The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Variational Probability Flow for Biologically Plausible Training of Deep Neural Networks

Zuozhu Liu, Tony Q.S. Quek, Shaowei Lin
Singapore University of Technology and Design
8 Somapah Road, Singapore, 487372

Abstract

The quest for biologically plausible deep learning is driven, not just by the desire to explain experimentally-observed properties of biological neural networks, but also by the hope of discovering more efficient methods for training artificial networks. In this paper, we propose a new algorithm named Variational Probably Flow (VPF), an extension of minimum probability flow for training binary Deep Boltzmann Machines (DBMs). We show that weight updates in VPF are local, depending only on the states and firing rates of the adjacent neurons. Unlike contrastive divergence, there is no need for Gibbs confabulations; and unlike backpropagation, alternating feedforward and feedback phases are not required. Moreover, the learning algorithm is effective for training DBMs with intra-layer connections between the hidden nodes. Experiments with MNIST and Fashion MNIST demonstrate that VPF learns reasonable features quickly, reconstructs corrupted images more accurately, and generates samples with a high estimated log-likelihood. Lastly, we note that, interestingly, if an asymmetric version of VPF exists, the weight updates directly explain experimental results in Spike-Timing-Dependent Plasticity (STDP).

Introduction

With the immense success of deep learning in machine learning and artificial intelligence, there has been significant interest in determining the extent to which it explains learning in biological neural networks. For example, Bengio, et al. (2015) have pointed out several challenges in finding biologically plausible implementations of backpropagation, such as the need for precisely-clocked linear feedback paths that have exact knowledge of the weights and the derivatives involved in the feedforward paths. Moreover, backpropagation requires output targets, and passes real-valued signals rather than binary-valued spikes between neurons (Bengio et al. 2015; Rumelhart et al. 1988). Contrastive divergence, on the other hand, trains a binary generative model, but it involves Gibbs sampling to generate data confabulations for the computation of a negative gradient (Hinton 2002). It is unclear how this may be implemented biologically.

Minimally, the update rules for a biologically plausible learning algorithm should be local. More precisely, the change in the weight of a synapse should depend only on information that is directly accessible by the synapse, such as the states and the firing rates of the adjacent neurons. Local rules are also beneficial for machine computation. When the weight updates are not local, such as in the case of backpropagation, it may be necessary to store all the weights and activities of a massive neural network in the memory of the CPU or GPU. Schemes which bypass the need to store all the weights often pay the price with slower convergence to the optimal solution (Jaderberg et al. 2016).

Spike-Timing-Dependent Plasticity (STDP) is another attribute of biological networks (Markram and Sakmann 1995; Gerstner et al. 1996; Bi and Poo 1998; Xie and Seung 2000). This rule states that the change in the weights increases as the time interval between the pre-synaptic and post-synaptic spike decreases, and the sign of the change depends on the temporal ordering of the two spikes. If the pre-synaptic neuron spikes before the post-synaptic neuron, then the weight of the synapse increases, almost as if to increase the probability of the post-synaptic spike at the next pre-synaptic spike. If the order of the spikes is switched, then the synaptic weight decreases. Bengio, et al. (2015) proposed a novel weight update and showed via simulation that it gives rise to STDP. This update rule was then justified using a new learning framework and energy-based objective function.

The main contribution of this paper is the proposal of a new learning algorithm, Variational Probability Flow (VPF), that is biologically plausible in the following ways.

1. The update rule for a given weight w_{ij} is local. It only depends on the binary-valued states y_i, y_j and the real-valued firing rates δ_i, δ_j of the adjacent neurons.
2. An asymmetric version of the update explains STDP.
3. The model neurons are binary-valued, not real-valued.
4. The model is generative so output targets are not required.
5. The updates avoid feedback paths and confabulations.
6. The updates allow intra-layer connections to be trained.

In fact, the asymmetric update that we discovered is

$$\Delta w_{ij} \propto y_i \delta_j$$

where y_i is the pre-synaptic state and δ_j is the post-synaptic firing rate. This form is different from the update proposed

in (Bengio et al. 2015) and (Hinton 2007), where Δw_{ij} depends not on the rate δ_j of post-synaptic transitions but on the *rate of change* of the integrated post-synaptic activity.

As its name suggests, VPF is a variational extension of minimum probability flow (Sohl-Dickstein, Battaglino, and DeWeese 2011; 2009). We derive a variational objective function for training Boltzmann machines (BMs) with observed and hidden variables, and prove that it is an upper bound to the probability flow of the observed variables. We then outline a variational expectation-maximization (EM) algorithm that minimizes this objective (Neal and Hinton 1998). Experiments with restricted Boltzmann machines (RBMs) show that VPF is able to quickly learn features which resemble pen strokes and reconstruct corrupted digits more accurately than previous methods. For deep Boltzmann machines (DBMs), we also propose a new strategy that does not require greedy layer-wise pre-training (Hinton and Salakhutdinov 2006). Moreover, VPF applies directly to BMs with intra-layer connections which significantly improve the model's ability to learn sparse representations. When applied to MNIST and Fashion MNIST data, experiments show that VPF is able to generate realistic samples with estimated log-likelihoods similar to that of generative adversarial networks (GANs). In future work, we will explore how the local learning rules may be exploited to design model-parallel algorithms that run efficiently on GPUs.

Background

Boltzmann Machines

BMs are energy-based probabilistic models (Ackley, Hinton, and Sejnowski 1985). Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ without loops or multiple edges, we associate a binary random variable x_i and a real *bias* b_i to each vertex $i \in \mathcal{V}$, and a real *weight* w_{ij} to each edge $ij \in \mathcal{E}$. Let the probability of each vector $\boldsymbol{x} = (x_i)_{i\in\mathcal{V}}$ be $p(\boldsymbol{x}) \propto \exp(-\frac{1}{T}\text{Energy}(\boldsymbol{x}))$ where $\text{Energy}(\boldsymbol{x}) = -\sum_{ij\in\mathcal{E}} w_{ij}x_ix_j - \sum_{i\in\mathcal{V}} b_ix_i$, and T is the temperature. For simplicity, we fix $T = 1$ in this paper.

In statistical learning, it is useful to designate some of the variables x_i to be observed and the others to be hidden. The probability of each observed state vector is obtained by marginalizing over all the hidden states. In this paper, we use multilayer networks to refer to BMs with a layered structure, where the connections are either within the layers or between consecutive layers. For instance, DBM is a special case with layers $\boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_\ell$ where the first layer $\boldsymbol{h}_0$ is observed while the others are hidden. RBM is special cases of DBMs with $\ell = 1$. In these underlying graph, there are only edges between consecutive layers $\boldsymbol{h}_i$ and $\boldsymbol{h}_{i+1}$. DBMs with intra-layer connections are studied as well.

Minimum Probability Flow

Maximum likelihood estimation can be intractable for many important classes of statistical models. To overcome this problem, Sohl-Dickstein, et el. (2011) designed a new learning objective known as Minimum Probability Flow (MPF) for finite state models which may be sampled effectively using a suitable continuous-time Markov chain (CTMC) (Sohl-Dickstein, Battaglino, and DeWeese 2011; 2009). Let $\boldsymbol{p}^{(t)}(\theta)$ be a CTMC parameterized by θ such that $p^{(t)}(\boldsymbol{x};\theta)$ is the probability of state $\boldsymbol{x}$ at time t in the Markov chain. Let $\boldsymbol{p}^{(0)}(\theta) = \boldsymbol{p}^{(0)}$ denote the empirical distribution of the data set $\mathcal{D} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$ of binary vectors, and we assume the stationary distribution $\boldsymbol{p}^{(\infty)}(\theta)$ is a distribution in the finite state model that we hope to fit to the data.

Using this notation, maximum likelihood estimation is equivalent to minimizing the Kullback-Leibler divergence $D_{\text{KL}}(\boldsymbol{p}^{(0)}\|\boldsymbol{p}^{(\infty)}(\theta))$ over θ. On the contrary, minimum probability flow computes

$$\min_\theta \mathcal{K}(\theta), \quad \mathcal{K}(\theta) := D_{\text{KL}}(\boldsymbol{p}^{(0)}\|\boldsymbol{p}^{(\varepsilon)}(\theta)),$$

for infinitesimal time $\varepsilon > 0$. Given an energy-based model $p^{(\infty)}(\boldsymbol{x};\theta) \propto \exp(-\text{Energy}(\boldsymbol{x};\theta))$ where $\text{Energy}(\boldsymbol{x};\theta)$ is the energy of state $\boldsymbol{x}$, we may construct a CTMC $\boldsymbol{p}^{(t)}(\theta) = \exp(\boldsymbol{\Gamma}(\theta)t)\boldsymbol{p}^{(0)}$ that attains this stationary distribution by defining its transition rate matrix $\boldsymbol{\Gamma} = (\Gamma_{xy})$ to be

$$\Gamma_{xy} = g_{xy}\exp\left(\frac{1}{2}\text{Energy}(\boldsymbol{y};\theta) - \frac{1}{2}\text{Energy}(\boldsymbol{x};\theta)\right), \boldsymbol{x} \neq \boldsymbol{y},$$

where $\Gamma_{xx} = -\sum_{\boldsymbol{y}\neq\boldsymbol{x}} \Gamma_{xy}$ and each $g_{xy} \in \{0, 1\}$ is some choice of connectivity between states $\boldsymbol{x}$ and $\boldsymbol{y}$. In this case, one can show that for small $\varepsilon > 0$,

$$\begin{aligned}\mathcal{K}(\theta) &= \frac{\varepsilon}{N}\sum_{\boldsymbol{x}\notin\mathcal{D},\boldsymbol{y}\in\mathcal{D}} \Gamma_{xy} \\ &= \frac{\varepsilon}{N}\sum_{\boldsymbol{x}\notin\mathcal{D},\boldsymbol{y}\in\mathcal{D}} g_{xy}\exp\left(\frac{1}{2}\text{Energy}(\boldsymbol{y};\theta) - \frac{1}{2}\text{Energy}(\boldsymbol{x};\theta)\right).\end{aligned}$$

Therefore, MPF is a learning framework for energy-based models that avoids computing the intractable partition function and its derivatives.

Variational Probability Flow

In this section, we derive VPF for hidden variable models by applying variational principles to probability flow.

Fully-Observed Boltzmann Machines

We first apply MPF to the training of fully-observed Boltzmann machines, by defining the connectivity g_{xy} to be 1 if and only if the states $\boldsymbol{x}$ and $\boldsymbol{y}$ are one-hop neighbors, i.e. they differ only by one bit. Given training data $\mathcal{D} = \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}\}$, the MPF objective function becomes

$$\mathcal{K}(\theta) = \frac{\varepsilon}{N}\sum_{k=1}^{N}\mathcal{K}^{(k)}(\theta), \quad \mathcal{K}^{(k)}(\theta) = \sum_{j=1}^{|\mathcal{V}|}\delta_j^{(k)},$$

where for convenience we define

$$\alpha_j^{(k)} = \tfrac{1}{2} - y_j^{(k)}, z_j^{(k)} = \textstyle\sum_{i\neq j} w_{ij}y_i^{(k)} + b_j,$$

$$\delta_j^{(k)} = \exp(\alpha_j^{(k)}z_j^{(k)}).$$

To update the parameters, we use the negative gradients

$$\Delta b_i \propto -\alpha_i^{(k)}\delta_i^{(k)}, \tag{1}$$

$$\Delta w_{ij} \propto - y_j^{(k)} \alpha_i^{(k)} \delta_i^{(k)} - y_i^{(k)} \alpha_j^{(k)} \delta_j^{(k)}. \quad (2)$$

We note that $\delta_j^{(k)}$ is the transition rate Γ_{xy} where $\boldsymbol{y} = \boldsymbol{y}^{(k)}$ and $\boldsymbol{x}$ is the one-hop neighbor that differs in the j-th bit. In other words, it is the firing rate of a Poisson process that determines the flipping of the state of the j-th vertex. Please refer to the supplementary material for detailed proofs.

VPF Objective Function

Consider a Boltzmann machine with observed and hidden variables $\boldsymbol{x}, \boldsymbol{h}$. Let $\tilde{\mathcal{D}} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$ be IID samples of $\boldsymbol{x}$, and $\tilde{\boldsymbol{p}}^{(0)}$ be the corresponding empirical distribution. Let $p^{(t)}(\boldsymbol{x}, \boldsymbol{h}; \theta)$ be the associated CTMC. Our goal is to minimize the marginal probability flow $D_{\mathrm{KL}}(\tilde{\boldsymbol{p}}^{(0)} \| \tilde{\boldsymbol{p}}^{(\varepsilon)}(\theta))$ where $\tilde{p}^{(t)}(\theta)$ represents the marginal probability $p^{(t)}(\boldsymbol{x}; \theta)$. Let the VPF objective function be

$$\mathcal{L}^{(t)}(q, \theta) = D_{\mathrm{KL}}(q(\boldsymbol{h}, \boldsymbol{x}) \| p^{(t)}(\boldsymbol{h}, \boldsymbol{x}; \theta))$$

where $q := q(\boldsymbol{h}|\boldsymbol{x})$ is some distribution over the hidden variables given the observed variables, and $q(\boldsymbol{x}, \boldsymbol{h})$ denotes $q(\boldsymbol{h}|\boldsymbol{x})\tilde{p}^{(0)}(\boldsymbol{x})$. As shown in the supplementary material,

$$\mathcal{L}^{(t)}(q, \theta) = \mathbb{E}_{\boldsymbol{x} \sim \tilde{p}^{(0)}(\boldsymbol{x})} \left[D_{\mathrm{KL}}(q(\boldsymbol{h}|\boldsymbol{x}) \| p^{(t)}(\boldsymbol{h}|\boldsymbol{x}; \theta)) \right] + D_{\mathrm{KL}}(\tilde{\boldsymbol{p}}^{(0)} \| \tilde{\boldsymbol{p}}^{(t)}(\theta)). \quad (3)$$

Therefore, if we hold θ constant and minimize $\mathcal{L}^{(t)}(q, \theta)$ over all conditional distributions q, then the minimum occurs when $q(\boldsymbol{h}|\boldsymbol{x}) = p^{(t)}(\boldsymbol{h}|\boldsymbol{x}; \theta)$. However, it is not possible to compute the latter distribution unless the joint distribution $p^{(t)}(\boldsymbol{x}, \boldsymbol{h})$ at time $t = 0$ is known. We will therefore approximate it by assuming that the CTMC is close to the stationary distribution $p^{(\infty)}(\boldsymbol{x}, \boldsymbol{h})$, and select $q(\boldsymbol{h}|\boldsymbol{x}) = p^{(\infty)}(\boldsymbol{h}|\boldsymbol{x}; \theta)$. This approximation implies that $\mathcal{L}^{(t)}(q, \theta)$ is an upper bound to $D_{\mathrm{KL}}(\tilde{\boldsymbol{p}}^{(0)} \| \tilde{\boldsymbol{p}}^{(t)}(\theta))$, and the bound improves as $p^{(t)}(\boldsymbol{h}|\boldsymbol{x}; \theta)$ tends to the model $p^{(\infty)}(\boldsymbol{h}|\boldsymbol{x}; \theta)$.

If it is biologically desirable to perform the above optimizations while constraining the weights w_{ij} to a bounded space, one could do so by adding a weight decay term to the objective functions, e.g., $\min_{q,\theta} \mathcal{L}^{(t)}(q, \theta) + \lambda \sum_{ij} w_{ij}^2$.

VPF Learning Algorithm

The objective function is optimized with a variational EM algorithm. The E-step and M-step are repeated until convergence of the parameters θ of the Boltzmann machine.

E-step. We set $q(\boldsymbol{h}, \boldsymbol{x}) = p^{(\infty)}(\boldsymbol{h}|\boldsymbol{x}; \theta)\, \tilde{p}^{(0)}(\boldsymbol{x})$ and compute $\mathcal{L}^{(\varepsilon)}(q, \theta)$ given this choice. We further approximate

$$\mathcal{L}^{(\varepsilon)}(q, \theta) = \int q(\boldsymbol{x}, \boldsymbol{h}) \log q(\boldsymbol{x}, \boldsymbol{h})\, d\boldsymbol{x} d\boldsymbol{h} - \int q(\boldsymbol{x}, \boldsymbol{h}) \log p^{(\varepsilon)}(\boldsymbol{h}, \boldsymbol{x}; \theta)\, d\boldsymbol{x} d\boldsymbol{h} \quad (4)$$

by substituting the latter integral with an average over samples from $q(\boldsymbol{x}, \boldsymbol{h})$. By optimizing the resulting stochastic objective $\mathcal{S}(\theta)$, the parameters will still converge to the same limit as that of $\mathcal{L}^{(\varepsilon)}(q, \theta)$ (Gal and Ghahramani 2015).

Algorithm 1 VPF for training a BM.

```
x ← training data
w, b ← random initialization
while (w, b) not converged do
    for i = 1 to ℓ do
        h_i ← Bernoulli sample of h_i given h_{i−1}
        if intra-layer connections exist then
            Asynchronously update h_i following Eq. 5
        end if
    end for
    y ← (x, h)
    for minibatch y_m in y do
        Δw, Δb ← MPF updates at w, b for y_m
        Update parameters w, b with Δw, Δb using Adam optimizer
    end for
end while
return w, b
```

Algorithm 2 Generating confabulations from a given BM.

```
h_ℓ ← random initialization
for i = ℓ to 1 do
    for j = 1 to r do
        h_{i−1} ← Bernoulli sample of h_{i−1} given h_i
        h_i ← Bernoulli sample of h_i given h_{i−1}
        if intra-layer connections exist then
            Asynchronously update h_i following Eq. 5
        end if
    end for
end for
return Bernoulli probability of h_0 given h_1
```

To sample from the conditional distribution $p^{(\infty)}(\boldsymbol{h}|\boldsymbol{x}; \theta)$ of the Boltzmann machine, the hidden states could be initialized randomly before applying Gibbs sampling. For the RBM and the DBM, we will discuss approximate sampling schemes in next section. From our experiments, we observe that it is enough to generate just one sample to achieve good convergence of the learning algorithm.

M-step. We minimize the stochastic objective $\mathcal{S}(\theta)$. In the E-step, we obtained samples of $\boldsymbol{h}$ given each data point $\boldsymbol{x}$, giving us the set $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, \boldsymbol{h}^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, \boldsymbol{h}^{(N)})\}$. The objective $\mathcal{S}(\theta)$ is precisely the probability flow of a fully-observed Boltzmann machine with data $\mathcal{D}$, so Eq. 1, 2 may be applied to update the parameter θ.

VPF for Multilayer Networks

The variational EM algorithm allows us to train any BMs with hidden variables. In this section, we discuss strategies for improving its performance on BMs where the hidden variables have a layered structure.

RBM. For RBMs, the hidden variables $\boldsymbol{h}$ are conditionally independent given the observed variables $\boldsymbol{x}$ with

$$p^{(\infty)}(h_j = 1|\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{b}) = \mathrm{sigmoid}(\textstyle\sum_i w_{ij} x_i + b_j).$$

These Bernoulli probabilities can be computed and sampled efficiently on a machine. The pseudocode for applying VPF

to RBMs is given in Algorithm. 1 with $\ell = 1$. Stochastic gradient descent (SGD) is exploited during training but with a minor change. Traditionally, when SGD is performed in deep learning, the parameters are updated after each mini-batch. In VPF, at the start of each epoch, we perform the E-step for the entire data set, sampling the hidden states for each data point using the parameters attained at the end of the last epoch. The observed-hidden pairs are then partitioned into mini-batches for the M-step. This procedure seems to promote convergence to better local minima.

DBM. For DBMs with $\ell \geq 2$ hidden layers, the hidden variables $\boldsymbol{h}$ are no longer conditionally independent given the observed variables $\boldsymbol{x}$. One strategy for training a DBM is to train each pair $(\boldsymbol{h}_{i-1}, \boldsymbol{h}_i)$ of layers as an RBM, starting from $\boldsymbol{h}_0$ and generating data for $\boldsymbol{h}_i$ after $(\boldsymbol{h}_{i-1}, \boldsymbol{h}_i)$ has been trained. Here, we propose a different strategy that does not require greedy layer-wise training. We approximate the conditional distribution $p^{(\infty)}(\boldsymbol{h}|\boldsymbol{x}; \boldsymbol{w}, \boldsymbol{b})$ by computing Bernoulli probabilities for $\boldsymbol{h}_i$ using only the weighted inputs from $\boldsymbol{h}_{i-1}$ while zeroing out the inputs from $\boldsymbol{h}_{i+1}$. We sample $\boldsymbol{h}_i$ from these probabilities before proceeding in the same way to sample $\boldsymbol{h}_{i+1}$. The generated hidden states are then used for the M-step where the weights for every layer are updated at the same time. The pseudocode for training a DBM is shown in Algorithm. 1.

To generate confabulations from a given DBM, we start at the top layer with a binary vector $\boldsymbol{h}_\ell$ that was picked uniformly or from some priors. We perform r Gibbs sampling steps, alternating between $\boldsymbol{h}_\ell$ and $\boldsymbol{h}_{\ell-1}$. In our experiments, it seems that $r = 5$ was enough to ensure good mixing. The final state of $\boldsymbol{h}_{\ell-1}$ is then used to initialize the next r Gibbs sampling steps for generating $\boldsymbol{h}_{\ell-2}$, and so on. The pseudocode for this process is given in Algorithm. 2.

DBM with Intra-layer Connections. We introduce how to employ VPF to train DBMs with intra-layer connections, which can be viewed as a stack of RBMs but with connections between the hidden nodes, see Fig.5.(a). To train these generalized DBMs, we take an additional step of performing asynchronous Gibbs updates for the hidden units $\boldsymbol{h}_i$ given the values of $\boldsymbol{h}_{i-1}$, $\boldsymbol{h}_i$ in the previous update. More specifically, we update the neurons in $\boldsymbol{h}_i$ one by one in a fixed order, sampling the j-th neuron h_{ij} from the probability

$$p_{ij} = \text{sigmoid}(\textstyle\sum_{k \neq j} w_{kj} h_{ik} + \sum_k w'_{kj} h_{(i-1)k} + b_j). \quad (5)$$

where $\boldsymbol{w}, \boldsymbol{w'}$ denote the intra- and inter-layer weights respectively. Updates for the intra-layer weights also follow the local rules Eq. 1, 2. The confabulation-generating procedure has an additional step involving intra-layer Gibbs updates. The modifications are indicated in Algorithm. 1, 2.

As with traditional associative memories, intra-layer connections help the model to learn rich energy landscapes and sparse distributed representations which are otherwise impossible for disconnected layers with factorial distributions. By embracing them in VPF, we hope to learn, from the observed data, hidden priors with greater explanatory power. Usually, one assumes the absence of connections within each layer so as to derive tractable training algorithms. VPF overcomes this limitation, through the combination of probability flow with variational techniques. In our experiments, we explore the effect that these intra-layer connections have on the model's performance in generative tasks.

Spike-Timing-Dependent Plasticity

STDP is a phenomenon observed in biological neural networks, but little is understood about how it contributes to learning. In this section, we show how VPF could give rise to STDP. First, we present a simplified model of learning in biological neural networks. Each neuron y_i in our model are either in an excited state $y_i = 1$ or in a resting state $y_i = 0$ (Amari 1977). The time it takes for an excited neuron to transit to a resting state follows an exponential distribution with rate $\delta_i = \exp(-z_j)$, while the time for a resting neuron to become excited occurs with rate $\delta_i = \exp(+z_j)$. The synaptic weight w_{ij} is updated if the post-synaptic neuron j transits and if the pre-synaptic neuron i is excited. The change in value is proportional to

$$\Delta w_{ij} \propto -y_i \alpha_j \delta_j, \quad (6)$$

where δ_j is the post-synaptic rate before the transition, y_j is the post-synaptic state after the transition, and $\alpha_j = 1/2 - y_j = \pm 1/2$ only affects the sign of the update. This update rule is local, in the sense that it depends only on the states y_i, y_j and the firing rates δ_i, δ_j of the adjacent neurons. The weights $w_{ij} = w_{ji}$ are symmetric in BMs (Scellier and Bengio 2017), so we combine the updates for Δw_{ij} and Δw_{ji} from Eq. 6, which gives the VPF update Eq. 2.

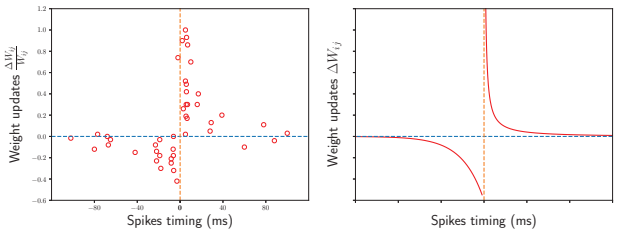
We now show that update Eq. 6 explains STDP. Suppose that we have a pre-synaptic spike before a post-synaptic spike. Specifically, at time $t = 0$, we have $y_i(t) = 1$ and $y_j(t) = 0$, and at some $t = \varepsilon$, we have $y_j(t) = 1$. There are two cases to consider: either the pre-synaptic neuron is still excited and $y_i(\varepsilon) = 1$, or the pre-synaptic neuron has returned to resting state and $y_i(\varepsilon) = 0$. The first case occurs with probability $\exp(-\delta_i \varepsilon)$ and the weight update is $\Delta w_{ij} \propto \delta_j$. Here, $\delta_j = 1/\varepsilon$ because the expected time to transition is the inverse of the transition rate. The second case occurs with probability $1 - \exp(-\delta_i \varepsilon)$, but the weight update is 0 because $y_i(\varepsilon) = 0$. Thus, the expected weight change is

$$\Delta w_{ij} \propto (1/\varepsilon) \exp(-\delta_i \varepsilon).$$

Suppose instead that the pre-synaptic spike happens after a post-synaptic spike: $y_i(0) = 0, y_j(0) = 1$, and $y_i(\varepsilon) = 1$. Again, there are two cases: either the post-synaptic neuron is still excited and $y_j(\varepsilon) = 1$, or the post-synaptic neuron has returned to resting state and $y_j(\varepsilon) = 0$. The first case occurs with probability $\exp(-\delta_j \varepsilon)$. After the pre-synaptic spike, there is no immediate weight change, until the first post-synaptic flip to state 0 takes place. The update is $\Delta w_{ij} \propto -\delta_j$ for this first flip. The second case happens with probability $1 - \exp(-\delta_j \varepsilon)$. However, in the absence of stimuli, the post-synaptic neuron does not become excited again before the pre-synaptic neuron returns to rest. Hence, the update is 0. Overall, the average weight change is

$$\Delta w_{ij} \propto -\delta_j \exp(-\delta_j \varepsilon).$$

Plotting the two average weight updates derived above in Fig.1, we see a good fit with the experimental measurements



(a) STDP function (b) Weight updates under VPF

Figure 1: **(a)**. The trend in STDP, reproduced from (Bi and Poo 1998). **(b)**. A visualization of how expected weight updates are inversely proportional to time intervals between the spikes, as predicted by VPF.

for STDP. In future, we hope to find a statistical model that, unlike BMs, does not require symmetry, and whose update rule under VPF is precisely given by Eq. 6.

Experiments

We conducted experiments on MNIST digits and Fashion MNIST images. MNIST is binarized by thresholding with 0.5, and Fashion MNIST is scaled to $[0, 1]$ before thresholding. The Adam optimizer is used during optimization with default values for the parameters, e.g., $\beta_1 = 0.9$, $\beta_2 = 0.999$ (Kingma and Ba 2014). The learning rate is $\eta = 0.001$. The weight decay parameter λ is set to 0.0001 for all the RBMs and DBMs. We used a mini-batch size of $M = 40$, and the code is implemented with Theano (Team et al. 2016). More details can be found in the supplementary material.

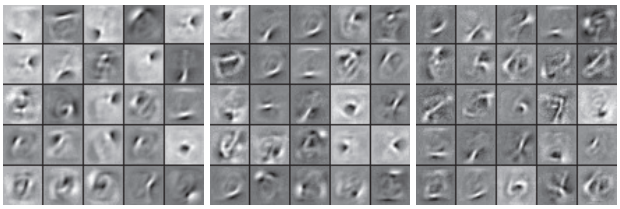
Restricted Boltzmann Machines

In the first experiment, we trained three RBMs: RBM(100), RBM(196), RBM(400), using Algorithm. 1 with $\ell = 1$, where 100, 196, 400 denote the number of hidden units. Fig. 2 shows 25 randomly selected weight filters of the RBMs. All of the models learned reasonable filters (pen strokes), which are sharper in RBM(196) and RBM(400). We also noticed that the training objective converged quickly. As shown in Fig.3.(a), all the curves flattened out in less than 50 epochs.

To verify that VPF learns a distributed representation, we plotted a histogram of the mean activations of hidden units, which we averaged over 10,000 training examples. As shown in Fig.3.(b), majority of the hidden units have mean activations between 0.1 and 0.4. The overall mean activation is 0.26, which is reasonably sparse for MNIST data. We also tracked two other metrics, the weight sparsity ρ and the squared weight w^2, which are given by

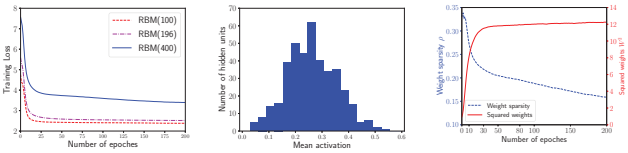
$$\rho = \frac{1}{|\boldsymbol{x}||\boldsymbol{h}|}\sum_j[(\sum_i w_{ij}^2)^2/\sum_i w_{ij}^4], \quad w^2 = \frac{1}{|\boldsymbol{h}|}\sum_{ij} w_{ij}^2,$$

where $|\boldsymbol{x}|, |\boldsymbol{h}|$ are the dimensions of $\boldsymbol{x}, \boldsymbol{h}$, and w_{ij} is the weight from visible unit i to hidden unit j. The weight sparsity ρ is a rough measure of the number of nonzero weight parameters. It decreased to 0.15 during training, as shown in Fig.3.(c), which suggests that many weights are near zero.



(a) RBM(100) (b) RBM(196) (c) RBM(400)

Figure 2: Randomly selected filters learned from MNIST.



(a) (b) (c)

Figure 3: **(a)**. The training loss of different RBMs. **(b)**. Histogram of mean activations of RBM(400). **(c)**. Evolution of the weight sparsity and squared weight of RBM(400).

The remaining nonzero weights grew in value during training, as may be seen in the increasing squared weight w^2 in Fig.3.(c) and the high-contrast features in Fig. 2. These results demonstrate that RBMs trained with VPF operate in a compositional phase with distributed representations (Tubiana and Monasson 2017).

Image Reconstruction with RBMs

We explored the expressiveness of the VPF model by comparing RBM(196) to RBMs trained with CD and persistent contrastive divergence (PCD) in terms of the ability to reconstruct corrupted parts of MNIST digits. To reconstruct the images, we run Markov chains starting from the corrupted images and performed 1000 Gibbs transitions for CD and PCD to make them mix well. For RBM(196), we found that 2 Gibbs transitions were enough for good results. Codes for CD and PCD were derived from (Team et al. 2016).

From Fig. 4 we can see that RBM(196) is able to reconstruct the initially corrupted pixels under 4 different corruption patterns. The reconstruction errors were measured as the $\mathcal{L}$-1 norm of the pixel differences between the original image $\boldsymbol{x}$ and reconstructed image $\hat{\boldsymbol{x}}$, i.e., $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_1$. Using the same number of hidden units, the errors for various training methods, which were averaged over 10,000 MNIST test samples and 3 independent experiments, are reported in Table. 1. As we can see, VPF performs much better than CD-1, CD-10 and PCD-1. It is also better than PCD-10 in most of the cases. Although PCD-10 is slightly better than VPF in the Right case, it takes a longer time in doing 10 Gibbs transitions during training and 1000 during reconstruction.

Deep Boltzmann Machines

The performance of VPF to train DBMs is evaluated on two datasets: MNIST and Fashion MNIST. For each dataset, we

Table 1: Average reconstruction error for 10,000 corrupted MNIST digits. 12 rows or columns out of 28 are randomly corrupted under 4 patterns, e.g., Top means top 12 rows corrupted. We tested 1 or 10 Gibbs updates for CD and PCD.

Corruption	CD-1	CD-10	PCD-1	PCD-10	VPF
Top	57.8	55.7	57.6	47.9	**32.9**
Bottom	74.7	54.5	58.2	48.9	**46.9**
Left	59.2	58.2	46.2	39.1	**36.7**
Right	69.9	55.4	53.6	**43.7**	44.1

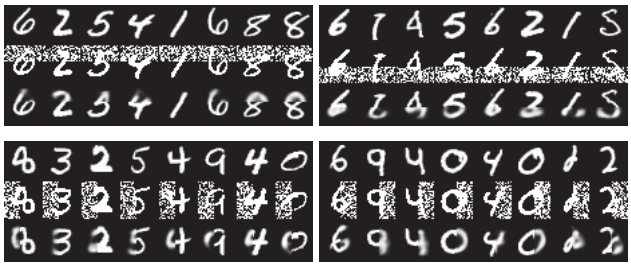


Figure 4: Examples of reconstructing the corrupted MNIST digits by RBM(196). Top row: original digits, middle row: corrupted digits, bottom row: reconstructed digits.

trained two DBMs with the same architecture 784-196-196-64 based on Algorithm. 1. The first DBM (DBM1) has no intra-layer connections while the second one (DBM2) does. See Fig. 5.(a) for an illustration.

Algorithm. 2 is essentially a top-down approach for generating confabulations. To first initialize the top hidden layer $\boldsymbol{h}_\ell$, we experimented with two different strategies: (1) Randomly set states in $\boldsymbol{h}_\ell$ to be 0 or 1 with equal probability (denoted as DBM-R), (2) Initialize $\boldsymbol{h}_\ell$ with a Bernoulli prior which is computed as the mean feedforward activations of the training dataset (denoted as DBM-P). We conjecture that in the absence of intra-layer connections, the bottom-up signals are crucial for initializing the factorial priors in a good neighborhood. This second strategy was used by (Bengio et al. 2015) as well for the generative task.

Confabulations generated from the trained DBMs using Algorithm. 2 are displayed in Fig. 5 and Fig. 6. We also estimated the log-likelihood of 10,000 MNIST test samples by fitting a Gaussian Parzen density window to 10,000 samples generated by the learned DBMs (Goodfellow et al. 2014). The standard deviation for the Parzen density estimator is chosen as 0.2 using a validation set following the procedure in (Desjardins et al. 2010). The estimated log-likelihoods are reported in Table. 2.

MNIST. From Table. 2 we clearly see that DBM1-P and DBM2-P are able to achieve log-likelihoods $LL = 210$ or $LL = 222$, which are much higher than RBM(196) ($LL = 78$) because of the using of the high layers. They also perform better than many previous generative models such as deep Boltzmann machines (DBM, $LL = 32$) (Bengio et al. 2014), deep belief networks (DBN, $LL = 138$) (Bengio et al. 2013) and stacked contractive auto-encoders (Stacked-CAE, $LL = 121$) (Bengio et al. 2013). While the log-likelihood of DBM1-P is slightly worse than that of deep generative stochastic nets (Deep-GSN, $LL = 214$) (Bengio et al. 2014), DBM2-P does better, thanks to its intra-layer connections. In fact, the log-likelihood of DBM2-P is quite close to that of generative adversarial nets (GAN, $LL = 225$) (Goodfellow et al. 2014), and not far away from (Bengio et al. 2015) ($LL = 236$), which used a bottom-up signal to initialize the top layer.

As seen in Fig. 5, trained DBMs generate realistic digit samples. We also note that DBM2 produced better samples than DBM1 (higher LL). While DBM1-R only achieved a log-likelihood of 183, DBM2-R achieved $LL = 222$ which outperforms DBM1-P and is as good as DBM2-P. This is consistent with our conjecture that intra-layer connections enable the learning of sparse distributed representations and highly-explanatory latent priors, so bottom-up signals are not required. Meanwhile, we found that DBM1 often produced digits with undesirable gaps, see Fig.5.(b), which lower the log-likelihood. We postulate that the gaps are due to DBMs exploring a factorial prior, a phenomenon commonly observed in GAN (Goodfellow et al. 2014). This drawback is resolved through the use of intra-layer connections which enable non-factorial priors. Indeed, there are no obvious gaps in digits from DBM2-R and DBM2-P.

Fashion MNIST. We trained DBMs on Fashion MNIST, a data set that contains ten kinds of fashion images from Zalando (Xiao, Rasul, and Vollgraf 2017). As we can see from Fig. 6, the learned DBMs generate meaningful samples such as T-shirts, coats, and ankle boots, all without any other tricks for the learning procedure or hyper-parameters. The generated samples do not look as good as the samples for MNIST. One likely reason is that thresholding damages the Fashion MNIST data significantly. Samples from DBM1-R and DBM2-R are not shown here due to space constraints.

Related Work & Discussion

Our work is highly related to that of (Bengio et al. 2015). We both are able to train deep generative models without backpropagation, using variational ideas while making sense of STDP. However, our work differs from (Bengio et al. 2015) in two aspects. First, we use different weight update rules to explain STDP, as stated in the introduction. Second, we employ a different strategy to avoid backpropagation, i.e., we use the MPF update rule instead of target propagation.

Another similar work which also explores STDP using energy-based models is equilibrium propagation (Scellier and Bengio 2017). In equilibrium propagation, the objective function is the sum over the data of $E + \beta C$ where E is the energy of the network, C is the loss between a model prediction and an output target, and β a trade-off parameter. VPF is a generative rather than discriminative model, so it does not aim to minimize C. Instead of minimizing the network energy E, it optimizes the sum of $\exp(E(\boldsymbol{y})/2 - E(\boldsymbol{x})/2)$ over all one-hop neighbors $\boldsymbol{y}$ of a data point $\boldsymbol{x}$. This objective is a kind of exponential transition (kinetic) energy rather than a form of state (potential) energy.

We compare VPF with existing methods to train RBMs or

Table 2: Log-likelihood and standard error estimated with Parzen density window. The reference measure obtained from MNIST training dataset is 239.88 ± 2.3 (Desjardins et al. 2010).

DBM	DBN	Stacked-CAE	Deep GSN	GAN	Bengio et al.	RBM(196)	DBM1-R	DBM1-P	DMB2-R	DBM2-P
32±2	138±2	121±1.6	214±1.1	225±2	**236**	78±5.9	183±1.1	210±0.8	222±0.5	222±0.4

(a) DBM1 (left) & DBM2 (right) (b) DBM1-P (c) DBM2-R (d) DBM2-P

Figure 5: **(a)**. Different DBMs: left is traditional DBM, right is DBM with intra-layer connections. **(b-d)**. Digits generated from DBMs with different initialization for the top hidden layer, not cherry-picked. E.g., DBM2-P refers to a DBM with intra-layer connections, generating samples with a mean activation prior. Greyscale images represent the probabilities for the visible units.

(a) Train samples (b) DBM1-P (c) DBM2-P

Figure 6: Generation of fashion images, not cherry-picked. Training samples are binarized by thresholding with 0.5.

DBMs. The differences between VPF and CD are: (1) VPF has an explicit objective function, (2) VPF does not require Gibbs sampling to generate confabulations to compute the gradient, (3) VPF gives rise to STDP. As for training DBMs, instead of employing a two-stage process, i.e., pre-training a stack of RBMs or a separate recognition model to initialize a DBM and then updating it (Salakhutdinov and Hinton 2009; Salakhutdinov and Larochelle 2010), VPF learns all the parameters simultaneously without greedy layer-wise initialization of the weights and does not require data confabulations. Moreover, VPF trains neural nets with intra-layer connections, which is hard to achieve in conventional methods.

Recently, many groups have worked on embedding STDP into deep learning for biologically plausible training of neural nets, e.g., (Rezende, Wierstra, and Gerstner 2011) found STDP-like behavior in recurrent spiking networks, (Bengio et al. 2017) proposed a framework for training energy-based models without explicit backpropagation. VPF is yet another stab in this direction. Even with these advancements, there is still more to be done. For instance, a key question is whether it is possible or even necessary to do away with symmetric weights in energy-based models, so as to achieve biological plausibility. Some studies suggest that methods such as batch-normalization could help ameliorate this "weight transport problem" (Liao, Leibo, and Poggio 2016).

Conclusion & Future Work

In this paper, we derived a new training objective and learning algorithm known as Variational Probability Flow, which gives rise to biologically plausible training of deep neural networks. There are three main directions for future work. Firstly, we are interested in extending VPF to recurrent spiking networks which we believe are closer to biological networks. We hope that by doing so, we also uncover a directed version of VPF whose weight updates are given simply by $\Delta w_{ij} \propto -y_i \alpha_j \delta_j$ to eliminate the requirement for symmetric weights. Secondly, while neurons compute using binary values, real-world data is mostly represented by real values. Therefore, we hope to design natural interfaces between the two data types for more efficient learning. Finally, we will explore how the local learning rules may be exploited to design model-parallel algorithms that run efficiently on GPUs. Such algorithms may enable us to train massive neural networks with tens of billions of neurons in a distributed energy-efficient manner.

Acknowledgement

Shaowei Lin is funded by SUTD grant SRES15111 and SUTD-ZJU grant ZJURP1600103. Zuozhu Liu is supported by SUTD President's Graduate Fellowship. We would like to thank NVIDIA for their computational support. We also thank Christopher Hillar, Sai Ganesh, Binghao Ng, Dewen Soh, Thiparat Chotibut, Gary Phua, Zhangsheng Lai for their helpful work and discussions.

References

Ackley, D. H.; Hinton, G. E.; and Sejnowski, T. J. 1985. A learning algorithm for boltzmann machines. *Cognitive science* 9(1):147–169.

Amari, S.-I. 1977. Neural theory of association and concept-formation. *Biological cybernetics* 26(3):175–185.

Bengio, Y.; Mesnil, G.; Dauphin, Y.; and Rifai, S. 2013. Better mixing via deep representations. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 552–560.

Bengio, Y.; Laufer, E.; Alain, G.; and Yosinski, J. 2014. Deep generative stochastic networks trainable by backprop. In *International Conference on Machine Learning*, 226–234.

Bengio, Y.; Lee, D.-H.; Bornschein, J.; Mesnard, T.; and Lin, Z. 2015. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Bengio, Y.; Mesnard, T.; Fischer, A.; Zhang, S.; and Wu, Y. 2017. Stdp-compatible approximation of backpropagation in an energy-based model. *Neural computation*.

Bi, G.-q., and Poo, M.-m. 1998. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience* 18(24):10464–10472.

Desjardins, G.; Courville, A.; Bengio, Y.; Vincent, P.; and Delalleau, O. 2010. Parallel tempering for training of restricted boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 145–152. MIT Press Cambridge, MA.

Gal, Y., and Ghahramani, Z. 2015. Dropout as a bayesian approximation: Insights and applications. In *Deep Learning Workshop, ICML*.

Gerstner, W.; Kempter, R.; van Hemmen, J. L.; and Wagner, H. 1996. A neuronal learning rule for sub-millisecond temporal coding. *Nature* 383(6595):76.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

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

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

Hinton, G. 2007. How to do backpropagation in a brain. In *Invited talk at the NIPS 2007 Deep Learning Workshop*, volume 656.

Jaderberg, M.; Czarnecki, W. M.; Osindero, S.; Vinyals, O.; Graves, A.; and Kavukcuoglu, K. 2016. Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Liao, Q.; Leibo, J. Z.; and Poggio, T. A. 2016. How important is weight symmetry in backpropagation? In *AAAI*, 1837–1844.

Markram, H., and Sakmann, B. 1995. Action potentials propagating back into dendrites trigger changes in efficacy of single-axon synapses between layer v pyramidal neurons. In *Soc. Neurosci. Abstr*, volume 21, 2007.

Neal, R. M., and Hinton, G. E. 1998. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*. Springer. 355–368.

Rezende, D. J.; Wierstra, D.; and Gerstner, W. 2011. Variational learning for recurrent spiking networks. In *Advances in Neural Information Processing Systems*, 136–144.

Rumelhart, D. E.; Hinton, G. E.; Williams, R. J.; et al. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5(3):1.

Salakhutdinov, R., and Hinton, G. 2009. Deep boltzmann machines. In *Artificial Intelligence and Statistics*, 448–455.

Salakhutdinov, R., and Larochelle, H. 2010. Efficient learning of deep boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 693–700.

Scellier, B., and Bengio, Y. 2017. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience* 11.

Sohl-Dickstein, J.; Battaglino, P.; and DeWeese, M. R. 2009. Minimum probability flow learning. *arXiv preprint arXiv:0906.4779*.

Sohl-Dickstein, J.; Battaglino, P. B.; and DeWeese, M. R. 2011. New method for parameter estimation in probabilistic models: minimum probability flow. *Physical review letters* 107(22):220601.

Team, T. T. D.; Al-Rfou, R.; Alain, G.; Almahairi, A.; Angermueller, C.; Bahdanau, D.; Ballas, N.; Bastien, F.; Bayer, J.; Belikov, A.; et al. 2016. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

Tubiana, J., and Monasson, R. 2017. Emergence of compositional representations in restricted boltzmann machines. *Physical Review Letters* 118(13):138301.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xie, X., and Seung, H. S. 2000. Spike-based learning rules and stabilization of persistent neural activity. In *Advances in neural information processing systems*, 199–208.