Techniques for Learning Binary Stochastic Feedforward Neural Networks

Tapani Raiko Aalto University
Mathias Berglund Aalto University
Guillaume Alain Université de Montréal
Laurent Dinh Université de Montréal

Abstract

Stochastic binary hidden units in a multi-layer perceptron (MLP) network give at least three potential benefits when compared to deterministic MLP networks. (1) They allow to learn one-to-many type of mappings. (2) They can be used in structured prediction problems, where modeling the internal structure of the output is important. (3) Stochasticity has been shown to be an excellent regularizer, which makes generalization performance potentially better in general. However, training stochastic networks is considerably more difficult. We study training using M samples of hidden activations per input. We show that the case $M = 1$ leads to a fundamentally different behavior where the network tries to avoid stochasticity. We propose two new estimators for the training gradient and propose benchmark tests for comparing training algorithms. Our experiments confirm that training stochastic networks is difficult and show that the proposed two estimators perform favorably among all the five known estimators.

1 Introduction

Feedforward neural networks, or multi-layer perceptron (MLP) networks, model mappings from inputs $\mathbf{x}$ to outputs $\mathbf{y}$ through hidden units $\mathbf{h}$. Typically the network output defines a simple (unimodal) distribution such as an isotropic Gaussian or a fully factorial Bernoulli distribution. In case the hidden units are deterministic (using a function $\mathbf{x} \rightarrow \mathbf{h}$ as opposed to a distribution $P(\mathbf{h}|\mathbf{x})$), the conditionals $P(\mathbf{y}|\mathbf{x})$ belong to the same family of simple distributions.

Stochastic feedforward neural networks (SFNN) (Neal, 1990, 1992) have the advantage when the conditionals $P(\mathbf{y}|\mathbf{x})$ are more complicated. While each configuration of hidden units $\mathbf{h}$ produces a simple output, the mixture over them can approximate any distribution, including multimodal distributions required for one-to-many type of mappings. In the extreme case of using empty vectors as the input $\mathbf{x}$, they can be used for unsupervised learning of the outputs $\mathbf{y}$.

Another potential advantage of stochastic networks is in generalization performance. Adding noise or stochasticity to the inputs of a deterministic neural network has been found useful as a regularization method (Sietsma and Dow, 1991). Introducing multiplicative binary noise to the hidden units (dropout, Hinton *et al.*, 2012) regularizes even better.

The early work on SFNNs approached the inference of $\mathbf{h}$ using Gibbs sampling (Neal, 1990, 1992) or mean field (Saul *et al.*, 1996), which both have their downsides. Gibbs sampling can mix poorly and the mean-field approximation optimizes a lower bound on the likelihood that may be too loose. More recent work proposes simply drawing samples from $P(\mathbf{h}|\mathbf{x})$ during the feedforward phase (Hinton *et al.*, 2012; Bengio, 2013; Tang and Salakhutdinov, 2013). This guarantees independent samples and an unbiased estimate of $P(\mathbf{y}|\mathbf{x})$.

We can use standard back-propagation when using stochastic continuous-valued units (e.g. with additive noise or dropout), but back-propagation is no longer possible with discrete units. There are several ways of estimating the gradient in that case. Bengio (2013) proposes two such estimators: an unbiased estimator with a large variance, and a biased version that approximates back-propagation.

Tang and Salakhutdinov (2013) propose an unbiased estimator of a lower bound that works reasonably well in a hybrid network containing both deterministic and stochastic units. Their approach relies on using more than one sample from $P(\mathbf{y}|\mathbf{x})$ for each training example, and in this paper we provide theory to show that using more than one sample is an important requirement. They also demonstrate interesting applications such as mapping the face of a person into varying expressions, or mapping a silhouette of an object into a color image of the object.

Tang and Salakhutdinov (2013) argue for the choice of a hybrid network structure based on the finite (and thus limited) number of hidden configurations in a fully discrete $\mathbf{h}$. However, we offer an alternate hypothesis: It is much easier to learn a deterministic network around a small number of stochastic units, so that it might not even be important to train the stochastic units properly. In an extreme case, the stochastic units are not trained at all, and the deterministic units do all the work.

In this work, we take a step back and study more rigorously the training problem with fully stochastic networks. We compare different methods of estimating the gradient and propose two new estimators. One is an approximate back-propagation with less bias than the one by Bengio (2013), and the other is a modification of the estimator by Tang and Salakhutdinov (2013) with less variance. We propose a benchmark test setting based on the well-known MNIST data.

2 Stochastic Feedforward Neural Networks

We study a model that maps inputs $\mathbf{x}$ to outputs $\mathbf{y}$ through stochastic binary hidden units $\mathbf{h}$. The equations are given for just one hidden layer, but the extension to multiple layers is easy[1]. The activation probability is computed just like the activation function in deterministic multi-layer perceptron (MLP) networks:

$$P(h_i = 1 \mid \mathbf{x}) = \sigma(a_i) = \sigma(\mathbf{W}_{i:}\mathbf{x} + b_i), \tag{1}$$

where $\mathbf{W}_{i:}$ denotes the ith row vector of matrix $\mathbf{W}$ and $\sigma(\cdot)$ is the sigmoid function. For classification problems, we use softmax for the output probability

$$P(y = i \mid \mathbf{h}) = \frac{\exp(\mathbf{V}_{i:}\mathbf{h} + c_i)}{\sum_j \exp(\mathbf{V}_{j:}\mathbf{h} + c_j)}. \tag{2}$$

For predicting binary vectors $\mathbf{y}$, we use a product of Bernoulli distributions

$$P(y_i = 1 \mid \mathbf{h}) = \sigma(\mathbf{V}_{i:}\mathbf{h} + c_i). \tag{3}$$

The probabilistic training criterion for deterministic MLP networks is $\log P(\mathbf{y}|\mathbf{x})$. Its gradient with respect to model parameters $\boldsymbol{\theta} = \{\mathbf{W}, \mathbf{V}, \mathbf{b}, \mathbf{c}\}$ can be computed using the back-propagation algorithm, which is based on the chain rule of derivatives. Stochasticity brings difficulties in both estimating the training criterion and in estimating the gradient. The training criterion of the stochastic network

$$\mathcal{C} = \log P(\mathbf{y} \mid \mathbf{x}) = \log \sum_{\mathbf{h}} P(\mathbf{y}, \mathbf{h} \mid \mathbf{x}) = \log \sum_{\mathbf{h}} P(\mathbf{y} \mid \mathbf{h}) P(\mathbf{h} \mid \mathbf{x}) \tag{4}$$

$$= \log \mathbb{E}_{P(\mathbf{h}|\mathbf{x})} P(\mathbf{y} \mid \mathbf{h}) \tag{5}$$

requires summation over an exponential number of configurations of $\mathbf{h}$. Also, derivatives with respect to discrete variables cannot be directly defined. We will review and propose solutions to both problems below.

2.1 Proposed Estimator of the Training Criterion

We propose to estimate the training criterion in Equation (4) by

$$\hat{\mathcal{C}}_M = \log \frac{1}{M} \sum_{m=1}^{M} P(\mathbf{y} \mid \mathbf{h}^{(m)}) \tag{6}$$

$$\mathbf{h}^{(m)} \sim P(\mathbf{h} \mid \mathbf{x}). \tag{7}$$

[1] Apply $P(\mathbf{h}|\mathbf{x})$ separately for each layer such that $\mathbf{x}$ denotes the layer below instead of the original input.

This can be interpreted as the performance of a finite mixture model over M samples drawn from $P(\mathbf{h}|\mathbf{x})$.

One could hope that using just $M = 1$ sample just like in many other stochastic networks (e.g. Hinton *et al.*, 2012) would work well enough. However, here we show in that case the network always prefers to minimize the stochasticity, for instance by increasing the input weights to a stochastic sigmoid unit such that it behaves as a deterministic step-function nonlinearity.

Theorem 1. *When maximizing the expectation of $\hat{\mathcal{C}}_1$ in Equation (6) using $M = 1$, a hidden unit h_i never prefers a stochastic output over a deterministic one. However, when maximizing the expectation of $\mathcal{C}$ in Equation (4), the hidden unit h_i may prefer a stochastic output over any of the deterministic ones.*

Proof. The expected $\hat{\mathcal{C}}_1$ over the data distribution can be upper-bounded as

$$\begin{aligned}
\mathbb{E}_{P_d(\mathbf{x},\mathbf{y})}\mathbb{E}_{P(\mathbf{h}|\mathbf{x})}\left[\hat{\mathcal{C}}_1\right] &= \mathbb{E}_{P_d(\mathbf{x},\mathbf{y})}\mathbb{E}_{P(\mathbf{h}|\mathbf{x})}\log P(\mathbf{y} \mid \mathbf{h}) && (8)\\
&= \mathbb{E}_{P_d(\mathbf{x})}\mathbb{E}_{P(h_i|\mathbf{x})}\left[\mathbb{E}_{P_d(\mathbf{y}|\mathbf{x})}\mathbb{E}_{P(\mathbf{h}_{\setminus i}|\mathbf{x},h_i)}\log P(\mathbf{y} \mid \mathbf{h})\right] && (9)\\
&= \mathbb{E}_{P_d(\mathbf{x})}\mathbb{E}_{P(h_i|\mathbf{x})}f(h_i,\mathbf{x}) && (10)\\
&\leq \mathbb{E}_{P_d(\mathbf{x})}\max_{h_i} f(h_i,\mathbf{x}), && (11)
\end{aligned}$$

where P_d denotes the data distribution. The value in the last inequality is achievable by selecting the distribution of $P(h_i|\mathbf{x})$ to be a Dirac delta around the value h_i which maximizes the deterministic function $f(h_i, x)$. This can be done for every x under the expectation in an independent way. This is analogous to a idea from game theory: since the performance achieved with $P(h_i|\mathbf{x})$ is a linear combination of the performances $f(h_i, \mathbf{x})$ of the deterministic choices h_i, any mixed strategy $P(h_i|\mathbf{x})$ cannot be better than the best deterministic choice.

Let us now look at the situation for the expectation $\mathcal{C}$ and see how it differs from the case of $\mathcal{C}_1$ that we had with one particle. We can see that the original training criterion can be written as the expectation of a KL-divergence.

$$\begin{aligned}
\mathbb{E}_{P_d(\mathbf{x},\mathbf{y})}\left[\mathcal{C}\right] &= \mathbb{E}_{P_d(\mathbf{x})}\mathbb{E}_{P_d(\mathbf{y}|\mathbf{x})}\log P(\mathbf{y} \mid \mathbf{x}) && (12)\\
&= \mathbb{E}_{P_d(\mathbf{x})}\left[\mathrm{KL}\left(P_d(\mathbf{y} \mid \mathbf{x}) \| P(\mathbf{y} \mid \mathbf{x})\right) + \text{const}\right] && (13)
\end{aligned}$$

The fact that this expression features a KL-divergence means that the minimum is achieved when the conditionals match exactly. That is, it it minimized when we have that $P(\mathbf{y}|\mathbf{x}) = P_d(\mathbf{y}|\mathbf{x})$ for each value of $\mathbf{x}$.

We give a simple example in which (x, h, y) each take values in $\{0, 1\}$. We define the following conditions on $P_d(y|x)$ and $P(y|h)$, and we show how any deterministic $P(h|x)$ is doing a bad job at maximizing (12).

$$P_d(y \mid x) = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}, \quad P(y \mid h) = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix} \qquad (14)$$

A deterministic $P(h \mid x) = \begin{bmatrix} a & b \\ 1-a & 1-b \end{bmatrix}$ is one in which a, b take values in $\{0, 1\}$.

The optimal solution is $(a, b) = (0.5, 0.5)$, regardless of the distribution $P_d(x)$. For the purposes of comparing solutions, we can simply take $P_d(x) = [0.5\ 0.5]$. In that case, we get that the expected $\mathcal{C}$ takes the value $0.5\log(0.5) + 0.5\log(0.5) \approx -0.30$. On the other hand, all the deterministic solutions yield a lower value $0.5\log(0.9) + 0.5\log(0.1) \approx -0.52$.

$\square$

2.2 Gradient for Training $P(\mathbf{y}|\mathbf{h})$

We will be exploring five different estimators for the gradient of a training criterion wrt. parameters $\boldsymbol{\theta}$. However, all of them will share the following gradient for training $P(\mathbf{y}|\mathbf{h})$.

For training $P(\mathbf{y}|\mathbf{h})$, we compute the gradient of the training criterion $\hat{\mathcal{C}}_M$ in Equation (6)

$$\hat{\mathcal{C}}_M = \log \frac{1}{M} \sum_{m=1}^{M} P(\mathbf{y} \mid \mathbf{h}^{(m)}) = \log \frac{1}{M} \sum_{m=1}^{M} \phi(\mathbf{o}^{(m)}) \tag{15}$$

$$G(\mathbf{o}^{(m)}) := \frac{\partial \hat{\mathcal{C}}_M}{\partial \mathbf{o}^{(m)}} = \frac{\phi'(\mathbf{o}^{(m)})}{\sum_{m'=1}^{M} \phi(\mathbf{o}^{(m')})} = \frac{\phi(\mathbf{o}^{(m)})}{\sum_{m'=1}^{M} \phi(\mathbf{o}^{(m')})} \frac{\partial \log \phi(\mathbf{o}^{(m)})}{\partial \mathbf{o}^{(m)}} \tag{16}$$

$$= \frac{P(\mathbf{y} \mid \mathbf{h}^{(m)})}{\sum_{m'=1}^{M} P(\mathbf{y} \mid \mathbf{h}^{(m')})} \frac{\partial \log \phi(\mathbf{o}^{(m)})}{\partial \mathbf{o}^{(m)}} = \frac{w^{(m)}}{\sum_{m'=1}^{M} w^{(m')}} \frac{\partial \log \phi(\mathbf{o}^{(m)})}{\partial \mathbf{o}^{(m)}}, \tag{17}$$

where $\mathbf{o} = \mathbf{V}\mathbf{h} + \mathbf{c}$ is the incoming signal to the activation function $\phi(\cdot)$ in the output, and $w^{(m)} = P(\mathbf{y}|\mathbf{h}^{(m)})$ are unnormalized weights. In other words, we get the gradient in the mixture by computing the gradient of the individual contribution m and multiplying it with normalized weights $\bar{w}^{(m)} = w^{(m)} / \sum_{m'=1}^{M} w^{(m')}$. The normalized weights $\bar{w}^{(m)}$ can be interpreted as responsibilities in a mixture model (see e.g. Bishop *et al.*, 2006, Section 2.3.9).

The gradients $G(\mathbf{V})$, $G(\mathbf{h})$, and $G(\mathbf{c})$ are computed from $G(\mathbf{o})$ using the chain rule of derivatives just like in standard back-propagation.

2.3 First Estimators of the Gradient for Training $P(\mathbf{h}|\mathbf{x})$

Bengio (2013) proposed two estimators of the gradient for $P(\mathbf{h}|\mathbf{x})$. The first one is unbiased but has high variance. It is defined as

$$G_1(a_i) := (h_i - \sigma(a_i))(L - \bar{L}_i) \tag{18}$$

$$\bar{L}_i = \frac{\mathbb{E}\left[(h_i - \sigma(a_i))^2 L\right]}{\mathbb{E}\left[(h_i - \sigma(a_i))^2\right]}, \tag{19}$$

where we plug in $L = \hat{\mathcal{C}}_M$ as the training criterion. We estimate the numerator and the denominator of $\bar{L}_i$ with an exponential moving average.

The second estimator is biased but has lower variance. It is based on back-propagation where we set $\frac{\partial h_i}{\partial a_i} := 1$ resulting in

$$G_2(a_i) := G(h_i) = (\mathbf{V}_{:i})^T G(\mathbf{o}). \tag{20}$$

2.4 Proposed Biased Estimator of the Gradient for Training $P(\mathbf{h}|\mathbf{x})$

We propose a new way of propagating the gradient of the training criterion through discrete hidden units h_i. Let us consider h_i continuous random variables with additive noise ϵ_i

$$h_i = \sigma(a_i) + \epsilon_i \tag{21}$$

$$\epsilon_i \sim \begin{cases} 1 - \sigma(a_i) & \text{with probability} \quad \sigma(a_i) \\ -\sigma(a_i) & \text{with probability} \quad 1 - \sigma(a_i) \end{cases} \tag{22}$$

Note that h_i has the same distribution as in Equation (1), that is, it only gets values 0 and 1. With this formulation, we propose to back-propagate derivatives through h_i by

$$G_3(a_i) := \sigma'(a_i) G(h_i) = \sigma'(a_i)(\mathbf{V}_{:i})^T G(\mathbf{o}) \tag{23}$$

This gives us a biased estimate of the gradient since we ignore the fact that the structure of the noise ϵ_i depends on the input signal a_i. One should note, however, that the noise is zero-mean with any input a_i, which should help keep the bias relatively small.

2.5 Variational Training

Tang and Salakhutdinov (2013) use a variational lower bound $\mathcal{L}(Q)$ on the training criterion $\mathcal{C}$ as

$$\mathcal{C} = \log P(\mathbf{y} \mid \mathbf{x}) = \sum_{\mathbf{h}} P(\mathbf{h} \mid \mathbf{y}, \mathbf{x}) \log \frac{P(\mathbf{y}, \mathbf{h} \mid \mathbf{x})}{P(\mathbf{h} \mid \mathbf{y}, \mathbf{x})} \geq \sum_{\mathbf{h}} Q(\mathbf{h}) \log \frac{P(\mathbf{y}, \mathbf{h} \mid \mathbf{x})}{Q(\mathbf{h})} =: \mathcal{L}(Q). \tag{24}$$

The above inequality holds for any distribution $Q(\mathbf{h})$, but we get more usefulness out of it by choosing $Q(\mathbf{h})$ so that it serves as a good approximation of $P(\mathbf{h} \mid \mathbf{y}, \mathbf{x})$.

We start by noting that we can use importance sampling to express $P(\mathbf{h} \mid \mathbf{y}, \mathbf{x})$ in terms of a proposal distribution $R(\mathbf{h}|\mathbf{y}, \mathbf{x})$ from which we can draw samples.

$$P(\mathbf{h} \mid \mathbf{y}, \mathbf{x}) \propto P(\mathbf{y} \mid \mathbf{h}) P(\mathbf{h} \mid \mathbf{x}) = \frac{P(\mathbf{y} \mid \mathbf{h}) P(\mathbf{h} \mid \mathbf{x})}{R(\mathbf{h} \mid \mathbf{y}, \mathbf{x})} R(\mathbf{h} \mid \mathbf{y}, \mathbf{x}) \tag{25}$$

We construct $Q(\mathbf{h})$ based on this expansion:

$$Q(\mathbf{h}) = \sum_{m=1}^{M} \bar{w}^{(m)} \delta(\mathbf{h}^{(m)}) \tag{26}$$

$$\mathbf{h}^{(m)} \sim R(\mathbf{h} \mid \mathbf{y}, \mathbf{x}) \tag{27}$$

$$w^{(m)} = \frac{P(\mathbf{y} \mid \mathbf{h}^{(m)}) P(\mathbf{h}^{(m)} \mid \mathbf{x})}{R(\mathbf{h}^{(m)} \mid \mathbf{y}, \mathbf{x})} \tag{28}$$

$$\bar{w}^{(m)} = \frac{w^{(m)}}{\sum_{m'=1}^{M} w^{(m')}} \tag{29}$$

where $\delta(\mathbf{h}^{(m)})$ is the Dirac delta function centered at $\mathbf{h}^{(m)}$, and $w^{(m)}$ and $\bar{w}^{(m)}$ are called the unnormalized and normalized important weights.

It would be an interesting line of research to train an auxiliary model for the proposal distribution $R(\mathbf{h}|\mathbf{x}, \mathbf{y})$ following ideas from (Kingma and Welling, 2013; Rezende *et al.*, 2014; Mnih and Gregor, 2014) that call the equivalent of R the recognition model or the inference network. However, we do not pursuit that line further in this paper and follow Tang and Salakhutdinov (2013) who chose $R(\mathbf{h}|\mathbf{x}, \mathbf{y}) := P(\mathbf{h}|\mathbf{x})$, in which case the importance weights simplify to $w^{(m)} = P(\mathbf{y}|\mathbf{h}^{(m)})$.

Tang and Salakhutdinov (2013) use a generalized EM algorithm, where they compute the gradient for the lower bound $\mathcal{L}(Q)$ given that $Q(\mathbf{h})$ is fixed

$$G_4(\boldsymbol{\theta}) := \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{\mathbf{h}} Q(\mathbf{h}) \log \frac{P(\mathbf{y}, \mathbf{h} \mid \mathbf{x})}{Q(\mathbf{h})} \tag{30}$$

$$= \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{m=1}^{M} \bar{w}^{(m)} \left[\log P(\mathbf{y} \mid \mathbf{h}^{(m)}) + \log P(\mathbf{h}^{(m)} \mid \mathbf{x}) \right]. \tag{31}$$

Thus, we train $P(\mathbf{h}^{(m)}|\mathbf{x})$ using $\mathbf{h}^{(m)}$ as target outputs.

It turns out that the resulting gradient for $P(\mathbf{y}|\mathbf{h})$ is exactly the same as in Section 2.2, despite the rather different way of obtaining it. The importance weights $\bar{w}^{(m)}$ have the same role as responsibilities $\bar{w}^{(m)}$ in the mixture model, so we can use the same notation for them.

Proposed Unbiased Estimator of the Gradient We propose a new gradient estimator by applying a variance reduction technique (Weaver and Tao, 2001; Mnih and Gregor, 2014) to the estimator by Tang and Salakhutdinov (2013). First we note that

$$\mathbb{E}_{P(\mathbf{h}|\mathbf{x})} \left[\frac{\partial}{\partial \boldsymbol{\theta}} \log P(\mathbf{h} \mid \mathbf{x}) \right] = \int P(\mathbf{h} \mid \mathbf{x}) \frac{\frac{\partial}{\partial \boldsymbol{\theta}} P(\mathbf{h} \mid \mathbf{x})}{P(\mathbf{h} \mid \mathbf{x})} \mathrm{d}\mathbf{h} = \frac{\partial}{\partial \boldsymbol{\theta}} \int P(\mathbf{h} \mid \mathbf{x}) \mathrm{d}\mathbf{h} = \frac{\partial}{\partial \theta} 1 = 0. \tag{32}$$

That is, when training $P(\mathbf{h}|\mathbf{x})$ with samples $\mathbf{h}^{(m)} \sim P(\mathbf{h}|\mathbf{x})$ drawn from the model distribution, the gradient is on average zero. Therefore we can change the estimator of $P(\mathbf{h}|\mathbf{x})$ by subtracting any constant c from the weights $\bar{w}^{(m)}$ without introducing any bias. We choose $c = \mathbb{E}\left[\bar{w}^{(m)}\right] = \frac{1}{M}$ which is empirically show to be sufficiently close to the optimum (see Figure 1). Finally, the proposed estimator becomes

$$G_5(\boldsymbol{\theta}) := \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{m=1}^{M} \left[\bar{w}^{(m)} \log P(\mathbf{y} \mid \mathbf{h}^{(m)}) + \left(\bar{w}^{(m)} - \frac{1}{M} \right) \log P(\mathbf{h}^{(m)} \mid \mathbf{x}) \right]. \tag{33}$$

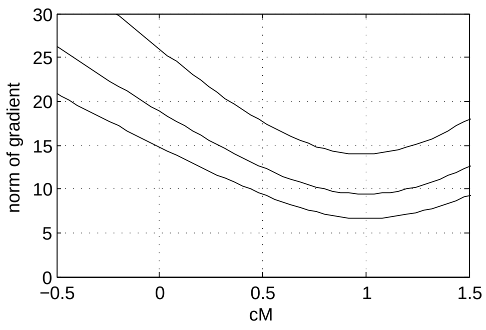

Figure 1: The norm of the gradient for the weights of the first hidden layer as a function of cM where the proposed $c = \frac{1}{M}$ in Equation (33) corresponds to $cM = 1$. The norm is averaged over a mini-batch, after {1,7,50} epochs of training (curves from top to bottom) with G_5 in the experiment described in Section 3.1. Varying c only changes the variance of the estimator, so the minimum norm corresponds to the minimum variance.

Test error (%)	$M = 1$	$M = 20$
G_1 (Bengio, 2013, unbiased)	7.85	11.30
G_2 (Bengio, 2013, biased)	7.97	7.86
G_3 (proposed biased)	1.82	1.63
G_4 (Tang and Salakhutdinov, 2013)	na	3.99
G_5 (proposed unbiased)	na	2.72
deterministic	1.51	na
deterministic tested as stochastic	1.54	na

Table 1: Results obtained on MNIST classification using various number of samples M during training and various estimators of the gradient G_i.

3 Experiments

We propose two simple tests as benchmarks for stochastic feedforward networks based on MNIST handwritten digit dataset (LeCun *et al.*, 1998b). The first one is classification, and the latter one is the prediction of the lower half of the image based on the upper half.

As a comparison method, we also trained deterministic networks (corresponding to G_3 with $\epsilon_i = 0$ in Equation (22)). We also consider training a network as deterministic and testing it as stochastic.

In all of the experiments, we used stochastic gradient with a mini-batch size of 100 and momentum of 0.9. We used a learning rate schedule where the learning rate increases linearly from zero to maximum during the first five epochs and back to zero during the remaining epochs. The maximum learning rate was chosen among $\{0.0001, 0.0003, 0.001, \ldots, 0.3\}$ and the best test error for each method is reported.[2] The models were trained with $M \in \{1, 20\}$, and during test time we always used $M = 100$.

3.1 Classification

MNIST classification is a well studied problem where performances of a huge variety of approaches are known. Since the output $\mathbf{y}$ is just a class label, the advantage of being able to model complex output distributions is not applicable. Still, the benchmark is useful for comparing training algorithms against each other.

We used a network structure with dimensionalities 784-200-200-10. The input data was first scaled to the range of $[0, 1]$, and the mean of each pixel was then subtracted. As a regularization method, Gaussian noise with standard deviation 0.4 was added to each pixel separately in each epoch (Raiko *et al.*, 2012). The models were trained for 50 epochs.

[2] We considered that using a separate validation set would not be necessary for choosing just one hyperparameter.

Test log-likelihood ($\hat{\mathcal{C}}_{100}$)	$M = 1$	$M = 20$
G_3 (proposed biased)	-60.6	-53.8
G_4 (Tang and Salakhutdinov, 2013)	na	-63.7
G_5 (proposed unbiased)	na	-63.2
deterministic	-68.3	na
deterministic tested as stochastic	-64.5	na

Table 2: Results obtained on MNIST structured prediction using various number of samples M during training and various estimators of the gradient G_i. We can observe that all stochastic networks outperform the deterministic network.

Table 1 gives the test set error rate for each method. As can be seen from the table, deterministic networks give the best results, followed by the proposed biased gradient G_3 and the proposed unbiased gradient G_5.

3.2 Structured Prediction

Predicting the lower half of an image based on the upper half is a structured prediction problem, where the output distribution is likely to be complex and multimodal. The task is hence appropriate for testing stochastic networks. The MNIST dataset used in the experiments was binarized as a preprocessing step by sampling each pixel independently using the grey-scale value as its expectation. We compared the performance of the previously best-performing estimators G_3–G_5, and used a network structure with dimensionalities 392-200-200-392 trained for 100 epochs.

As can be seen in Table 2, the proposed biased estimator G_3 performs the best, while the deterministic network performs the worst. It is notable that the performance of G_3 increased significantly when using more than $M = 1$ particles, as could be predicted from Theorem 1. In Figure 2 we plot the objective $\mathcal{C}_M$ at test time based on a number of particles $M = 1, \ldots, 100$. In theory, a larger number of particles M is always better (if given infinite computational resources), but here Figure 2 shows how the objective $\mathcal{C}_M$ is estimated very accurately with only $M = 20$ or $M = 40$.

These results leave lot of room for improvement: Firstly, the best results are produced by a biased estimator, so it might be beneficial to fine-tune those results with an unbiased one. Secondly, the networks could be larger and training could be continued longer if given enough computational capacity. However, the critical difference between the stochastic networks and the deterministic network can be be observed in Figure 3, where the stochastic networks are able to generate reconstructions that correspond to different digits for an ambiguous input. Clearly, the deterministic network cannot model such a distribution.

4 Discussion

It is well known (LeCun *et al.*, 1998a) that networks trained with sigmoid activation function converge slower than activation functions centered around zero, and presumably, the same applies to binary $\{0, 1\}$ hidden units. The training could benefit from using either $\{-1, 1\}$ activations, or from transformations that keep the activations centered around zero mean (Raiko *et al.*, 2012).

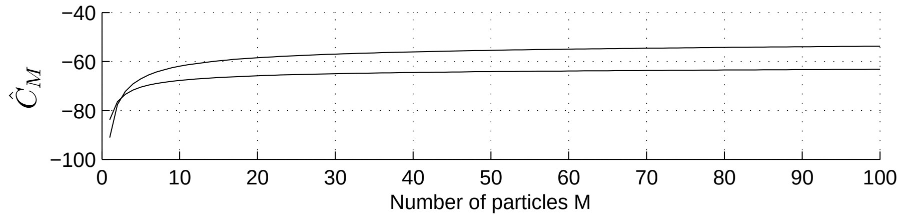

Figure 2: $\hat{\mathcal{C}}_M$ as a function of the number of particles used during test time for the structured prediction task for the two proposed models trained with $M = 20$.

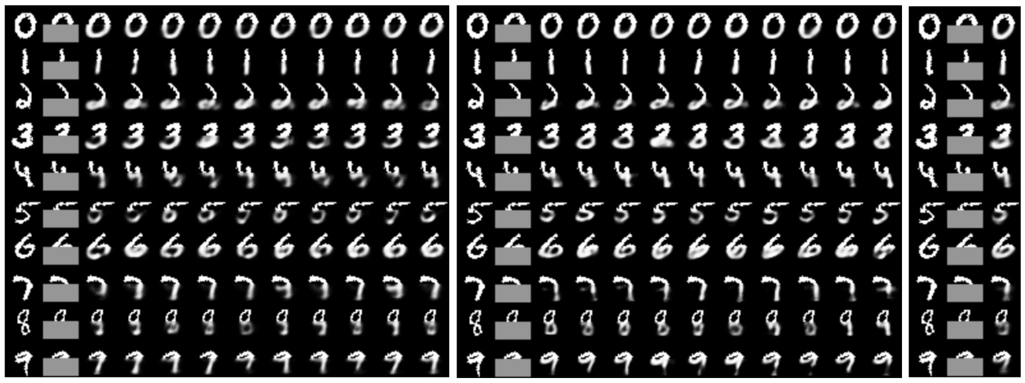
Figure 3: Samples drawn from the prediction of the lower half of the MNIST test data digits based on the upper half with models trained using G_3 (left), G_5 (middle), and for the deterministic network (right). The leftmost column is the original MNIST digit, followed by the masked out image and ten samples. The figures illustrate how the stochastic networks are able to model different digits in the case of ambiguous inputs.

In the proposed estimator of the gradient for $P(\mathbf{h}|\mathbf{x})$ in Equation (33), there are both positive and negative weights for various particles $\mathbf{h}^{(m)}$. Positive weights can be interpreted as pulling probability mass towards the particle, and negative weights as pushing probability mass away from the particle. The experiments showed that having both positive and negative weights worked better than using positive weights only.

One challenge with structured outputs $\mathbf{y}$ is to find samples $\mathbf{h}^{(m)}$ that give a reasonably large probability $P(\mathbf{y}|\mathbf{h}^{(m)})$ with a reasonably small sample size M. Training a separate $R(\mathbf{h}|\mathbf{x},\mathbf{y}) \neq P(\mathbf{h}|\mathbf{x})$ as a proposal distribution looks like a promising direction for addressing that issue. It might still be useful to use a mix of particles from R and $P(\mathbf{h}|\mathbf{x})$, and subtract a constant from the weights of the latter ones. This approach would yield both particles that explain $\mathbf{y}$ well, and particles that have negative weights.

5 Conclusion

Using stochastic neurons in a feedforward network is more than just a computational trick to train deterministic models. The model itself can be defined in terms of stochastic particles in the hidden layers, and we have shown in this paper how there are many valid alternatives to the usual formulation for the gradient.

These proposals for the gradient involve normalized weights associated to the particles in the hidden layers, and those weights represent how well the particles explain the output targets. We showed both theoretically and experimentally, how involving more than one particle enhances the modeling capacity in a significant way.

We demonstrated the validity of these techniques by training a classifier on MNIST that achieved a reasonable performance, and by training another network that could fill in the missing information when we deleted the bottom part of the MNIST digits.

We hope that we have provided some insight into the properties of stochastic feedforward neural networks, and that the theory can be applied to other contexts such as the study of Dropout or other important techniques that give a stochastic flavor to deterministic models.

References

Bengio, Y. (2013). Estimating or propagating gradients through stochastic neurons. *arXiv preprint arXiv:1305.2982*.

Bishop, C. M. *et al.* (2006). *Pattern recognition and machine learning*, volume 1. Springer New York.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.

LeCun, Y., Bottou, L., Orr, G., and Müller, K. (1998a). Efficient backprop. *Neural Networks: Tricks of the Trade*, pages 546–546.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998b). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324.

Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.

Neal, R. M. (1990). Learning stochastic feedforward networks. *Department of Computer Science, University of Toronto*.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial intelligence*, **56**(1), 71–113.

Raiko, T., Valpola, H., and LeCun, Y. (2012). Deep learning made easier by linear transformations in perceptrons. In *International Conference on Artificial Intelligence and Statistics*, pages 924–932.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic back-propagation and variational inference in deep latent Gaussian models. *arXiv preprint arXiv:1401.4082*.

Saul, L. K., Jaakkola, T., and Jordan, M. I. (1996). Mean field theory for sigmoid belief networks. *arXiv preprint cs/9603102*.

Sietsma, J. and Dow, R. J. (1991). Creating artificial neural networks that generalize. *Neural Networks*, **4**(1), 67–79.

Tang, Y. and Salakhutdinov, R. (2013). Learning stochastic feedforward neural networks. In *Advances in Neural Information Processing Systems*, pages 530–538.

Weaver, L. and Tao, N. (2001). The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 538–545. Morgan Kaufmann Publishers Inc.