On the Duality between Belief Networks and Feed-Forward Neural Networks

Dr. Paul M. Baggenstoss
Fraunhofer FKIE
Wachtberg, Germany
+49-228-9435-150
p.m.baggenstoss@ieee.org
http://class-specific.com/csf/index.html

***Abstract*—This paper addresses the duality between deterministic feed-forward neural networks (FF-NNs), and linear Bayesian networks (LBNs), which are generative stochastic models representing probability distributions over the visible data based on a linear function of a set of latent (hidden) variables. The maximum entropy principle is used to define a unique generative model corresponding to each FF-NN, called projected belief network (PBN). The FF-NN exactly recovers the hidden variables of the dual PBN. The large-N asymptotic approximation to the PBN has the familiar structure of an LBN, with the addition of an invertible non-linear transformation operating on the latent variables. It is shown that the exact nature of the PBN depends on the range of the input (visible) data - details for three cases of input data range are provided. The likelihood function of the PBN is straight-forward to calculate, allowing it to be used as a generative classifier. An example is provided in which a generative classifier based on the PBN has comparable performance to a deep belief network in classifying handwritten characters. In addition, several examples are provided that demonstrate the duality relationship, for example by training networks from either side of the duality.**

I. Introduction

A. Motivation and Background

Discriminative and generative networks are two fundamental but opposing types of networks in common use today. The classical discriminative network is the feed-forward neural network (FF-NN), also known as multi-layer perceptron (MLP). This is a deterministic "inference" process that maps the input (visible) data to a set of (usually lower-dimension) intermediate variables. The corresponding generative network is a Bayesian belief network with directed linear connections, called linear Bayesian networks (LBN). The LBN is a generative stochastic models that models the probability distribution of the visible data based on a set of latent (hidden) variables.

The relationship between the two types of networks has been exploited for some time. For example, the parameters of the generative form can be tuned by training an inference network using back-propagation [1]. Or, the hidden variables of the generative model can be approximated by an inference network [2]. The structure of the FF-NN and LBN appear to be duals. This paper examines the exact nature of this duality.

This duality relationship can be better understood by examining a restricted Boltzmann machine (RBM), which is comprised of a set of opposing single-layer LBNs sharing a common set of weights [3]. Once the RBM is trained, and the weights of one LBN are used as an FF-NN, they approximate the hidden variables of the opposing LBN. Thus, the FF-NN created from one LBN is approximately the dual network of the opposing LBN. But, this is an approximate duality forced by special way that the RBM is trained. It would be interesting to know if the dual network can be derived without needing to approximate it through specialized training. Understanding this duality could be helpful in cross-training (obtain the parameters of one by training the other), applying the dual networks without needing to explicitly train them, or developing new types of classifiers. In what follows, the duality relationship is formalized to find the dual generative counterpart for any FF-NN. The necessary tools for this task are found in a method called probability density function (PDF) projection, and in the principle of maximum entropy, which together are called maximum entropy PDF projection (ME-PP) [4]. The dual network of the FF-NN turns out to be an LBN with weight matrix equal to the transpose of the FF-NN weight matrix, but with an additional non-linear operator preceeding the linear operation. This non-linear operator depends on the input data range.

B. Main Contributions and Paper Outline

The main contributions of this paper are (a) the use of ME-PP to construct a PBN which is the dual LBN corresponding to a given FF-NN, (b) the decription of the large-N asymptotic form of the PBN, which has the structure of an LBN together with a special invertible function operating on the latent variables, (c) the use of PBN to construct an auto-encoder with tied analysis and reconstruction weights, (d) the use of PBN to construct a generative classifier for handwritten character recognition, and (e) the direct generative training of a PBN by maximization of the likelihood function. Other constributions include various experiments demonstrating the duality relationship between PBN and FF-NN.

Section II is devoted to laying the mathematical foundation needed for PBN including review of FF-NNs and LBNs (Sections II-A and II-B), PDF projection (Section II-C), maximum entropy PDF projection (Section II-D), and uniform manifold

sampling (Section II-E). The PBn is described in Section III which covers the general form of the PBN (Section III-A), the specifics for unbounded data (Section III-B), positive-valued data (Section III-C), and for data in $[0,\ 1]$ (Section III-D). Various arrangements of the PBN are presented in Sections III-F through III-H and potential applications of the PBN are described in Section III-I. Experimental results are divided into illustrative experiments (Section IV) and classification experiments (Section V).

II. Mathematical Background

A. Feed-Forward Neural Network (FF-NN)

Figure 1 shows one layer of a feed-forward neural network (FF-NN), which is a purely deterministic mapping, transforming vector $\mathbf{x} \in \mathcal{R}^N$ to $\mathbf{f} \in \mathcal{R}^M$, where it is assumed that $M < N$. This paper is concerned only with network layers where $M < N$. The intermediate feature vector $\mathbf{z}$ is generated from $\mathbf{x}$ using the linear transformation

$$\mathbf{z} = \mathbf{A}'\mathbf{x}. \tag{1}$$

The components f_i of the output variable $\mathbf{f}$ are then generated by adding the bias vector $\mathbf{c}$ then applying a non-linear "activation" function $f(z)$, which forces f_i in the range $[0, 1]$. In matrix form, $\mathbf{f} = f(\mathbf{z} + \mathbf{c})$. Examples of activation functions are the sigmoid function

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{2}$$

or the "TED" non-linearity

$$f(x) = \frac{e^x}{e^x - 1} - \frac{1}{x}. \tag{3}$$

which is derived from the truncated exponential distribution (TED), as shown in Section II-B. On the bottom of the figure,

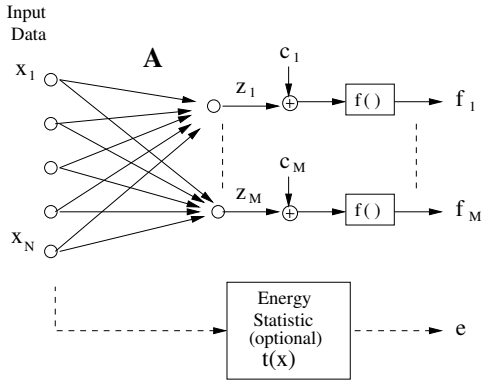

Fig. 1. One layer of a feed-forward network of perceptrons.

the optional energy statistic $e = t(\mathbf{x})$ is shown - it will be explained in Section II-D.

B. Linear Bayesian Networks (LBN)

Figure 2 shows one layer of a linear Bayesian network (LBN). The figure shows how the "visible" data $\mathbf{x} = [x_1, x_2 \ldots x_N]$ is generated conditioned on the "hidden" random variables $\mathbf{h} = [h_1, h_2 \ldots h_M]$, where it is assumed that $M < N$. In order to generate $\mathbf{x}$, one must first create the intermediate parameter $\boldsymbol{\alpha} = [\alpha_i, \ldots \alpha_N]$, by affine transformation $\boldsymbol{\alpha} = \mathbf{W}\mathbf{h} + \mathbf{b}$. Then, for each $1 \leq i \leq N$, x_i is generated according to the element distribution $p(x_i; \alpha_i)$. Conditioned on $\mathbf{h}$, the elements of $\mathbf{x}$ are independent.

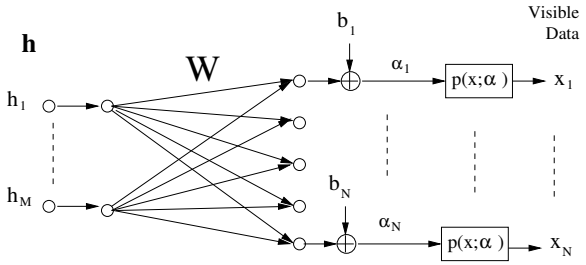

Fig. 2. One layer of a linear Bayesian network (LBN).

The sigmoid (Bernoulli) distribution is used for discrete-valued x_i, taking values of 0 or 1,

$$P(x_i = 1; \alpha_i) = \frac{1}{1 + e^{-\alpha_i}}. \tag{4}$$

An example of a continuous-valued element distribution often used in the literature is the Gaussian

$$p(x_i; \alpha_i, \sigma_i^2) = (2\pi\sigma_i^2)^{-1/2} \ \exp\left\{-\frac{(x_i - \alpha_i)^2}{2\sigma_i^2}\right\}. \tag{5}$$

For continuous data x_i limited to the range $[0,\ 1]$, it is more natural to use the *truncated exponential distribution* (TED) given by

$$p(x_i; \alpha_i) = \left(\frac{\alpha_i}{e^{\alpha_i} - 1}\right) \ e^{\alpha_i x_i}, \quad 0 \leq x_i \leq 1. \tag{6}$$

The mean of this density is

$$\lambda(\alpha) = \frac{e^\alpha}{e^\alpha - 1} - \frac{1}{\alpha}. \tag{7}$$

It is a characteristic of the duality between LBN and FF-NN, that the expected value of the element distributions, notably (4) and (7) correspond to activation functions, (2) and (3) respectively. In Figure 3, the TED non-linearity (3) is compared with the sigmoid function. It will be shown later that the TED element distribution and TED non-linearity have a special theoretical role in PBNs.

C. PDF Projection

Let $\mathbf{z} = T(\mathbf{x})$ be some deterministic mapping from $\mathcal{X} \in \mathcal{R}^N$ to $\mathcal{Z} \in \mathcal{R}^M$, where $M < N$ and where $\mathcal{X}$ and $\mathcal{Z}$ are the allowable ranges of random variables $\mathbf{x}$ and $\mathbf{z}$, respectively. Let $p(\mathbf{x}; H_0)$ be some known reference distribution with support on $\mathcal{X}$, and let $p(\mathbf{z}; H_0)$ be the corresponding distribution with full support (non-zero everywhere) on $\mathcal{Z}$ imposed by mapping

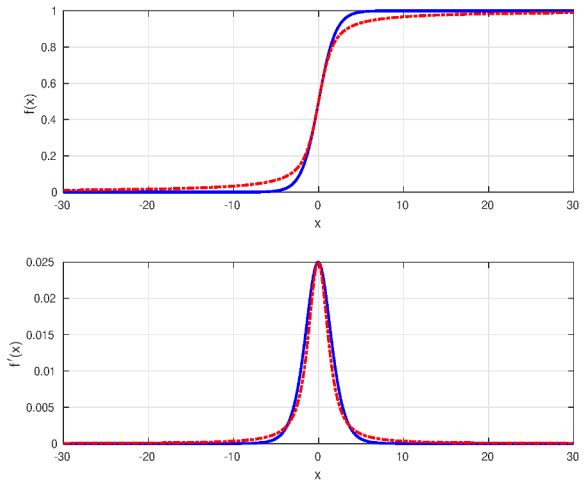

Fig. 3. Solid line: sigmoid function $f(x)$. Dotted line: TED non-linearity $\lambda(3x)$. Upper: function, Lower: derivative.

T. The projected probability density function (PDF) is given by

$$G(\mathbf{x}; H_0, T, g(\mathbf{z})) = \frac{p(\mathbf{x}; H_0)}{p(\mathbf{z}; H_0)} g(\mathbf{z}), \quad \mathbf{z} = T(\mathbf{x}), \tag{8}$$

where $g(\mathbf{z})$ is any distribution on $\mathcal{Z}$. It can be shown [5] that (8) is a PDF (integrates to 1 on $\mathcal{X}$), and is consistent with $g(\mathbf{z})$, meaning that samples drawn from $G(\mathbf{x}; H_0, T, g(\mathbf{z}))$ and mapped to $\mathcal{Z}$ using T will have exactly distribution $g(\mathbf{z})$. The PDF $g(\mathbf{z})$ is said to be "projected" from $\mathcal{Z}$ back to $\mathcal{X}$.

An approximation to an arbitrary distribution $p(\mathbf{x}; H)$ can be obtained by PDF projection, if $g(\mathbf{z})$ is replaced by the apropriate feature PDF estimate, denoted by $\hat{p}(\mathbf{z}; H)$. Then $G(\mathbf{x}; H_0, T, \hat{p}(\mathbf{z};H))$ will be a reasonable estimate of $p(\mathbf{x}; H)$. The PDF estimate can become exact under the condition of statistical sufficiency (See [6], p. 674. Sec. II.A, or [7], Sec. II.B) .

A generative model has two functions. First, it is a likelihood function, which is written down in equation (8). Second, it is a data generation recipe. Generation from (8) is a two-stage process [6] : (a) generate a sample of $g(\mathbf{z})$, denoted by $\mathbf{z}^*$, then (b) generate a sample $\mathbf{x}$ on the level set $\mathcal{M}(\mathbf{z}^*, T)$ (usually a manifold) defined by

$$\mathcal{M}(\mathbf{z}^*, T) = \{\mathbf{x} : T(\mathbf{x}) = \mathbf{z}^*, \quad \mathbf{x} \in \mathcal{X}\}, \tag{9}$$

with relative probability proportional to $p(\mathbf{x}; H_0)$. This second step, drawing a sample from the manifold, can be interpreted as drawing from the *a posteriori* distribution $p(\mathbf{x}|\mathbf{z}; H_0) = \frac{p(\mathbf{x};H_0)}{p(\mathbf{z};H_0)}$, which is not a proper distribution, having all its probability mass on the set $\mathcal{M}(\mathbf{z}^*, T)$ which, has zero volume [4], [5]. But, if the deterministic relationship of $\mathbf{x}$ and $\mathbf{z}$ is ignored, PDF projection has the form of Bayes rule: $G(\mathbf{x}; H_0, T, g(\mathbf{z})) = p(\mathbf{x}|\mathbf{z}; H_0) g(\mathbf{z})$.

D. Maximum Entropy PDF Projection

While PDF projection finds a PDF on $\mathcal{X}$ consistent with $g(\mathbf{z})$, it requires specifying an arbitrary reference distribution H_0. To find a unique PDF, one can seek the reference distribution that maximizes the entropy of the resulting projected PDF. Maximizing the entropy is a well established principle in PDF estimation [8], [5]. That allows the PDF estimate to express not only the available knowledge about $\mathbf{x}$, but also the ignorance. The entropy of a distribution $p(\mathbf{x})$ is given by

$$Q\{p\} = -\int_{\mathbf{x}} \log p(\mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}.$$

The entropy is maximized subject to constraints imposed by the available "knowledge" about p. For example, if the mean and variance are known, the Gaussian distribution has the highest entropy among all distributions with the given mean and variance [9]. In recent papers [4], [5], it is explained how to select H_0 in (8) so that $G(\mathbf{x}; H_0, T, g(\mathbf{z}))$ has the highest possible entropy. Let the H_0 that maximizes the entropy be denoted by H_0^*:

$$H_0^* = \arg\max_{H_0} Q\left\{G(\mathbf{x}; H_0, T, g(\mathbf{z}))\right\},$$

and the highest-entropy PDF be denoted by

$$G^*(\mathbf{x}; T, g(\mathbf{z})) = G(\mathbf{x}; H_0^*, T, g(\mathbf{z})).$$

This results in a unique generative model corresponding to the given transformation T and the given feature distribution $g(\mathbf{z})$.

The identity of H_0^* and the conditions under which it can be found depend on the range of $\mathbf{x}$, denoted by $\mathcal{X}$, but they do not depend on $g(\mathbf{z})$. Depending on $\mathcal{X}$, the selection of H_0 for maximum entropy may require appending the feature $\mathbf{z}$ with an *energy statistic* [5], denoted by $e = t(\mathbf{x})$. The complete transformation is then $T(\mathbf{x}) = [\mathbf{z},\ e]$. An energy statistic is a scalar feature measuring information about the size of $\mathbf{x}$, for example a norm $\|\mathbf{x}\|$. Important is that there exists a function f such that $f(T(\mathbf{x})) = \|\mathbf{x}\|$, for some valid norm $\|\mathbf{x}\|$. The type of reference hypothesis and norm depends on $\mathcal{X}$. Three cases will be discussed in Sections III-B through III-D.

E. UMS: Generating from the Maximum Entropy Projected PDF $G^*(\mathbf{x}; T, g(\mathbf{z}))$

Interestingly, despite the dependence of H_0^* on $\mathcal{X}$, the maximum entropy (ME) sampling procedure for $G^*(\mathbf{x}; T, g(\mathbf{z}))$ is always the same: (a) generate a sample $\mathbf{z}^*$ from $g(\mathbf{z})$, then (b) generate a sample $\mathbf{x}$ on the level set $\mathcal{M}(\mathbf{z}^*, T)$ with *uniform probability*. In other words, choose $\mathbf{x}$ randomly from $\mathcal{M}(\mathbf{z}^*, T)$, insuring that no member of $\mathcal{M}(\mathbf{z}^*, T)$ is more likely to be chosen that any other. This process is called uniform manifold sampling (UMS) [4], [5] and is the mathematical basis of the PBN.

III. Projected Belief Network (PBN)

A. Definition and General Form

A PBN arises when ME-PP is applied to a feed-forward neural network (FF-NN), i.e. by applying (8) and selecting H_0 for maximum entropy. The PBN is best understood by first analyzing the linear transformation alone without any bias or non-linearity. Consider the linear transformation

$$\mathbf{z} = T(\mathbf{x}) = \mathbf{A}'\mathbf{x}, \tag{10}$$

where $\mathbf{A}$ is assumed to be a full-rank $N \times M$ matrix. The ME projected PDF is

$$G^*(\mathbf{x}; T, p(\mathbf{z})) = \frac{p(\mathbf{x}; H_0^*)}{p(\mathbf{z}; H_0^*)}\, g(\mathbf{z}), \tag{11}$$

which is the likelihood function of the PBN. The basic PBN is illustrated in Figure 4 which shows the optional energy statistic $e = t(\mathbf{x})$. The energy statistic is needed so that H_0 can be selected for maximum entropy, but only when the range of $\mathbf{x}$ can be unbounded (exponential or Gaussian assumption), but is not needed for data values limited to the interval $[0,\ 1]$. Drawing samples from $G^*(\mathbf{x}; T, g(\mathbf{z}))$ requires UMS, which is illustrated in the box on the left side of the figure. The two sides of the figure are duals, and can be seen as inverse processes because the low-dimensional variable $\mathbf{z}$ (and energy statistic e if present) is exactly reproduced at the output.

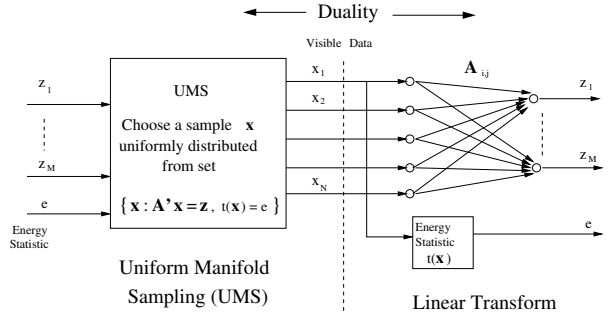

Fig. 4. Exact form of the most basic type of PBN corresponding to a linear transformation.

The UMS function on the left of Figure 4 does not have a recognizeable network structure. But, for large N, UMS can be closely approximated by an LBN. Figure 5 shows the

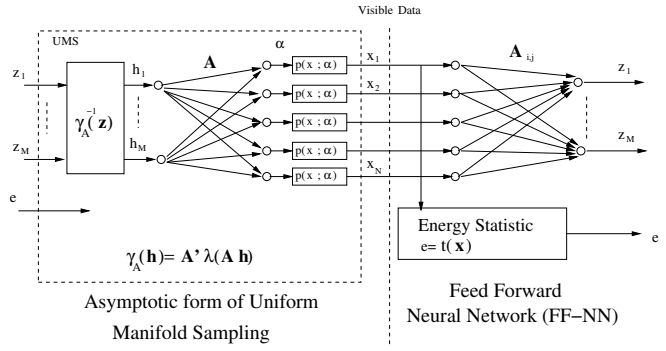

Fig. 5. Type I (inference) form of a PBN (left) and its dual FF-NN (right). Element distributions, energy statistics, and function $\lambda(\ )$ are shown in Table I. What is inside the dotted box is the asymptotic (large N) approximation to UMS.

asymptotic form of Figure 4, which is an LBN, but differs in two ways from the LBN in Figure 2: it has no explicit bias term $\mathbf{b}$, and at the input, there is a non-linear function

$$\mathbf{h} = \gamma_A^{-1}(\mathbf{z}) \tag{12}$$

that operates jointly (not element-wise) on the input variable $\mathbf{z}$. Function γ_A^{-1} is the inverse of the function $\mathbf{z} = \gamma_A(\mathbf{h})$, which is defined as the conditional mean $\mathbf{z} = \gamma_A(\mathbf{h}) = \mathcal{E}(\mathbf{z}|\mathbf{h})$, in other words the expected value of the output of the network on the right hand side of Figure 5 conditioned on knowing the intermediate variable $\mathbf{h}$. Looking at Figure 5, one can write γ_A as follows:

$$\gamma_A(\mathbf{h}) = \mathbf{A}'\lambda\left(\mathbf{A}\mathbf{h}\right), \tag{13}$$

where $\lambda(\alpha)$ operates element-wise on the vector $\boldsymbol{\alpha} = \mathbf{A}\mathbf{h}$ and is defined as the conditional mean of x given α, $\lambda(\alpha) = \mathcal{E}(x|\alpha)$, and is only a function of the element distribution $p(x;\alpha)$. From this definition of $\gamma_A(\mathbf{h})$, it is clear that the inverse function $\gamma_A^{-1}(\mathbf{h})$ cancels the effect of the network, so that the expected value of $\mathbf{z}$ on the right hand side of Figure 5 will equal the input $\mathbf{z}$ on the left hand side. Thus, whereas the exact PBN of Figure 4 exactly reproduces $\mathbf{z}$ at the output, the asymptotic dual pair in Figure 5 reproduces $\mathbf{z}$ in the mean.

The function $\lambda(\alpha)$ for various cases of input data range $\mathcal{X}$ are provided in Table I. These cases will be discussed in detail below. For completeness, the binary case $\mathcal{B}^N$ is included, where x_i can be either 0 or 1.

Range $\mathcal{X}$	Element Distr. $p(x;\alpha)$	Function $\lambda(\alpha) = \mathcal{E}(x\|\alpha)$	Energy Stat. t(**x**)
$\mathcal{R}^N$	$\frac{\exp\left\{\frac{-(x-\alpha)^2}{2\sigma^2}\right\}}{\sqrt{2\pi\sigma^2}}$	$\lambda(\alpha) = \alpha$	$\sum_{i=1}^N x_i^2$
$\mathcal{P}^N$	$\alpha e^{\alpha x}$	$\lambda(\alpha) = 1/\alpha$	$\sum_{i=1}^N x_i$
$\mathcal{U}^N$	$\left(\frac{\alpha}{e^\alpha - 1}\right) e^{x\alpha}$	$\lambda(\alpha) = \frac{e^\alpha}{e^\alpha - 1} - \frac{1}{\alpha}$	N/A
$\mathcal{B}^N$	$P_1 = \frac{1}{1+e^{-\alpha}}$ $P_0 = 1 - P_1$	$\lambda(\alpha) = \frac{1}{1+e^{-\alpha}}$	N/A

TABLE I
ELEMENT DISTRIBUTIONS, ENERGY STATISTICS, AND FUNCTION $\lambda(\alpha)$ FOR EACH CASE OF $\mathcal{X}$. IN ALL CASES, $\gamma_A(\mathbf{h}) = \mathbf{A}'\lambda(\mathbf{A}\mathbf{h})$, WHERE $\lambda(\boldsymbol{\alpha})$ OPERATES ELEMENT-WISE ON THE ARGUMENT $\boldsymbol{\alpha}$.

B. PBN for $\mathbf{x} \in \mathcal{R}^N$

The case where $\mathcal{X} = \mathcal{R}^N$ happens when the data is not strictly limited to a given range, for example when the input data is the output of microphones, sensors, or has been averaged. It could also be, for example, the logarithm of power spectra, intensity images, or energy data. In a previous publication (in [5], Section III.B), the case of UMS for the linear transformation $\mathbf{z} = \mathbf{A}'\mathbf{x}$ for $\mathbf{x} \in \mathcal{R}^N$ (unconstrained case) is covered in detail. The main results are now presented.

When the range of $\mathbf{x}$ is unlimited, it is necessary to append the second-order energy statistic $t_2(\mathbf{x}) = \sum_{i=1}^N x_i^2$. So, for example, the feature can be augmented with $t_2(\mathbf{x})$ or preferably $\log t_2(\mathbf{x})$. For $\mathcal{X} = \mathcal{R}^N$, the Gaussian reference hypothesis will produce maximum entropy:

$$p(\mathbf{x}; H_0^*) = \prod_{i=1}^N \frac{1}{\sqrt{2\pi}} e^{-x_i^2/2} = (2\pi)^{-N/2}\ e^{-t_2(\mathbf{x})/2}. \tag{14}$$

Given a pair t_2^*,$\mathbf{z}^*$, drawing a sample $\mathbf{x}$ using UMS is done as follows. (a) form the vector $\boldsymbol{\alpha} = \mathbf{A}\left(\mathbf{A}'\mathbf{A}\right)^{-1}\mathbf{z}^*$, (b) using Gramm-Schmidt procedure, form the ortho-normal $N \times (N - M)$ matrix $\mathbf{B}$ spanning the linear subspace orthogonal to $\mathbf{A}$, (c) form the $(N - M) \times 1$ vector $\tilde{\mathbf{u}}$ of independent Gaussian random variables of mean zero and variance 1, and (d) let $\mathbf{x} = \boldsymbol{\alpha} + \mathbf{B}\mathbf{u}$, where $\mathbf{u} = \frac{\tilde{\mathbf{u}}}{\sqrt{\|\tilde{\mathbf{u}}\|^2}}\sqrt{t_2^* - \|\boldsymbol{\alpha}\|^2}$. Drawing a sample $\mathbf{x}$ in this way will exactly reproduce the features $\mathbf{z}^*$, t_2^*. For more information on computing $G^*(\mathbf{x}; T, {}_{g(\mathbf{z})})$, see [4] (Section IV.C, page 2821). For more information on sampling $G^*(\mathbf{x}; T, {}_{g(\mathbf{z})})$, refer to [5] Section III.B.

For the asymptotic case (large N), data created by UMS for a fixed feature $\mathbf{z}^*, t_2^*$ will be approximately Gaussian with mean $\boldsymbol{\alpha}$ with independent Gaussian noise of variance $\sigma^2 = \frac{(t_2^* - \|\boldsymbol{\alpha}\|^2)}{N}$. It can be seen that if data from this distribution is reduced to features, it approximately reproduces $\mathbf{z}^*, t_2^*$. The entry for $\mathcal{X} = \mathcal{R}^N$ in Table I shows that $\lambda(\alpha) = \alpha$.

C. PBN for $\mathbf{x} \in \mathcal{P}^N$ (Positive Quadrant)

Let $\mathcal{P}^N$ be the positive quadrant of $\mathcal{R}^N$, where for all i, $x_i > 0$, but are unlimited in size. This is the case for power spectra or intensity data. When $\mathcal{X} = \mathcal{P}^N$, it is necessary to append $\mathbf{z}$ the first-order energy statistic [5] $t_1(\mathbf{x}) = \sum_{i=1}^N x_i$, which can be integrated into matrix $\mathbf{A}$. For $\mathbf{x} \in \mathcal{P}^N$, the exponential reference hypothesis produces highest entropy:

$$p(\mathbf{x}; H_0^*) = \prod_{i=1}^N e^{-x_i} = e^{-t_1(\mathbf{x})}. \tag{15}$$

In a previous publication (in [5], Section IV), the case of UMS for the linear transformation $\mathbf{z} = \mathbf{A}'\mathbf{x}$ when $\mathbf{x}$ is constrained to $\mathcal{P}^N$ is covered in detail. Instead of defining a separate energy statistic $e = t(\mathbf{x})$, one can insure that it is "contained" in $\mathbf{z}$, for example by specifying one column of $\mathbf{A}$ to be a constant.

Given a fixed feature $\mathbf{z}^*$, UMS is implemented by drawing a sample $\mathbf{x}$ from the set (9), most efficiently accomplished using "hit and run", a type of MCMC for uniform sampling on linear, convex manifolds [10]. The main challenge in computing $G^*(\mathbf{x}; T, {}_{g(\mathbf{z})})$ is calculating the distribution $p(\mathbf{z}; H_0)$ which can be done using the saddle point approximation (See [11], page 2244-2246). For more information on computing $G^*(\mathbf{x}; T, {}_{g(\mathbf{z})})$, see [4] (Section IV.B, page 2820). For more information on sampling $G^*(\mathbf{x}; T, {}_{g(\mathbf{z})})$, refer to [5] Section IV.

In the asymptotic case (large N), data $\mathbf{x}$ created by UMS for a fixed feature $\mathbf{z}^*$ will have independent elements x_i that are exponentially distributed

$$p(x_i; \alpha_i) = \alpha_i e^{\alpha_i x_i},$$

with mean $\lambda(\alpha_i) = \mathcal{E}\{x_i|\alpha_i\} = \frac{1}{\alpha_i}$, where $\boldsymbol{\alpha} = \mathbf{A}\mathbf{h}$, and the vector $\mathbf{h}$ is the solution to the equation $\mathbf{A}'\lambda(\mathbf{A}\mathbf{h}) = \mathbf{z}^*$, where $\lambda(\alpha) = 1/\alpha$ operates element-wise. If data from this distribution is reduced to features, it approximately reproduces $\mathbf{z}^*$. The asymptotic form of UMS is illustrated in Figure 5 with functions defined in Table I.

D. PBN for $\mathbf{x} \in \mathcal{U}^N$ (Unit Hypercube)

In this case, $\mathbf{x}$ is limited to the unit hypercube, denoted by $\mathcal{X} = \mathcal{U}^N$, meaning that x_i is in the range $[0, 1]$ for all i. This case is most common in a neural network, especially in hidden layers. For this case, no energy statistic is needed and the uniform reference hypothesis $p(\mathbf{x}; H_0^*) = 1$ will produce highest entropy [5]. Therefore, (11) simplifies to

$$G^*(\mathbf{x}; T, {}_{g(\mathbf{z})}) = \frac{1}{p(\mathbf{z}; H_0^*)} g(\mathbf{z}), \quad \mathbf{z} = T(\mathbf{x}). \tag{16}$$

The computation of $p(\mathbf{z}; H_0^*)$ is derived in appendix of [7].

In a previous publication (in [5], Section V), the case of UMS for the linear transformation $\mathbf{z} = \mathbf{A}'\mathbf{x}$ when $\mathbf{x}$ is constrained to the unit hypercube is covered in detail. As was true for the case of $\mathcal{X} = \mathcal{P}^N$, UMS is most efficiently accomplished using "hit and run", a type of MCMC for uniform sampling on linear, convex manifolds [10], but the sampling procedure is slightly modified to account for the added upper bound [5].

For large N, UMS can be shown to be asymptotically the same as sampling from the multi-variate TED distribution

$$p(\mathbf{x}; \mathbf{h}) = \prod_{i=1}^N \left(\frac{\alpha_i}{e^{\alpha_i} - 1}\right) e^{\alpha_i x_i}, \quad 0 \le x_i \le 1, \tag{17}$$

where $\alpha_i = \sum_{m=1}^M A_{i,m} h_m$, which can be written (in matrix notation)

$$p(\mathbf{x}|\mathbf{h}) = C(\boldsymbol{\alpha}) e^{\mathbf{x}'\boldsymbol{\alpha}}, \tag{18}$$

where $C(\alpha) = \left(\frac{\alpha}{e^{\alpha} - 1}\right)$ operates element-wise on a vector. This distrubution has mean

$$\boldsymbol{\mu} = \lambda(\boldsymbol{\alpha}). \tag{19}$$

The function $\lambda(\,)$ is tabulated in Table I and in this case is the TED non-linearity (7) applied element-wise. The asymptotic form of UMS is illustrated in Figure 5. Note that the energy statistic t is not needed [1].

Solving the equation $\mathbf{h} = \gamma_A^{-1}(\mathbf{z})$ can be accomplished using the method of [5], Section V.C. This solution is rapid and is based on a Newton-Raphson iteration to find the $\mathbf{h}$ that minimizes the square error $q = \|\mathbf{z} - \mathbf{A}'\lambda(\mathbf{A}\mathbf{h})\|^2$. The algorithm terminates when q is driven to zero. The solution may not exist, but will always exist if there is at least one element in the set (9), which will always be the case if $\mathbf{z}$ was generated by the forward network.

As an aside, the special non-linear function $\mathbf{h} = \gamma_A^{-1}(\mathbf{z})$ is related to the solution of the saddle point for $p(\mathbf{z}; H_0^*)$ for $\mathcal{X} = \mathcal{U}^N$ (and also for $\mathcal{X} = \mathcal{P}^N$). In fact, the hidden variable $\mathbf{h}$ *is* the saddle point.

[1] However, it has been observed that by including the first-order statistic $t_1(\mathbf{x})$ into the column space of matrix $\mathbf{A}$ often results in a better model.

E. Discussion

The mathematical details for PBNs corresponding to three input data ranges have been presented in Sections III-B through III-D. It is interesting that the maximum entropy dual counterpart of a feed-forward network for a given input data range $\mathcal{X}$ is an LBN with a specific element distribution. In the case $\mathcal{X} = \mathcal{R}^N$, it is the Gaussian distribution. In the case $\mathcal{X} = \mathcal{P}^N$, it is the exponential distribution. And in the case $\mathcal{X} = \mathcal{U}^N$, it is the TED distribution. This occurs because as N becomes large, the distribution of UMS tends toward the maximum entropy distribution under mean (and/or variance) constraints (See appendix in [5]). Most interesting is the case of $\mathcal{X} = \mathcal{U}^N$ because this is the case most often encountered in neural networks. In linear inference, one seeks to "infer" the distribution of data that has been observed through a dimension-reducing transformation. Just like Gaussian distributions are the most "natural" choice for unbounded data, the TED distribution is the most natural choice for continuous-valued data that is limited to the unit interval [0, 1].

Missing in the discussion is the binary (Bernoulli) type of network, which is in common use. PDF projection, which is the basis of the PBN, is not well suited to binary networks because of the intractability of the distribution of linear functions of binary random variables, and of sampling from a manifold within the binary input data space. This does not diminish the importance of the discussion because the TED distribution serves as the continuous cousin of the Bernoulli distribution. Even the energy function of the TED and Bernoulli RBMs are identical [7].

F. PBN with bias and non-linearity

Most FF-NN networks used in practice, such as in Figure 1, include bias $\mathbf{c}$ and non-linearity $f(\ )$, These can be easily added to the PBN, but must be inverted at the input of the PBN, as illustrated in Figure 6. This form of the PBN is called Type I (inference form) because it is based on the inference latent variables $\mathbf{z}$ or $\mathbf{f}$.

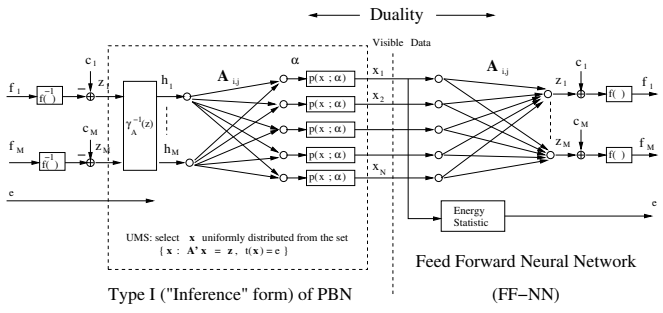

Fig. 6. Form I (inference form) of PBN including optional bias and activation functions. The dotted box suggests that the generation of $\mathbf{x}$ can be accomplished either with UMS (exact form) or with the asymptotic generative network shown. See Table I for element distributions, energy statistics, and function $\lambda(\ )$ which is needed to construct $\gamma_A(\mathbf{h})$ according to (13).

To apply PDF projection to an FF-NN with bias and activation function, the transformation is broken into two parts, $\mathbf{z} = T(\mathbf{x}) = \mathbf{A}'\mathbf{x}$, followed by the invertible 1:1 transformation $f_i = f(z_i + c_i)$. Let the full transformation be denoted by $\mathbf{f} = T_f(\mathbf{x})$. Due to the 1:1 transformation, the distributions of $\mathbf{z}$ and $\mathbf{h}$ are related by $p(\mathbf{z}) = |J_f(\mathbf{z})|\ p(\mathbf{f})$, where $|J_f(\mathbf{z})|$ is the absolute value of the determinant of the Jacobian matrix $J_f(\mathbf{z}) = \left[\frac{\partial f_i}{\partial z_j}\right]$. This gives

$$G^*(\mathbf{x}; T_f, p(\mathbf{f})) = \frac{p(\mathbf{x}; H_0^*)\ |J_f(\mathbf{z})|}{p(\mathbf{z}; H_0^*)}\ p(\mathbf{f}). \qquad (20)$$

Equation (20) defines the exact likelihood function for the PBN which is the dual of the FF-NN in Figure 1 and on the right side of Figure 6. It assumes $\mathbf{f}$ is a random variable drawn from the given distribution $p(\mathbf{f})$. To sample from (20), (a) draw a sample $\mathbf{f}^*$ from $p(\mathbf{f})$, (b) convert to $\mathbf{z}^*$ by inverting the bias and non-linearity, then (c) draw a sample $\mathbf{x}$ uniformly from the set $\mathcal{M}(\mathbf{z}^*, T)$ defined in (9), where feature $\mathbf{z}$ would include the energy statistic e, if present. Since the FF-NN on the right side of Figure 6 is the exact dual of the UMS generative process, the variable $\mathbf{f}$ is exactly reproduced at the output.

G. Linear Bayesian Network (LBN) with bias

Linear Bayesian networks used in practice (Figure 2) have an explicit bias term $\mathbf{b}$. This can be implemented by adding the bias vector $\mathbf{b}$ as a column to matrix $\mathbf{A}$,

$$\mathbf{A}_b = [\mathbf{A} \mid \mathbf{b}]$$

and driving this column with a fixed scalar element h_0 appended to vector $\mathbf{h}$. This is shown in Figure 7 on the left side. The addition of this term implies that the added column should also appear in the dual FF-NN network, producing an extra output z_0. It has been shown, for example, that the added column is necessary based on sufficiency arguments when applied to the restricted Boltzmann machine [7]. This is called the PBN Form II (generative form) because it is based on the generative latent variable $\mathbf{h}$.

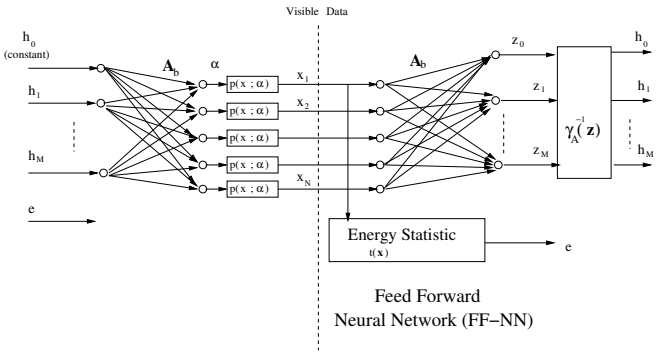

Fig. 7. Type II (generative) form of a PBN with bias term (left) and its dual FF-NN (right). The bias vector $\mathbf{b}$ is appended to the weight matrix $\mathbf{A}$ and is driven by the constant input h_0.

H. Multi-layer PBN

A multi-layer PBN is just an application of the chain-rule known to PDF projection ([6], page 676, Section II.D). A

two-layer PBN is formed recursively when $p(\mathbf{f})$ is replaced by the projected PDF of the second layer, itself expressed as (20), with $\mathbf{f}$ taking the role of $\mathbf{x}$. This recursion repeats for an arbitrary numbers of layers. Consider a 3-layer network. Let $\mathbf{x}$ be the network input, $\mathbf{f}_1$ and $\mathbf{f}_2$ the outputs of the first two layers, and $\mathbf{z}_3$ the output of the linear transformation in the third layer. The chain rule would be written

$$G^*(\mathbf{x}, g(\mathbf{z})) = \frac{p(\mathbf{x}; H_0^1)|J_f(\mathbf{z}_1)|}{p(\mathbf{z}_1; H_0^1)} \frac{p(\mathbf{f}_1; H_0^2)|J_f(\mathbf{z}_2)|}{p(\mathbf{z}_2; H_0^2)} \frac{p(\mathbf{f}_2; H_0^3)}{p(\mathbf{z}_3; H_0^3)} g(\mathbf{z}_3),$$

where H_0^k is the hypothesis that layer k input is the uniform distribution (or whatever the MaxEnt reference hypothesis is for that layer). Note that the Jacobian is missing in the last layer - at the output of the network it is prefereble to estimate the distribution of $\mathbf{z}$ before the non-linearity. The MaxEnt property also extends to the chain rule [4], so that $G^*(\mathbf{x}, g(\mathbf{z}))$ will be the distribution on $\mathcal{X}$ with maximum entropy among all distributions consistent with $g(\mathbf{z})$.

Sampling also occurs recursively, starting from the last layer. Sampling multi-layer PBNs suffers from a flaw: a UMS-generated sample may not be a "valid" output of the layer before it. In other words, the manifold (9) may be an empty set for some values of $\mathbf{z}^*$. The efficiency (the fraction of UMS-generated samples that are valid outputs of the previous layer) can potentially be very low. An LBN does not suffer from this problem. However, the efficiency of the sampling process is an increasing function of the likelihood function (20) of the following layer. Therefore, in PBN design, increasing sampling efficiency goes hand in hand with improved PDF design (improved feature extraction) of the down-stream layers.

I. Applications

The inference form (Form I, Figures 5 and 6) and generative form (Form II, Figure 7) of the PBN can be used to find the dual counterparts of existing LBN or FF-NN, respectively. Applications of this include

1) Training the FF-NN as a generative model. Examples are provided in Sections IV-A and V-C.
2) Training the generative PBN by training an FF-NN with back-propagation (See Section V-B).
3) Approximating the hidden variables of an LBN using the dual FF-NN.
4) Creating an auto-encoder from an arbitrary weight matrix without the need for a complementary reconstruction matrix. This is demonstrated in Section IV-C.

IV. Illustrative Experiments

A. Training and Sampling a PBN

In this simple example, it is demonstrated that equation (20) can be used as a likelihood function that can be maximized to train the weights of the FF-NN as a generative model. It is also shown that random samples can be generated from the corresponding generative model. For illustrative putposes, the dimension was kept very low. A FF-NN with sigmoid activation function was created with input data dimension is $N = 4$ and output feature dimension $M = 2$. The input data range was the unit hypercube $\mathcal{U}^N$. Training data consisted of 1000 samples of independent Gaussian-distributed samples with mean $[.6,\ .4,\ .5,\ .5]$, and variance .000625 (care was taken that there were no samples outside the range $[0,\ 1]$). The matrix $\mathbf{A}$ and constant $\mathbf{c}$ were randomly initialized, and a uniform feature PDF $p(\mathbf{f}) = 1$ was assumed. Note that since $p(\mathbf{x}; H_0^*) = 1$, and $p(\mathbf{f}) = 1$, (20) reduces to

$$G^*(\mathbf{x}; T_f, p(\mathbf{f})) = \frac{|J_f(\mathbf{z})|}{p(\mathbf{z}; H_0^*)}. \tag{21}$$

The network was trained by maximizing the log-likelihood averaged over n samples of training data $L = \frac{1}{n}\sum_{k=1}^{n} \log G^*(\mathbf{x}_k; T_f, p(\mathbf{f})=1)$. The PDF $p(\mathbf{z}; H_0^*)$ was computed by the saddle-point approximation (See appendix in [7]). Maximization of L was accomplished by gradient-based steepest descent. The gradient $\frac{\partial}{\partial\theta}L$ was calculated in closed form for all parameters θ including elements of $\mathbf{A}$ and $\mathbf{c}$.

Once L was maximized, $G^*(\mathbf{x}; T_f, p(\mathbf{f})=1)$ was computed on a grid on a slice through $\mathcal{X}$ at $x_3 = .5$, $x_4 = .5$. This is displayed in Figure 8 on the left side. A contour plot of $G^*(\mathbf{x}; T_f, p(\mathbf{f})=1)$ is plotted on the right side along with some synthetic data that will be explained below. When evaluated on 2-D plane in the 4-D space, $G^*(\mathbf{x}; T_f, p(\mathbf{f})=1)$, looks very much like other types of PDF estimates, such as Gaussian mixture model (GMM), with probability density concentrated near the training data.

To test the generation capability of the PBN, 20,000 samples of synthetic data were generated using the procedure at the end of Section III-F. Of the 20,000 generated samples, all were discarded except about 100 samples that were in the slice $.48 \le x_3 \le .52$, $.48 \le x_4 \le .52$. These were plotted on the right side of Figure 8 (dots). Note that these samples agree very well with the density contours that were computed on the slice at $(x_3, x_4) = (.5, .5)$.

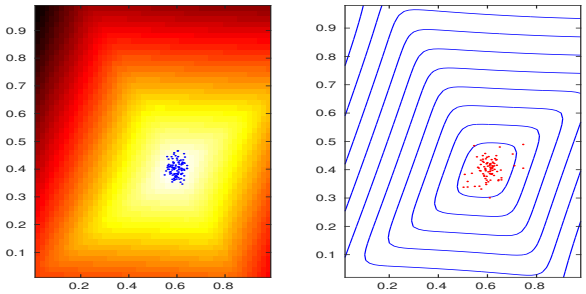

Fig. 8. Visualization of the implied (projected) PDF for a perceptron layer. Left: intensity plot of PDF slice at $(x_3, x_4) = (.5, .5)$ with samples of training data (dots). Right, contour plot of PDF slice at $(x_3, x_4) = (.5, .5)$ with samples of synthetic (UMS-generated) data.

B. Validating Asymptotic Duality

In the last example, it was demonstrated that a PBN can be trained and samples can be drawn that match the theoretical distribution. In this example, it is demonstrated that the PBN can be approximated by the asymptotic form illustrated on the left side of Figure 6, for the case $\mathcal{X} = \mathcal{U}^N$. In contrast to the

previous example, it is necessary to use a higher dimension: $N = 128$, $M = 4$. A FF-NN (as depicted on the right side of Figure 6) was created. Matrix $\mathbf{A}$ was chosen to be the $N \times M$ matrix of discrete cosine transform (DCT) basis functions. The bias vector $\mathbf{c}$ was a random $M \times 1$ vector, and $f(\ )$ was the sigmoid function (2).

Experiments were conducted with a fixed feature vector $\mathbf{z}$ that was created by first generating a random input vector $\mathbf{x}$ in the unit hypercube [2], then letting $\mathbf{z} = \mathbf{A}'\mathbf{x}$.

In the first experiment, data was generated by UMS (i.e the exact form of the PBN), and using the asymptotic PBN, and the samples were compared. The UMS samples were generated on the manifold (9) using method of ([5] Section V.A). To generate samples from the asymptotic PBN, samples were draw from the conditional TED distribution (18), where $\boldsymbol{\alpha} = \mathbf{A}\gamma_A^{-1}(\mathbf{z})$. Figure 9 compares the average of 3000 UMS-generated samples with the theoretical mean of the conditional TED distribution, which equals $\lambda(\boldsymbol{\alpha})$, where $\lambda(\ )$ is the TED mean (7). There is very close agreement.

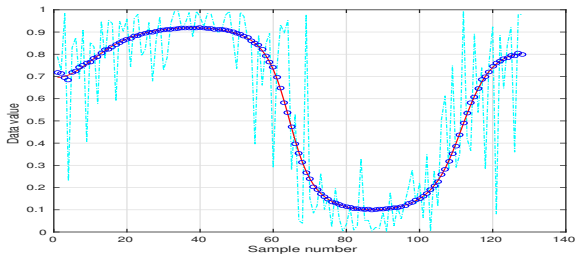

Fig. 9. Light dashed line: one random UMS sample of $\mathbf{x}$. Solid red line: theoretical TED mean (19), Circles: average of 3000 UMS samples.

As an additional comparison of the distributions, Figure 10 shows scatter-plots of $\mathbf{x}$ projected on a randomly-chosen pair of indexes, both for UMS-generated data (left) and for the asymptotic TED distribution (right). There is also good agreement.

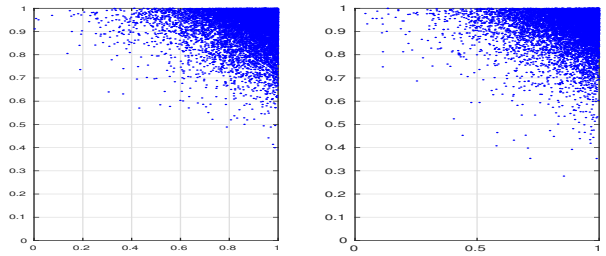

Fig. 10. Left: scatter plot of UMS-generated samples (two randomly-selected elements). Right: samples generated according to the TED distribution.

The above experiments have compared the distribution of samples generated by UMS and by the asymptotic TED distribution. The conditional likelihood functions are now compared given a fixed value of $\mathbf{z}$. For the exact form of the PBN, with PDF (16), the conditional distribution is just $p(\mathbf{x}|\mathbf{z}; H_0^*) = \frac{1}{p(\mathbf{z}; H_0^*)}$, which can be compared with the asymptotic approximation (18). This approximation can be written

$$\frac{1}{p(\mathbf{z}; H_0^*)} \simeq C(\boldsymbol{\alpha}) e^{\mathbf{x}'\boldsymbol{\alpha}}, \tag{22}$$

where $\boldsymbol{\alpha} = \mathbf{A}\gamma_A^{-1}(\mathbf{z})$. In each of 100 trials, random sample $\mathbf{x}$ was computed as described above, then the log of both sides of (22) were plotted on the two axes of Figure 11. Although there is a small bias that increases as $p(\mathbf{x})$ is larger, they track very well. Note that the approximation gets better for increasing N (see appendix in [5]), and the value used in the experiment was relatively small ($N = 128$).

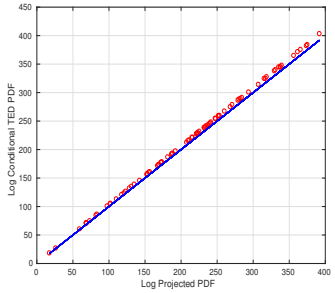

Fig. 11. Conditional projected PDF compared with conditional TED distribution (circles). Solid line signifies X=Y.

C. Auto-encoder

In this example, the duality relationship is used in order to create an auto-encoder with tied analysis and reconstruction weights. Given an arbitrary FF-NN, as illustrated on the right side of Figure 6 with sigmoid activation function, the task is to create an auto-encoder, by following the FF-NN by the asymptotic form of the PBN, which is the conditional TED distribution explained in Section IV-B. This approximately recovers $\mathbf{x}$. For deterministic reconstruction, the random TED element distribution (6) is replaced with the TED non-linearity (3). Note that the non-linearity used in the FF-NN does not need to be the same as the one used for deterministic reconstruction - this one needs to match the asymptotic element distribution - in this case TED.

To demonstrate the idea, an arbitrary 1-layer FF-NN was created with $M = 12$ nodes. The columns of the weight matrix were initialized to the top P singular values obtained by principal component analysis (PCA) analysis of the training data from 200 samples of character "8" from the MNIST-389R corpus (See section V-A). The bias vector $\mathbf{c}$ was initialized to provide zero-mean to the input to the sigmoid non-linearity. Examples of re-constructed samples are shown in Figure 12.

V. Classification Experiments

In this section, a PBN is trained from both sides of the duality relationship, first as a FF-NN, and then directly as a generative model, and then used in a classification experiment.

[2] $\mathbf{x}$ was created by passing data of the form $\mathbf{A}\mathbf{u}$, where $\mathbf{u}$ is Gaussian, through the sigmoid function to force it in the range $[0,\ 1]$.

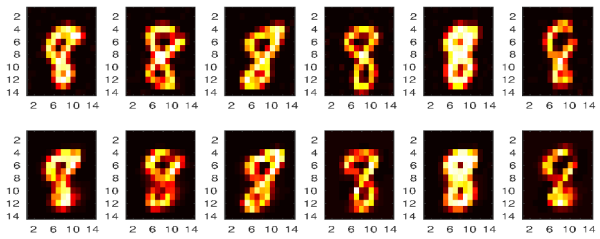

Fig. 12. Examples of reconstructed data from an auto-encoder constructed from an FF-NN followed by its dual SBN. Top row: original data. Bottom row: deterministic reconstruction.

A. MNIST Data description

The MNIST OCR data corpus [12] set consists of ten handwritten digits 0-9 divided into two sub-corpora: the training sub-corpus with about 6000 training samples of each digit, and the testing sub-corpus has 10,000 samples, about 1000 testing samples of each digit. In addition to the full MNIST corpus, a reduced MNIST corpus was created to facilitate processing-costly experiments. The reduced corpus is called MNIST-389R and consists of just the digits "3", "8", and "9". Train and test data was gathered into one pool, then 200 samples per class were randomly chosen for training. The 28×28 images were also 2:1 down-sampled to 14×14. In the classifier experiments on the reduced corpus, the classification performance was measured in eight independent trials, each trial using a different random set of 200 training examples per class, and testing using all the remaining samples (20324).

The MNIST and MNIST-R data is positive-valued in the range [0,1]. In addition to the original data, "expanded" data was created by passing the pixel values into the inverse sigmoid function. Most of the expanded data was in the range [-10, 10].

Four existing classifiers were used to get a benchmark for performance on the reduced corpus.

1) The Gaussian mixture model (GMM) was applied directly to the raw data. Both full covariance matrices and diagonal covariance matrices were tried, and performance was measured using both expanded and non-expanded data. In all cases, a "variance floor" was implemented by adding the value $\rho^2 \hat{\sigma}_i^2$ to the i-th position of the diagonal of the covariance matrix, where $\hat{\sigma}_i^2$ is the sample variance of feature i. The best performance achieved on non-expanded data was 8.5%, and the best performance on expanded data was 5.85% with $d = 2$ mixture components and $\rho = .85$.
2) The SVM Light toolbox [13] was used to create a support vector machine (SVM) classifier. Best performance was obtained using polynomial kernel which achieved 4.5% with a standard deviation of 0.14%.
3) Deep neural network. The PDNN toolkit [14] was used to pre-train a deep network using stacked RBMs, then fine-tune using back-propagation. The network specification was "196:196:196:32:3", which means an input dimension of 196 (14×14), followed by hidden layers of 196, 196, and 32 units, and ending with an output layer of three units. The average performance over eight independent trials was 3.76% with a standard deviation of 0.37%.
4) Hinton's deep belief network (DBN), a generative model based on the restricted Boltzmann machine (RBM) [3] with one layer of 100 hidden units and a top-layer RBM with 300 hidden units. The network was trained using the up-down algorithm [3], then tested using the free energy approach as described in Hinton's paper. The DBN achieved average 3.28% error.

Benchmark performances for MNIST-389R data are tabulated in Table II.

Classifier	Expand	Error	Std. Dev.	Conditions
GMM	N	8.5%	0.4%	$\rho = .85$, 2 components
GMM	Y	5.85%	0.2%	$\rho = .85$, 2 components
SVM	N	4.5%	0.14%	polynomial kernel
DBN	N	3.28%	0.25%	100:300
DNN	N	3.76%	0.37%	196:196:196:32:3
PBN/CSFM	N	3.4%	0.16%	$M = 36$ Auto-Encoder
PBN/CSFM	N	3.2%	0.12%	$M = 36$ Gen-Opt

TABLE II
CLASSIFIER PERFORMANCE ON MNIST-389R.

B. Training a generative model using an auto-encoder FF-NN.

In this example, a generative classifier based on the PBN is trained by training the dual FF-NN as an auto-encoder with back-propagation. In the simplest form of an auto-encoder, a two-layer FF-NN is constructed with reduced dimension at the output of the first layer. In the second layer, the dimension is expanded back to the dimension of the input. The network is then trained using back-propagation to reconstruct the visible data at the network output, then the second layer is discarded. Using the MNIST-389R data with 200 training samples from each class, a separate class-dependent auto-encoder was trained on each of the digits "3", "8", and "9" with a hidden layer dimension of $M = 36$. The weights of the auto-encoder were randomly initialized.

After the auto-encoder networks were trained, the second layer and the bias and non-linearities of the first layer were removed, resulting in the basic form of the PBN, which has likelihood function (11). Let $\mathbf{z}_l = T_l(\mathbf{x})$, where l ranges over the classes, be the feature transformation of the network trained on data from class l, and let $\hat{p}(\mathbf{z}_l|H_m)$ be the PDF estimate of feature $\mathbf{z}_l$ made using data from class m. Let $G^*(\mathbf{x}; T_l, \hat{p}(\mathbf{z}_l|H_m))$, be the PBN likelihood function created from the network that was trained on class l, but with output distribution trained on class m. A straight class-specific classifier [15] can be formed as follows

$$\arg\max_m G^*(\mathbf{x}; T_m, \hat{p}(\mathbf{z}_m|H_m)),$$

but this classifier is effectively "putting all eggs in one basket" because class hypothesis H_m is tested only using feature $\mathbf{z}_m$.

A better approach is the class-specific feature mixture (CSFM), which is given by

$$p(\mathbf{x}|H_m) = \sum_{l=1}^{n} w_{l,m} G^*(\mathbf{x}; T_l, \hat{p}(\mathbf{z}_l|H_m)),$$

where n is the number of classes ($n = 3$ here), and where $\sum_{l=1}^{n} w_{l,m} = 1$. This type of classifier has been found to have superior performance compared to a straight class-specific classifier [16], [17], [5]. A further improvement upon the CSFM is attained by using an annealed mixture [18], [5], [17] given by

$$p(\mathbf{x}|H_m; C) = \frac{1}{K} \left(\sum_{l=1}^{n} w_{l,m} G^*(\mathbf{x}; T_l, \hat{p}(\mathbf{z}_l|H_m))^{1/C} \right)^C, \quad (23)$$

where C is a heurisic parameter that functions like the temperature parameter in the "softmax" function, and where the constant K is the normalizing constant required to make the annealed mixture a PDF. This constant can be found by Monte Carlo integration [5], [18], but in these experiments, it was ignored, and $K = 1$ was used. Values of C that were used are provided in Table III. Mixture weights of $w_{l,m} = \frac{1+\epsilon\,\delta[l-m]}{1+\epsilon}$ were used, where ϵ is a heurisic parameter that determines how much $p(\mathbf{x}|H_m; C)$ "prefers" the network trained on class m over the networks trained on the other classes. A straight class-specific classifier results if $\epsilon \longrightarrow \infty$. When estimating $\hat{p}(\mathbf{z}_l|H_m)$, a Gaussian mixture of d mixture components and covariance floor factor ρ was used. Parameter values are provided in Table III.

A maximum likelihood classifier was formed by maximizing (23) over m. Classification performance was measured as the auto-encoders were trained. The experiment was repeated 8 independent times, each time with a random set of 200 training samples, using all the remaining samples for testing. In Figure 13, the classification performance is seen as a function of training iterations[3]. As the auto-encoder trains, the mean square error decreases and the classification performance generally improves. A mean classification performance of 3.4% was achieved after 100 iterations. This performance improves upon the benchmark GMM performance of 5.85% and SVM performance of 4.5%, and even surpasses the DNN performance.

Training	Hidden nodes	ρ	C	ϵ	d
Auto-enc.	36	.4	100	.02	3
Generative.	36	.5	150	.3	3

TABLE III
CSFM PARAMETERS USED IN THE EXPERIMENTS. ρ IS THE "VARIANCE FLOOR" PARAMETER, C IS THE ANNEALING FACTOR, ϵ IS USED TO COMPUTE THE MIXING WEIGHTS, AND d IS THE NUMBER OF MIXTURE COMPONENTS FOR THE GMM.

[3]In each "iteration" 10 steps of steepest descent were performed to minimize the mean square auto-encoder reconstruction error.

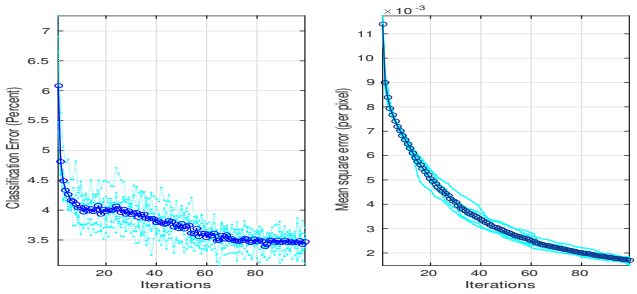

Fig. 13. Classification performance of auto-encoder-trained network on MNIST-389R data as a function of training iterations for $M = 36$ hidden nodes. Each of the 8 independent trials (dotted) are shown as well as the overall mean (solid, circles) .

C. Training a generative model directly (Generative Training).

In the last example, the PBN was trained by training its dual, the FF-NN using back-propagation. Now, the PBN is trained directly bu maximizing the likelihood function, showing that it is possible to train the same network from either side of the duality. The experiment in Section V-B was repeated, except that back-propagation was replaced by maximizing the likelihood function as explained in Section IV-A. To simplify the training, it is temporarily assumed that the output distribution is uniform $p(\mathbf{f}) = 1$, so that the likelihood function reduces to (21). After training, the estimated feature PDF is used, as in Section V-B. Parameters are tabulated in Table III. Classifier results on MNIST-389R data are shown in Figure 14 for $M = 36$ hidden nodes as a function of iteration. As the networks train, the average log-likelihood L generally increases (L was measured using separate testing data) and the classification error decreases. The direct-trained generative networks had better performance than the auto-encoder-trained networks. In fact, a performance of 3.2% is achieved after 40 iterations, significantly improving upon the DNN which achieved 3.76%, and surpassing the DBN. This suggests using the more efficient auto-encoder training first, then fine-tuning with generative training.

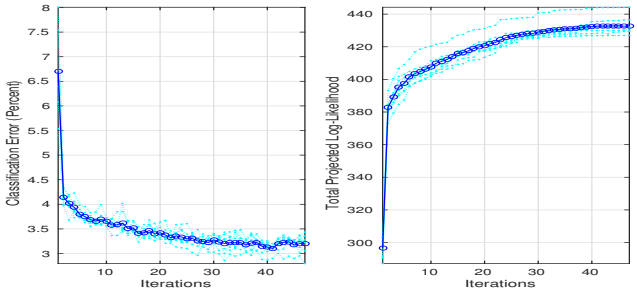

Fig. 14. Classification performance of generative trained network on MNIST-389R as a function of training iterations for $M = 36$ hidden nodes. Left: classification error, right: total log likelihood. Each of the 8 independent trials (dotted) are shown as well as the overall mean (solid, circles).

The quality of a generative model can be evaluated not only by its classification error performance, but also by the quality

of the synthetic data it can generate. Samples of character "9" were randomly selected from the MNIST-389R corpus, which are shown on the left column of Figure 15. A network as described in Section IV-C with $M = 15$ was initialized using PCA. Reconstructed digits are shown in the center column of Figure 15. Reconstruction was done again after generative training (right column of Figure 15). Notice the improved

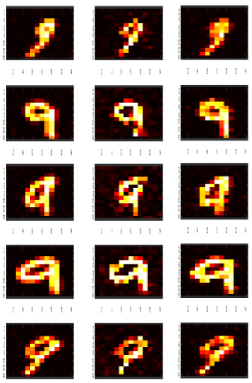

Fig. 15. Five samples from MNIST-389R, character "9" reconstructed from 15-dimensional feature. Left column: original data. Center column: Reconstructed from network initialized using PCA. Right column, reconstructed after generative training.

quality afforded by generative training, such that the UMS-generated sample looks almost identical to the original data.

D. Performance on the full MNIST data set.

The PBN is now tested as a generative classifier on the full MNIST data set. To obtain a performance benchmark, the state of the art classifiers listed in Section V-A were applied. SVM Light toolbox attained 181 errors (1.81%). Using PDNN toolkit with network specification "784:500:500:10" and stacked-RBM pre-training, the DNN attained 200 errors (2.00%). The performance of Hinton's deep belief net (DBN) of 1.25% is taken from published results [3]. In addition, a deep convolutional neural network (CNN) was tested using the PDNN toolkit [14]. The first convolutional layer had 20 5×5 kernels with 2:1 pooling. The second layer used 50 5×5 kernels with 2:1 pooling. These were followed by a neural network with 512 hidden units, ending with an output layer of 10 units. This network achieved 95 errors (0.95%).

Then, the PBN-based CSFM generative model with generative training described in Section V-C was tried on the full MNIST data set. A CSFM model was created using PBNs trained on each class with with $M = 32$ hidden units. Separate classifiers for expanded and non-expanded data were created. Note that for the expanded data, the PBN described in Section III-B is required, whereas Section III-D is apropriate for the non-expanded data. The PBN-CSFM attained 1.41% error on the non-expanded data, and 1.37% on the expanded data. Interestingly, although the performances are silimar, the two approaches produce sufficiently independent results that a linear combination of the two methods is advantageous (see Figure 16). Best performance of 1.28% was achieved for equal weighting (linear combination factor = 1). This performance is on par with state of the art "permutation invariant" state of the art classifiers (not taking advantage of correlations between adjacent pixels or distortions), which does not include the CNN.

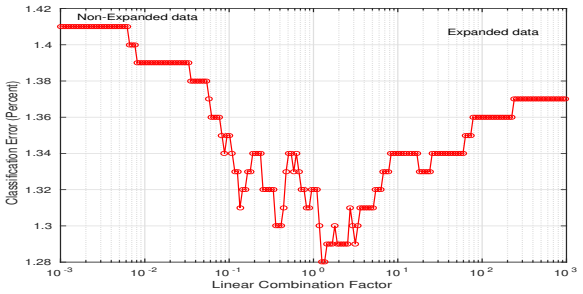

Fig. 16. Linear combination results for combining two CSFMs (designed for expanded and non-expanded data).

			Settings				
Classifier	Expand	Error	M	ρ	C	ϵ	d
SVM	N	1.81%	polynomial kernel				
DNN	N	2.00%	196:500:500:10				
DBN	N	1.25%	500:500:2000				
CNN	N	0.95%	20(5x5):50(5x5):512:10				
CSFM	N	1.41%	32	.25	1000	.13	14
CSFM	Y	1.37%	32	.25	500	.015	14
CSFM (comb)	N+Y	1.28%					

TABLE IV
CLASSIFIER PERFORMANCE ON FULL MNIST CORPUS. FOR A DEFINTION OF SETTINGS SYMBOLS, SEE TABLE III.

VI. CONCLUSIONS

In this paper, the projected belief network (PBN) was created by applying maximum entropy PDF projection to a feed-forward neural network. The exact and asymptotic (large N) forms of the PBN were described. The asymptotic form of the PBN is a linear Bayesian network employing a special non-linear function followed by multiplication by the matrix $\mathbf{A}$, which is the same matrix used in the feed-forward network, followed by random data generation using certain element distributions. The special non-linear function and the element distributions depend on the range of the input (visible) data $\mathbf{x}$. The details of the PBN are described for three cases of input data range. In the case of data in the unit hypercube, which is the most common case in neural networks, the element distributions are the truncated exponential distribution (TED). This establishes the theoretical importance of TED in machine learning. The duality relationship between the FF-NN and the PBN has been validated and demonstrated in several experiments. For example, it was demonstrated that a PBN-based classifier can be trained from either side of the duality - as a FF-NN or as a PBN. On the MNIST corpus, a classifier based on the PBN achieved state of the art performance for permutation-invariant classifiers.

REFERENCES

[1] S. Ramachandran and R. J. Mooney, "Revising Bayesian network parameters using back-propagation," in *Proceedings of the 1996 IEEE International Conference on Neural Networks*, pp. 86–87, Jun 1996.

[2] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The "wake-sleep" algorithm for unsupervised neural networks," *Science*, vol. 268, pp. 1158–1161, May 1995.

[3] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," in *Neural Computation 2006*, 2006.

[4] P. M. Baggenstoss, "Maximum entropy PDF design using feature density constraints: Applications in signal processing," *IEEE Trans. Signal Processing*, vol. 63, June 2015.

[5] P. M. Baggenstoss, "Uniform manifold sampling (UMS): Sampling the maximum entropy pdf," *IEEE Transactions on Signal Processing*, vol. 65, pp. 2455–2470, May 2017.

[6] P. M. Baggenstoss, "The PDF projection theorem and the class-specific method," *IEEE Trans Signal Processing*, pp. 672–685, March 2003.

[7] P. M. Baggenstoss, "Evaluating the RBM without integration using pdf projection," in *Proceedings of EUSIPCO 2017, Island of Kos, Greece*, Aug 2017.

[8] E. T. Jaynes, "On the rationale of maximum-entropy methods," *Proceedings of IEEE*, vol. 70, no. 9, pp. 939–952, 1982.

[9] J. N. Kapur, *Maximum Entropy Models in Science and Engineering*. Wiley (Eastern), 1993.

[10] S. Kiatsupaibul, R. Smith, and Z. Zabinsky, "An analysis of a variation of hit-and-run for uniform sampling from general regions," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 21, no. 3, 2011.

[11] S. M. Kay, A. H. Nuttall, and P. M. Baggenstoss, "Multidimensional probability density function approximations for detection, classification, and model order selection," *IEEE Transactions on Signal Processing*, vol. 49, pp. 2240–2252, Oct 2001.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[13] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods - Support Vector Learning* (B. Schölkopf, C. Burges, and A. Smola, eds.), ch. 11, pp. 169–184, Cambridge, MA: MIT Press, 1999.

[14] Y. Miao, "Kaldi+PDNN: Building DNN-based ASR systems with Kaldi and PDNN," in *arXiv:1401.6984*, 2014.

[15] P. M. Baggenstoss, "Class-specific features in classification.," *IEEE Trans Signal Processing*, pp. 3428–3432, December 1999.

[16] P. M. Baggenstoss, "Optimal detection and classification of diverse short-duration signals," in *Proceedings of the International Conference on Cloud Engineering*, (Boston, MA), pp. 534–539, 2014.

[17] P. M. Baggenstoss, "Class-specific model mixtures for the classification of acoustic time-series," *IEEE Trans. AES*, Aug. 2016.

[18] P. M. Baggenstoss, "Maximum entropy pdf projection: A review," in *Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Proceedings of the 36th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering (MaxEnt 2016)*, pp. 070001.1–070001.8, July 2016.