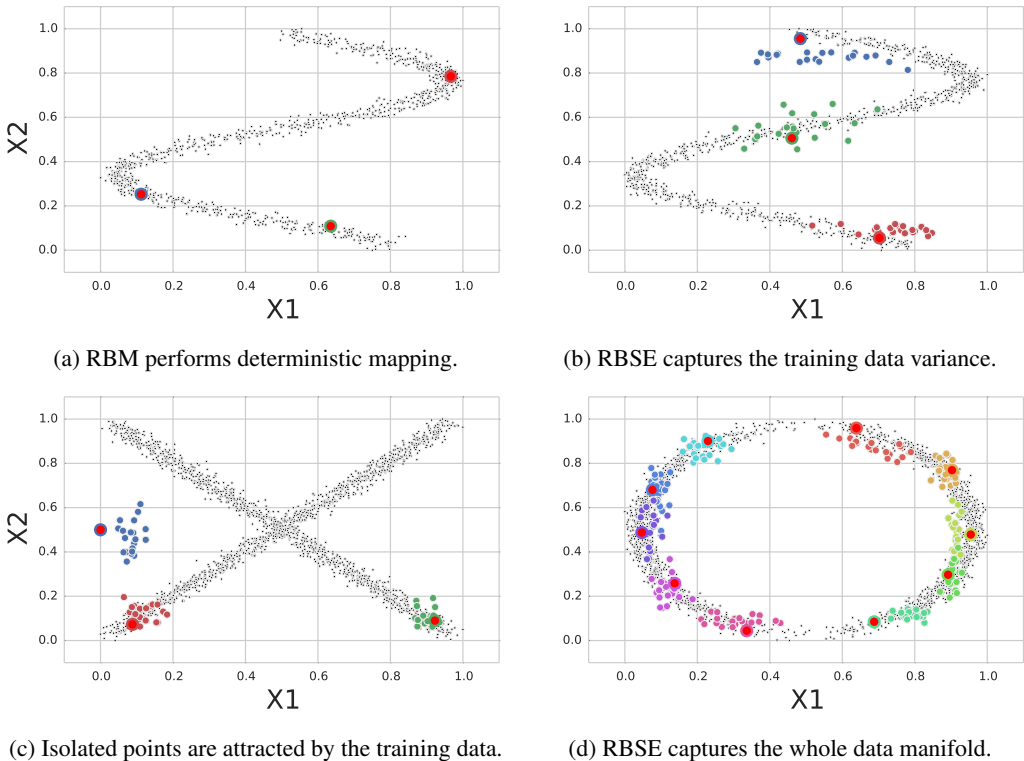
(d) RBSE captures the whole data manifold.

Figure 2: Visualization of the experiments with the synthetic two dimensional data. On all panels, the black points represent the training data, and the red points are the testing. Best viewed in color. (a) The classical RBM can perform only deterministic mapping. (b) RBSE maps the testing points to multiple representations which map backwards to different points in the original space. (c) RBSE attracts the representations of the isolated points towards the training data manifold. (d) Stochastic representations can capture the variance of the entire manifold from a few examples.

3.1 SYNTHETIC DATA

In order to visually test how a stochastic ensemble can capture the training data manifold, we generated several simple two-dimensional datasets $\{x_1, x_2\} \in [0, 1]^2$ (Figure 2). We trained an ordinary RBM with 2 visible and 8 hidden units and an RBSE of the same structure. Using these models, we mapped the testing points to the latent space multiple times, and then back to the original space.

For RBM, we used the mean field activations for new representations: $\boldsymbol{h}_{new} = P(\boldsymbol{h} \mid \boldsymbol{v})$ and $\boldsymbol{v}_{new} = P(\boldsymbol{v} \mid \boldsymbol{h}_{new})$. Unsurprisingly, the two consecutive transformations, from visible to hidden and back to visible space, performed by a properly trained RBM always mapped the testing points to themselves (Figure 2a). Notice that this holds not only for RBMs but for any deterministic model: Other point-wise deterministic representation learning techniques, e.g., autoencoders (Bengio, 2009), exhibit the same behavior.

We performed a similar procedure multiple times for RBSE: first, we sampled a model from the conditional distribution $P(\theta \mid \boldsymbol{v})$, then using $P(\boldsymbol{h} \mid \boldsymbol{v}, \tilde{\theta})$, we transformed testing points from the visible to the hidden space, then mapped backwards using the average model from the ensemble. Stochastic representations of the testing data, when mapped back to the visible space, were distributed along the training data manifold near the testing points they belonged to (Figure 2b).

The experiments also demonstrated that the representations of an isolated testing point (an outlier) will be attracted towards the manifold captured by RBSE (Figure 2c), which cannot be done by the RBM. Finally, an entire manifold can be captured by the generated stochastic representations for only a small number of initial data points (Figure 2d). These experiments confirmed the capability of RBSEs to refine its internal distribution over the models and to capture the variance of the data.
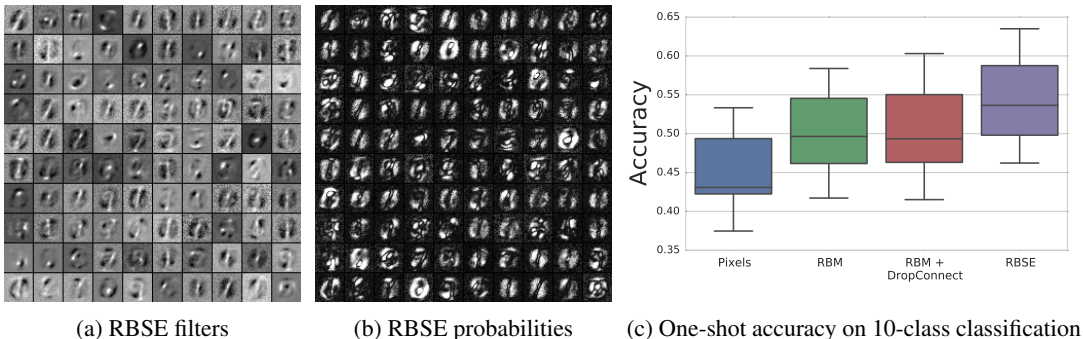
6

(a) RBSE filters   (b) RBSE probabilities   (c) One-shot accuracy on 10-class classification

Figure 3: After training on MNIST for several epochs: (a) RBSE filters, i.e., the $\bar{W}_{ij}$ values. (b) RBSE connection probabilities, i.e., the p_{ij} values—the dark pixels are close to 0; the light pixels are close to 1. (c) Performance of a logistic regression classifier trained on different representations of the data under the one-shot constraints.

Therefore, the stochasticity of such tunable ensembles should provide better regularization than just arbitrary noise injection and, as we show further, improve the perfomrance in one-shot learning.

3.2 ONE-SHOT LEARNING OF MNIST DIGITS

To test stochastic representations in the semi-supervised setting where only a few labeled data are available (one example per class), we performed experiments on MNIST digits dataset. We trained a Bernoulli RBSE model with 784 visible and 400 hidden units on 50,000 unlabeled MNIST digits using the proposed Algorithm 1 with a number of MCMC steps $k = 1$. The learned filters (i.e., $\bar{W}_{ij}$) and the Bernoulli connection probabilities (i.e., p_{ij}) arranged into tiles are presented on Figure 3. Notice that the connection probabilities encode a lot of structure (Figure 3b). For the purpose of comparison, we also trained an RBM of the same configuration on the same unlabeled MNIST data.

Next, we sampled 1000 objects (100 per class) from the remaining 20,000 MNIST digits: 1 training and 99 testing examples per class. For every training sample we used different approaches to construct the representations: (a) image pixels, (b) deterministic representations constructed by a pre-trained RBM, (c) stochastic representations constructed by an RBM with Bernoulli probabilities 0.5 for every connection (equivalent to DropConnect), and (d) RBSE-generated stochastic representations. In the (c) and (d) cases, we constructed 10 representations for every object.

Finally, we trained and tested a logistic regression classifier (with no additional regularization) on the these representations under one-shot learning constraints. The classification experiments were done multiple times for differently sampled objects. The results are presented on Figure 3c. About 10% improvement in classification accuracy is due to better representations learned by RBM (disentanglement of the classes). When we regularize the classifier by generating stochastic representations with DropConnect noise applied to trained RBM, the performance slightly drops. On the contrary, when the classifier is regularized through the representations generated by RBSE, we get about another 5% increase in accuracy on average.

4 DISCUSSION

In this paper, we introduced the concept non-deterministic representations that can be learned by Energy-based Stochastic Ensembles tuned to the data in unsupervised fashion. The ensemble stochasticity can capture the data variance and be used to adaptively regularize discriminative models and improve their performance in semi-supervised settings. The actual learning of a proper model perturbation from the data is the conceptual novelty compared to previous work (Bachman et al., 2014; Maaten et al., 2013; Wager et al., 2013). We illustrated the power of stochastic ensembles visually on synthetic two-dimensional data and demonstrated it quantitatively on one-shot learning of the MNIST hand-written digits.

The inspiration from synaptic stochasticity observed in biological nerve cells provides a number of insights and hypotheses for experimental neuroscience, which we will report separately elsewhere. From the artificial neural networks perspective, the proposed approach of using stochastic connections between neural units is interesting as well. For example, similar stochastic ensembles of feed-forward neural networks should be able to capture complex multi-modal data manifolds (Tang & Salakhutdinov, 2013). Also, recently proposed Generative Stochastic Networks (GSNs), which are used to encode probabilities in their sampling behavior (Bengio & Thibodeau-Laufer, 2013; Bengio et al., 2013), can be naturally endorsed with non-deterministic connections and might potentially realize richer families of distributions.

Interestingly, biological inspiration also suggests that neuromorphic computers that operate in a massively parallel fashion, while consuming a faction of the power of digital computers (Merolla et al., 2014) can be leveraged. They often natively support Bernoulli synaptic stochasticity (Goldberg et al., 2001), and neuromorphic variants of RBMs can be efficiently implemented and trained (Neftci et al., 2014). This suggests that the disadvantages associated to the computational overhead of RBSEs can be nullified by using an appropriate computational substrate.

REFERENCES

Bachman, Phil, Alsharif, Ouais, and Precup, Doina. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, pp. 3365–3373, 2014.

Bengio, Yoshua. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Bengio, Yoshua and Thibodeau-Laufer, Eric. Deep generative stochastic networks trainable by backprop. *arXiv preprint arXiv:1306.1091*, 2013.

Bengio, Yoshua, Yao, Li, Alain, Guillaume, and Vincent, Pascal. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pp. 899–907, 2013.

Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, pp. 3, 2010.

Bishop, Christopher M. *Pattern recognition and machine learning*, volume 1. Springer, 2006.

Branco, Tiago and Staras, Kevin. The probability of neurotransmitter release: variability and feedback control at single synapses. *Nature Reviews Neuroscience*, 10(5):373–383, 2009.

DiCarlo, James J and Cox, David D. Untangling invariant object recognition. *Trends in cognitive sciences*, 11(8):333–341, 2007.

Goldberg, D.H., Cauwenberghs, G., and Andreou, A.G. Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons. *Neural Networks*, 14(6–7):781–793, Sep 2001.

Hinton, Geoffrey E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Hinton, Geoffrey E, Srivastava, Nitish, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

LeCun, Yann, Chopra, Sumit, Hadsell, Raia, Ranzato, M, and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 2006.

Maaten, Laurens, Chen, Minmin, Tyree, Stephen, and Weinberger, Kilian Q. Learning with marginalized corrupted features. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 410–418, 2013.

Merolla, Paul A, Arthur, John V, Alvarez-Icaza, Rodrigo, Cassidy, Andrew S, Sawada, Jun, Akopyan, Filipp, Jackson, Bryan L, Imam, Nabil, Guo, Chen, Nakamura, Yutaka, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., and Cauwenberghs, G. Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience*, 7(272), 2014.

Stevens, Charles F and Wang, Yanyan. Changes in reliability of synaptic function as a mechanism for plasticity. *Nature*, 371(6499):704–707, 1994.

Tang, Yichuan and Salakhutdinov, Ruslan. Learning stochastic feedforward neural networks. In *Advances in Neural Information Processing Systems*, pp. 530–538, 2013.

Wager, Stefan, Wang, Sida, and Liang, Percy. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, pp. 351–359, 2013.

Wan, Li, Zeiler, Matthew, Zhang, Sixin, Cun, Yann L, and Fergus, Rob. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1058–1066, 2013.

SUPPLEMENTARY MATERIAL

A POSTERIOR BERNOULLI AND GAUSSIAN DISTRIBUTIONS

This section provides details on the posterior distributions for Bernoulli and Gaussian stochastic ensembles. RBM has three sets of parameters: connection strengths between the visible and hidden layers $\boldsymbol{W} \in \mathbb{R}^{D \times K}$, bias $\boldsymbol{b} \in \mathbb{R}^D$ for visible, and bias $\boldsymbol{c} \in \mathbb{R}^K$ for hidden units. We use i to index D visible dimensions, and j for K hidden dimensions. We denote unnormalized measures by $\tilde{P}(\cdot)$. When the equations for W_{ij}, b_i, and c_j have the same form, we refer to these components as θ_k.

For some fixed $(\boldsymbol{v}, \boldsymbol{h})$ we know that RBSE's energy is a linear function of θ. Since the prior $P(\theta; \alpha)$ factorizes over all the components of θ according to our definition. Thus, we can factorize unnormalized $\tilde{P}(\theta, \boldsymbol{v}, \boldsymbol{h}; \alpha)$ over all θ_k:

$$\tilde{P}(W_{ij}, v_i, h_j; \alpha_{ij}) = P(W_{ij}; \alpha_{ij}) \exp(v_i h_j W_{ij}),$$
$$\tilde{P}(b_i, v_i; \alpha_i) = P(b_i; \alpha_i) \exp(v_i b_i), \tag{A1}$$
$$\tilde{P}(c_j, h_j; \alpha_j) = P(c_j; \alpha_j) \exp(h_j c_j).$$

We can further renormalize (A1) and obtain the following posterior distribution:

$$
\begin{aligned}
P(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c} \mid \boldsymbol{v}, \boldsymbol{h}; \alpha) &= \frac{\prod_{ij} P(W_{ij}, v_i, h_j; \alpha_{ij}) \prod_i P(b_i, v_i; \alpha_i) \prod_j P(c_j, h_j; \alpha_j)}{\int \prod_{ij} P(W_{ij}, v_i, h_j; \alpha_{ij}) d\boldsymbol{W} \prod_i P(b_i, v_i; \alpha_i) d\boldsymbol{b} \prod_j P(c_j, h_j; \alpha_j) d\boldsymbol{c}} \\
&= \prod_{ij} \frac{P(W_{ij}, v_i, h_j; \alpha_{ij})}{\int P(W_{ij}, v_i, h_j; \alpha_{ij}) dW_{ij}} \prod_i \frac{P(b_i, v_i; \alpha_i)}{\int P(b_i, v_i; \alpha_i) db_i} \prod_j \frac{P(c_j, h_j; \alpha_j)}{\int P(c_j, h_j; \alpha_j) dc_j} \\
&= \prod_{ij} P(W_{ij} \mid v_i, h_j; \alpha_{ij}) \prod_i P(b_i \mid v_i; \alpha_i) \prod_j P(c_j \mid h_j; \alpha_j).
\end{aligned}
\tag{A2}
$$

As we see, if the prior distribution $P(\theta; \alpha)$ factorizes over the components of θ, the posterior distribution $P(\theta \mid \boldsymbol{v}, \boldsymbol{h}; \alpha)$ is also factorisable. Finally, we need to find the posterior distribution for each of the components for the Bernoulli and Gaussian ensembles.

For Bernoulli case, a priori, every component θ_k can either take some non-zero value $\bar{\theta}_k$ with probability $(1-p_k)$, or be equal to zero with probability p_k. Then, the distribution $P(\theta; \alpha)$ is parametrized by $\alpha = (\bar{\theta}_k, p_k)$ and has the following form:

$$P(\theta; \alpha) = \prod_k P(\theta_k; \alpha_k) = \prod_k \left[\delta(\theta_k)(1 - p_k) + \delta(\theta_k - \bar{\theta}_k) p_k \right], \tag{A3}$$

where $\delta(\cdot)$ is the Dirac delta function. The posterior for W_{ij} (similar for b_i and c_j) will be

$$P(W_{ij} \mid v_i, h_j; \alpha_{ij}) = \frac{\left(\delta(W_{ij})(1 - p_{ij}) + \delta(W_{ij} - \bar{W}_{ij}) p_{ij} \right) \exp(v_i h_j W_{ij})}{1 - p_{ij} + p_{ij} \exp(v_i h_j W_{ij})} \tag{A4}$$

For Gaussian case, components θ_k a priori have a Gaussian distribution with the mean $\bar{\theta}_k$ and variance σ_k^2, i.e., $\alpha_k = (\bar{\theta}_k, \sigma_k)$. Omitting the details of integration, the posterior for W_{ij} will have the following form (similar expressions are easy to derive for b_i and c_j):

$$P(W_{ij} \mid v_i, h_i; \alpha_{ij}) = \frac{1}{\sqrt{2\pi}\sigma_{ij}} \exp\left(-\left[\frac{W_{ij} - (\bar{W}_{ij} + v_i h_i \sigma_{ij}^2)}{\sqrt{2}\sigma_{ij}} \right]^2 \right). \tag{A5}$$

In both Bernoulli and Gaussian cases, due to the linear structure of the RBSE energy function and the structure of the prior $P(\theta; \alpha)$, we are able to get analytical expressions for the posterior of each component of θ. Moreover, the derivations will be the same not only for Bernoulli and Gaussian cases, but for any $P(\theta; \alpha)$ that is completely factorisable over $\{\theta_k\}$.

B   EXPECTATIONS OVER THE POSTERIOR

Here we provide details on the analytical computation of the expected stochastic gradient update for the two types of stochastic ensembles: Bernoulli and Gaussian. For the sake of space, as in Appendix A, when there is no ambiguity, we denote the components of $\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c}$ generally as θ_k.

The general form of the gradient expectation is the following (see Eq. 10 in the main text):

$$\int_\Theta P(\theta \mid \boldsymbol{v}, \boldsymbol{h}; \alpha) \frac{\partial \phi(\theta; \alpha)}{\partial \alpha} d\theta. \tag{B1}$$

Now, we start with the Bernoulli case, where $\alpha_k = (\theta_k, p_k)$. The prior distribution and the ϕ potential have the following form:

$$P(\theta_k; \alpha_k) \equiv e^{-\phi(\theta_k; \alpha_k)} = \delta(\theta_k)(1 - p_k) + \delta(\theta_k - \bar{\theta}_k)p_k,$$
$$\frac{\partial \phi}{\partial p_k} = \frac{\delta(\theta_k) - \delta(\theta_k - \bar{\theta}_k)}{e^{-\phi(\theta_k; \alpha_k)}}, \quad \frac{\partial \phi}{\partial \bar{\theta}_k} = \frac{\delta'(\theta_k - \bar{\theta}_k)p_k}{e^{-\phi(\theta_k; \alpha_k)}}, \tag{B2}$$

where $\delta(\cdot)$ is the Dirac delta function, and $\delta'(\cdot)$ is its derivative. When we plug (A2) and (B2) into (B1), we obtain the following expressions for the expectation over W_{ij} (similar for b_i and c_i):

$$\mathbb{E}\left[\frac{\partial \phi}{\partial p_{ij}} \mid v_i, h_i \right] = \frac{1 - p_{ij}}{1 - p_{ij} + p_{ij} \exp(v_i h_i \bar{W}_{ij})} \left(1 - e^{v_i h_i \bar{W}_{ij}} \right),$$
$$\mathbb{E}\left[\frac{\partial \phi}{\partial \bar{W}_{ij}} \mid v_i, h_i \right] = \frac{p_{ij}}{1 - p_{ij} + p_{ij} e^{v_i h_i \bar{W}_{ij}}} \int \delta'(W_{ij} - \bar{W}_{ij}) \exp(v_i h_i W_{ij}) dW_{ij} \tag{B3}$$

$$= [\text{integrating by parts}] = \frac{p_{ij} e^{v_i h_i \bar{W}_{ij}}}{1 - p_{ij} + p_{ij} \exp(v_i h_i \bar{W}_{ij})} (-v_i h_i).$$

Eventually, we get the following expressions for the gradient components of the log-likelihood:

$$\frac{\partial \log P(\boldsymbol{v}; \alpha)}{\partial \bar{W}_{ij}} = \langle P(W_{ij} \neq 0 \mid v_i, h_i)\, v_i h_i \rangle_{\text{data}} - \langle P(W_{ij} \neq 0 \mid v_i, h_i)\, v_i h_i \rangle_{\text{recon}},$$
$$\frac{\partial \log P(\boldsymbol{v}; \alpha)}{\partial \bar{p}_{ij}} = \left\langle P(W_{ij} = 0 \mid v_i, h_i)\left(1 - e^{v_i h_i \bar{W}_{ij}} \right) \right\rangle_{\text{data}} - \tag{B4}$$
$$\left\langle P(W_{ij} = 0 \mid v_i, h_i)\left(1 - e^{v_i h_i \bar{W}_{ij}} \right) \right\rangle_{\text{recon}}.$$

The gradient over $\bar{W}_{ij}$ resembles the original contrastive divergence but has additional posterior probability multipliers.

Similarly, we do the same derivations for the Gaussian case, where $\alpha_k = (\bar{\theta}_k, \sigma_k)$. The prior and the ϕ derivatives have the following form:

$$P(\theta_k; \alpha_k) \equiv e^{-\phi(\theta_k; \alpha_k)} = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left[-\left(\frac{\theta_k - \bar{\theta}_k}{\sqrt{2}\sigma_k} \right)^2 \right],$$
$$\frac{\partial \phi}{\partial \bar{\theta}_k} = \frac{\bar{\theta}_k - \theta_k}{\sigma_k^2}, \quad \frac{\partial \phi}{\partial \sigma_k} = \frac{1}{\sigma_k}\left[1 - \left(\frac{\theta_k - \bar{\theta}_k}{\sigma_k} \right)^2 \right]. \tag{B5}$$

After marginalization, we obtain the following expressions for W_{ij} (similar for b_i and c_j):

$$\mathbb{E}\left[\frac{\partial \phi}{\partial \bar{W}_{ij}} \mid v_i, h_i \right] = -v_i h_i, \quad \mathbb{E}\left[\frac{\partial \phi}{\partial \sigma_{ij}} \mid v_i, h_i \right] = -v_i^2 h_i^2 \sigma_{ij}, \tag{B6}$$

and get the following expressions for the gradient of the log-likelihood components:

$$\frac{\partial \log P(\boldsymbol{v}; \alpha)}{\partial \bar{W}_{ij}} = \langle v_i h_i \rangle_{\text{data}} - \langle v_i h_i \rangle_{\text{recon}},$$
$$\frac{\partial \log P(\boldsymbol{v}; \alpha)}{\partial \bar{\sigma}_{ij}} = \left\langle v_i^2 h_j^2 \sigma_{ij} \right\rangle_{\text{data}} - \left\langle v_i^2 h_j^2 \sigma_{ij} \right\rangle_{\text{recon}}. \tag{B7}$$

Notice that when $\boldsymbol{W}$, $\boldsymbol{b}$, $\boldsymbol{c}$ are deterministic (i.e., $p_{ij} = 0$ and $\sigma_k = 0$ for Bernoulli and Gaussian ensembles, respectively), the expected derivatives over $\bar{W}_{ij}$ (the same is for $\bar{b}_i$ and $\bar{c}_j$) give the same expressions as the standard contrastive divergence. Thus, RBM is the deterministic limit of RBSE.

Another point is related to the implementation: During the optimization, the algorithm steps might be suboptimal, hence we restrict probabilities be in the range $[\epsilon, 1 - \epsilon]$ for some $\epsilon \ll 1$. In other words, we do not allow the model occasionally turn into classic RBM by keeping some ϵ-stochasticity in the connection strengths.