# Function Optimization with Latent Variable Models

**Soo-Yong Shin**
Artificial Intelligence Lab (SCAI)
School of Computer Sci. & Eng.
Seoul National University
Seoul 151-742, Korea.
syshin@scai.snu.ac.kr

**Dong-Yeon Cho**
Artificial Intelligence Lab (SCAI)
School of Computer Sci. & Eng.
Seoul National University
Seoul 151-742, Korea.
dycho@scai.snu.ac.kr

**Byoung-Tak Zhang**
Artificial Intelligence Lab (SCAI)
School of Computer Sci. & Eng.
Seoul National University
Seoul 151-742, Korea.
btzhang@cse.snu.ac.kr

**Abstract**

Most of estimation of distribution algorithms (EDAs) try to represent explicitly the relationship between variables with factorization techniques or with graphical models such as Bayesian networks. In this paper, we propose to use latent variable models such as Helmholtz machine and probabilistic principal component analysis for capturing the probabilistic distribution of given data. The latent variable models are statistical models that specify the relationships between a set of random variables and a set of latent variables; Latent variables are not directly observable and there are a smaller number of variables in latent variables than in input variables. In statistics latent variable models are used for density estimation. Since latent variable models are generative models, it is easy to sample a new data. Our experimental results support that the proposed latent variable models can find good solutions more efficiently than other EDAs for continuous functions.

## 1 Introduction

Recently, a number of algorithms known as the estimation of distribution algorithms (EDAs) have been proposed to explicitly model the population of good solutions and use the resulting constructed model to guide further search. Instead of using local information through crossover or mutation of individuals, they use global information contained in the population. From the population, statistics of the hidden structure are derived and used for generating new individuals.

One of the main issues in this field is how to estimate the accurate distribution that can capture the structure of the given problem. The simplest way of distribution estimation is to assume that variables are independent. Population-based incremental learning (PBIL) [1], univariate marginal distribution algorithm (UMDA) [13], and compact genetic algorithm (cGA) [8] for the discrete space and PBILc [17] for the continuous space belong to this class. However they are not appropriate for learning an arbitrary interdependence. To capture the pairwise dependencies, mutual information maximizing input clustering (MIMIC) [7], dependency tree algorithms [2], and bivariate marginal distribution algorithms (BMDAs) [15] were proposed. Chain, tree, and forest structures were used in each algorithm respectively. For more complex relationship, the factorized distribution algorithm (FDA) [14] and the Bayesian optimization algorithm (BOA) [16] were suggested. To search good probability density models in continuous spaces, Bosman et al. [4] proposed the integrated density estimation evolutionary algorithm (ID𝔼A) as a general EDA framework, and Larrañaga *et al.* [11] replaced the Bayesian network with the Gaussian network on the continuous domain.

Except the simplest ones, all these methods use distribution models to explicitly represent the relationship between variables in the problems. However, determining the best model with respect to a given score metric is usually NP-complete. That is, enormous time is required for building the models when the problem size is large. Thus most researchers used greedy approaches to prevent this ill-behavior although the exact distributions cannot be estimated.

Recently, Zhang and Shin [20, 18] developed another type of EDA, where Helmholtz machine is used to model and sample from the distribution of selected individuals without explicit expression of multivariate interactions. Also Cho and Zhang [5] propose a new distribution estimation algorithm with probabilistic principal component analysis (PPCA) which can also cover higher order interactions as in Helmholtz machine. These approaches do not explicitly represent the relationship between variables, but implicitly capture the distribution using the latent variables. Since there is no explicit search procedure for the probability density structure, it is possible to rapidly estimate the distribution. And it is also easy to sample the new individuals from generative models.

The paper is organized as follows. In Section 2 we explain the latent variable models. Section 3 presents the EDA with PPCA and Helmholtz machine for the continuous domain. Section 4 reports the results of experiments for some benchmark functions and Section 5 summarizes our findings in this study.

## 2 Latent Variable Models

Usually underlying concepts are not directly observable. We rather observe their "outcroppings" or effects on observable variables. To be confident that differences among observations on variables really do reflect "true" differences in the underlying concepts, it is very useful to have multiple measures or indicators of a supposed underlying latent variable or factor.

The goal of a latent variable model is to express the distribution $p(\mathbf{x})$ of the variables $x_1, \ldots, x_d$ in terms of a smaller number of latent variables $\mathbf{z} = (\mathbf{z_1}, \ldots, \mathbf{z_q})$ where $q < d$. This is achieved by first decomposing the joint distribution

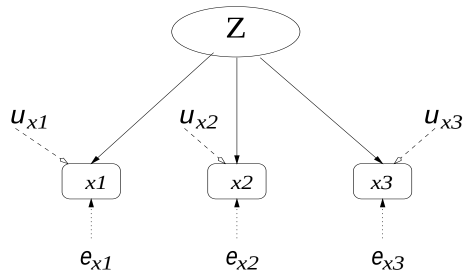

Figure 1: The latent variable model as a diagram. (*u* = unique variance, *e* = random error)

$p(\mathbf{x},\mathbf{z})$ into the product of the marginal distribution $p(\mathbf{z})$ of the latent variables and the conditional distribution $p(\mathbf{x}|\mathbf{z})$ of the data variables given the latent variables. The conditional distribution $p(\mathbf{x}|\mathbf{z})$ is expressed in terms of a mapping from latent variables to data variables with a noise, so that

$$\mathbf{x} = \mathbf{y}(\mathbf{z};\mathbf{w}) + \boldsymbol{\epsilon}, \tag{1}$$

where $\mathbf{y}(\mathbf{z};\mathbf{w})$ is a function of the latent variable $\mathbf{z}$ with parameters $\mathbf{w}$, and $\boldsymbol{\epsilon}$ is a $\mathbf{z}$-independent noise process.

The desired model for the distribution $p(\mathbf{x})$ of the data is obtained by marginalizing over the latent variables

$$p(\mathbf{x}) = \int \mathbf{p}(\mathbf{x}|\mathbf{z})\mathbf{p}(\mathbf{z})\mathbf{dz}. \tag{2}$$

One of the simplest latent variable models is *factor analysis*. Figure 1 shows the conceptual diagram of latent variable model.

## 2.1 Probabilistic Principal Component Analysis

Principal component analysis (PCA) is a powerful technique in data analysis. It can significantly reduce the dimensionality of the data by searching for the direction in data-space which have the highest variance, and subsequently projecting the data onto it [10].

For a set of observed *d* dimensional data vector $\{\mathbf{x}_i\}$, $i \in \{1, 2, ..., N\}$, we define the *q* principal axes $\{\mathbf{w}_j\}$, $j \in \{1, 2, ..., q\}$ as those orthonormal axes onto which the retained variance under projection is maximal. Then it can be shown that the vector $\{\mathbf{w}_j\}$ are given by the *q* dominant eigenvectors which correspond to the largest eigenvalues of the sample covariance matrix

$$\mathbf{S} = \frac{1}{N}\sum_{i}^{N}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}}, \tag{3}$$

where $\boldsymbol{\mu}$ is the data sample mean,

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i}^{N}\mathbf{x}_i, \tag{4}$$

such that $\mathbf{S}\mathbf{w}_j = \lambda_j\mathbf{w}_j$. The *q* principal components of the observed data $\mathbf{x}_i$ are given by the vector $\mathbf{z}_i = \mathbf{W}^{\mathrm{T}}(\mathbf{x}_i - \boldsymbol{\mu})$, where $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_q)$. The variables $z_j$ are uncorrelated such that the covariance matrix $\sum_i \mathbf{z}_i\mathbf{z}_i^{\mathrm{T}}/N$ is diagonal with elements $\lambda_j$. A complementary property of PCA is that the principal component projection of all orthogonal linear projections minimizes the squared reconstruction error $\sum_i \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$, where the optimal linear reconstruction of $\mathbf{x}_i$ is given by $\hat{\mathbf{x}}_i = \mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}$.

However, PCA is not a probabilistic model thus Tipping and Bishop [19] addressed this limitation by using the latent variable model which is closely related to factor analysis.

A latent variable model tries to relate a *d* dimensional data **x** to a corresponding *q* dimensional latent variables **z**. In standard factor analysis, the relationship is linear:

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{5}$$

where the latent variables $\mathbf{z} \sim N(0, \mathbf{I})$ have a unit isotropic Gaussian and the noise is Gaussian $\boldsymbol{\epsilon} \sim N(0, \boldsymbol{\Sigma})$ with diagonal covariance matrix $\boldsymbol{\Sigma}$. From this formulation, the data **x** has also Gaussian distribution $\mathbf{x} \sim N(\boldsymbol{\mu}, \mathbf{W}\mathbf{W}^{\mathrm{T}} + \boldsymbol{\Sigma})$.

The key assumption for this model is that the observed variables $x_i$ are conditionally independent given the values of the latent variables **z** because of the diagonality of $\boldsymbol{\Sigma}$. Thus these latent variables are intended to explain the correlations between observation variables while $\boldsymbol{\epsilon}$ represents independent noise.

For the isotropic Gaussian noise model of $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2\mathbf{I})$, equation (5) implies that **z** conditional probability distribution over **x**-space is given by $\mathbf{x}|\mathbf{z} \sim N(\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$, i.e,

$$p(\mathbf{x}|\mathbf{z}) = (2\pi\sigma^2)^{-d/2}\exp\left\{-\frac{\|\mathbf{x} - \mathbf{W}\mathbf{z} - \boldsymbol{\mu}\|^2}{2\sigma^2}\right\}. \tag{6}$$

When the marginal distribution of the latent variable **z** is unit Gaussian;

$$p(\mathbf{z}) = (2\pi)^{-q/2}\exp\left\{-\frac{1}{2}\mathbf{z}^{\mathrm{T}}\mathbf{z}\right\}, \tag{7}$$

the marginal distribution of the observed data **x** is obtained by integrating out the latent variables as follows:

$$\begin{aligned} p(\mathbf{x}) &= \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \\ &= (2\pi)^{-d/2}|\mathbf{C}|^{-1/2}\exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\}, \end{aligned} \tag{8}$$

where the covariance is specified by $\mathbf{C} = \mathbf{W}\mathbf{W}^{\mathrm{T}} + \sigma^2\mathbf{I}$ and implying $\mathbf{x} \sim N(\boldsymbol{\mu}, \mathbf{C})$.

By Bayes' rule, the posterior distribution of the latent variables **z** given the observed **x** becomes:

$$\begin{aligned} p(\mathbf{z}|\mathbf{x}) = (2\pi)^{-q/2}|\sigma^{-2}\mathbf{M}|^{1/2} \\ \times \exp\Big[-\frac{1}{2}\Big\{\mathbf{z} - \mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu})\Big\}^{\mathrm{T}}(\sigma^{-2}\mathbf{M}) \\ \Big\{\mathbf{z} - \mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu})\Big\}\Big], \end{aligned} \tag{9}$$

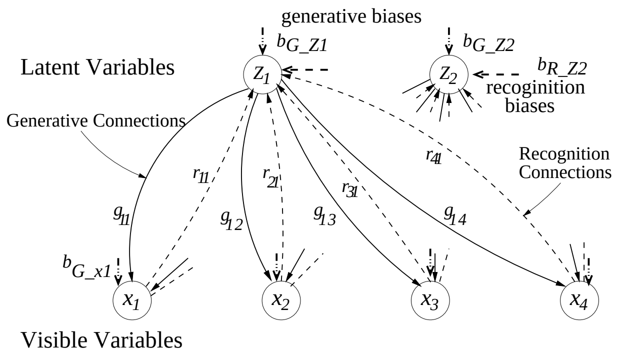

Figure 2: The Helmholtz machine (two-layer network).

where $\mathbf{M} = \mathbf{W}^{\mathrm{T}}\mathbf{W} + \sigma^2\mathbf{I}$.

The log-likelihood of observing the data under this model is

$$\begin{aligned} L &= \sum_{i=1}^{N} \ln\{p(\mathbf{x}_i)\} \\ &= -\frac{N}{2}\{d\ln(2\pi) + \ln|\mathbf{C}| + \mathrm{tr}(\mathbf{C}^{-1}\mathbf{S})\}, \end{aligned} \quad (10)$$

where $\mathbf{S}$ is the sample covariance matrix given by equation (3). Although there is no closed form analytic solution for $\mathbf{W}$ and $\sigma$, the parameters for this model can be obtained by iterative procedure, e.g. by using expectation-maximization (EM) algorithms of the next section.

### 2.2 Helmholtz Machine

The Helmholtz machine [6] uses two entirely different sets of connections: the generative model and recognition model. A Helmholtz machine is illustrated in Figure 2 for the case of a network with two layers of stochastic units. The units in the lowest layer of the network are observable since the inputs are observed on them; units in the upper layer of the network are latent, since they are not directly observable from inputs.

The recognition model uses the bottom-up direction recognition biases and weights $\mathbf{R} = \{\mathbf{r_{ij}}, \mathbf{b_{R_{Y_j}}}\}$. This model is to infer a probability distribution over the underlying causes $\mathbf{z}$ of the input vector $\mathbf{x}$,

$$P(\mathbf{z}|\mathbf{x}, \mathbf{R}), \quad (11)$$

$$p(z_i) = \sigma\left(b_{R_{Y^i}} + \sum_i \mathbf{r_{ij}x_i}\right), \quad (12)$$

where $\sigma(x) = 1/(1 + exp(-x))$ is the conventional sigmoid function.

The generative model uses the top-down generative biases and weights, $\mathbf{G} = \{\mathbf{g_{ij}}, \mathbf{b_{G_{Y_i}}}, \mathbf{b_{G_{X_i}}}\}$. The purpose of this model is to reconstruct an approximation to the original input vector from the underlying representations captured by the latent variables,

$$P(\mathbf{x}|\mathbf{z}, \mathbf{G}). \quad (13)$$

1. (**Initialize**) Randomly generate the initial population of *M* individuals from the prior distribution. Set generation count $g \leftarrow 0$.
2. (**Selection**) Select *N* promising solutions.
3. (**Density Estimation**) Calculate the distribution of given data using a latent variable model.
4. (**Generate**) Sampling $L$ variations by the latent variable model.
5. (**Loop**) Set $g \leftarrow g + 1$ and go to Step 2.

Figure 3: Outline of the estimation of distribution algorithms with latent variable models.

The original Helmholtz machine has binary stochastic units. Therefore we modified the Helmholtz machine to capture continuous distribution. First, we assumed the joint probability function of continuous $d$-dimensional input vector $\mathbf{x}$ to be a multivariate normal distribution, and put generative noise $\nu_G$ and recognition noise $\nu_R$ with Gaussian distribution $N(\mathbf{b}, \boldsymbol{\tau})$ to the input units to give randomness; where $\mathbf{b}$ is a bias, and $\boldsymbol{\tau}$ is a variance of noise. Second, we used sigmoid function instead of changing the value to 0 or 1. Also to cover the range of given data vectors, we put the slope parameter $\alpha$ to the sigmoid function; $\sigma'(x) = 1/(1 + \alpha\exp(-x))$.

The representation $z$ produced in the latent layer is

$$\mathbf{z} = \sigma'\left(\mathbf{xR} + \boldsymbol{\nu}_R\right). \quad (14)$$

similarly the generated data vectors $x$ are produced by

$$\mathbf{x} = \sigma'\left(\mathbf{zG} + \boldsymbol{\nu}_G\right). \quad (15)$$

## 3 Estimation of Distribution Algorithms with Latent Variable Models

Conceptual estimation of distribution algorithm (EDA) procedure is used in our proposed algorithms. The proposed methods use the latent variable models to recognize the relationship between the input data. The latent variable model has been used for density modelling [3]. By marginalizing a joint distribution of visible and latent variables, the corresponding distribution of the observed variables can be obtained.

The outline of the proposed algorithms is explained in the Figure 3 and more detailed explanation will be given.

### 3.1 Distribution Estimation by PPCA

The selected *N* individuals among the current population of size *M* are regarded as the observed data and the EM approach of maximizing the likelihood for PPCA is employed. Through this procedure, the distribution of the data points can be estimated.

For a selected individual $\mathbf{x}_i$, the corresponding value of $\mathbf{z}_i$ is unknown. Since the joint distribution $p(\mathbf{x}, \mathbf{z})$ of the given samples and latent variables is known, we can calculate the expectation of the corresponding log-likelihood. In the E-step of the EM algorithm, the expectation with respect to the posterior distribution of $\mathbf{z}_i$ given the selected $\mathbf{x}_i$ is computed. In the M-step, new parameter values of $\mathbf{W}$ and $\sigma^2$ are determined that maximize the expected log-likelihood.

Using equations (6) and (7), the log-likelihood is defined as follows:

$$\begin{aligned} L &= \sum_{i=1}^{N} \ln\{p(\mathbf{x}_i, \mathbf{z}_i)\} \\ &= \sum_{i=1}^{N} \ln \Big[(2\pi\sigma^2)^{-d/2} \exp\Big\{ -\frac{\|\mathbf{x}_i - \mathbf{W}\mathbf{z}_i - \boldsymbol{\mu}\|^2}{2\sigma^2}\Big\} \\ &\qquad \times (2\pi)^{-q/2} \exp\Big\{ -\frac{1}{2}\mathbf{z}_i^{\mathrm{T}}\mathbf{z}_i\Big\}\Big]. \end{aligned} \quad (16)$$

In the E-step, we take the expectation of $L$ with respect to the distribution $p(\mathbf{x}_i|\mathbf{z}_i, \mathbf{W}, \sigma^2)$:

$$\begin{aligned} E\{L\} = -\sum_{i=1}^{N} \Big[&\frac{d}{2}\ln\sigma^2 + \frac{1}{2}\mathrm{tr}\big(E\{\mathbf{z}_i\mathbf{z}_i^{\mathrm{T}}\}\big) \\ &+ \frac{\|\mathbf{x}_i - \boldsymbol{\mu}\|^2}{2\sigma^2} - \frac{1}{\sigma^2}E\{\mathbf{z}_i\}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}(\mathbf{x}_i - \boldsymbol{\mu}) \\ &+ \frac{1}{2\sigma^2}\mathrm{tr}\big(\mathbf{W}^{\mathrm{T}}\mathbf{W}E\{\mathbf{z}_i\mathbf{z}_i^{\mathrm{T}}\}\big)\Big], \end{aligned} \quad (17)$$

where we omitted terms independent of the model parameters and

$$\begin{aligned} E\{\mathbf{z}_i\} &= \mathbf{M}^{-1}\mathbf{W}^{\mathrm{T}}(\mathbf{x}_i - \boldsymbol{\mu}), &(18)\\ E\{\mathbf{z}_i\mathbf{z}_i^{\mathrm{T}}\} &= \sigma^2\mathbf{M}^{-1} + E\{\mathbf{z}\}E\{\mathbf{z}\}^{\mathrm{T}}, &(19) \end{aligned}$$

with $\mathbf{M} = \sigma^2\mathbf{I} + \mathbf{W}^{\mathrm{T}}\mathbf{W}$. Note that these statistics are computed using the current values of the parameters that follows from distribution (9).

In the M-step, the expectation of log-likelihood $E\{L\}$ is maximized with respect to $\mathbf{W}$ and $\sigma^2$ by differentiating equation (17) and setting the derivatives to be zero. This gives new parameter estimates

$$\mathbf{W}_{\mathrm{new}} = \Big[\sum_{i=1}^{N}(\mathbf{x}_i - \boldsymbol{\mu})E\{\mathbf{z}_i\}^{\mathrm{T}}\Big]\Big[\sum_{i=1}^{N}E\{\mathbf{z}_i\mathbf{z}_i^{\mathrm{T}}\}\Big]^{-1}, \quad (20)$$

$$\begin{aligned} \sigma^2_{\mathrm{new}} = \frac{1}{Nd}\sum_{i=1}^{N}\Big[&\|\mathbf{x}_i - \boldsymbol{\mu}\|^2 - 2E\{\mathbf{z}_i\}^{\mathrm{T}}\mathbf{W}_{\mathrm{new}}(\mathbf{x}_i - \boldsymbol{\mu}) \\ &+ \mathrm{tr}\big(E\{\mathbf{z}_i\mathbf{z}_i^{\mathrm{T}}\}\mathbf{W}_{\mathrm{new}}^{\mathrm{T}}\mathbf{W}_{\mathrm{new}}\big)\Big]. \end{aligned} \quad (21)$$

To maximize the likelihood, the sufficient statistics of the conditional distributions are calculated form the E-step equations (18) and (19), and revised estimates for the parameters are obtained from the M-step equations (20) and (21). These four equations are repeated sequentially until convergence is achieved or some other stopping criterion is met.

The conditional distribution of the new individual $\mathbf{x}'_i$ given the corresponding latent variable $\mathbf{z}_i$ is defined to be Gaussian, $N \sim (\mathbf{W}\mathbf{z}_i + \boldsymbol{\mu}, \sigma^2\mathbf{I})$, as equation (6). Thus to generate next population, we only have to sample $M/N$ data points for each latent variable $\mathbf{z}_i$ from the Gaussian distribution with parameters obtained by the previous EM algorithm.

### 3.2 Helmholtz Machine for Density Estimation

Hinton et al. [9] describes a stochastic algorithm, called wake-sleep algorithm to calculate the recognition and generative weights of the Helmholtz machine. We slightly modified the original wake-sleep algorithm to capture continuous distribution. There are two phases in the algorithm: a *wake* phase and a *sleep* phase. In the *wake* phase, from observed values of input variables, $\mathbf{x}$, values of the latent variables are randomly filled, $\mathbf{z}$, using the conditional distribution defined by the current recognition model. Although the nodes are driven by the recognition weights, only generative weights are actually learned during the wake phase using locally available information and the simple delta rule:

$$\Delta\mathbf{G} \propto \eta(\mathbf{x_c} - \mathbf{G}\mathbf{z_c})\mathbf{z_c}, \quad (22)$$

where $\mathbf{G}$ is the generative weight vector, $\mathbf{x_c}$ is the $c$-th sample, $\mathbf{z_c}$ is the value of the latent variables, and $\eta$ is the learning rate. The Gaussian noise of generative model, $N(\mathbf{b}, \boldsymbol{\tau})$ is updated by

$$\boldsymbol{\tau}'_G = \alpha\boldsymbol{\tau}_G, \quad (23)$$

$$\mathbf{b}_G \propto \eta(\mathbf{x_c} - \mathbf{G}\mathbf{z_c}). \quad (24)$$

The variances are decayed by a constant $\alpha$ during generation, and the biases are also updated by delta rule.

In the *sleep* phase of the algorithm, without reference to any input data, values for the latent variables are randomly chosen, and then *fantasy* input values are generated by the current parameters of the generative model. These fantasies supply an unbiased sample for the network's generative model of the data. Having produced a fantasy, the recognition weights are adjusted by a simple delta rule:

$$\mathbf{R} \propto \eta(\mathbf{z_k} - \mathbf{R}\mathbf{x_k})\mathbf{x_k}, \quad (25)$$

where $\mathbf{R}$ is the recognition weight vector, $\mathbf{z_k}$ is the $k$-th latent vector, and $\mathbf{x_k}$ is the $k$-th fantasy vector. The recognition Gaussian noise is updated similar as generative noise:

$$\boldsymbol{\tau}'_R = \alpha\boldsymbol{\tau}_R. \quad (26)$$

$$\mathbf{b}_R \propto \eta(\mathbf{z_k} - \mathbf{R}\mathbf{x_k}). \quad (27)$$

Using this wake-sleep algorithm and Helmholtz machine, we can capture the relationship between variables. Referred to Figure 3, sampling new populations are done by generative model of Helmholtz machine, and the relationship between

- Function 1: $f_1(\mathbf{x}) = \left\{10^{-5} + \sum_{i=1}^{d} |y_i|\right\}^{-1}; y_1 = x_1;\ y_i = y_{i-1} + x_i, i = 2, ..., d; -0.16 \le x_i \le 0.16$
- Function 2: $f_2(\mathbf{x}) = \mathbf{\sum_{i=1}^{d} \left((x_1 - x_i^2)^2 + (x_i - 1)^2\right); -10 \le x_i \le 10}$
- Function 3: $f_3(\mathbf{x}) = 1 + \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right); -600.0 \le x_i \le 600.0$
- Function 4: $f_4(\mathbf{x}) = 100 f_1(\mathbf{x}); -3.0 \le x_i \le 3.0$
- Function 5: $f_5(\mathbf{x}) = 100\left\{10^{-5} + \sum_{i=1}^{d} |y_i|\right\}^{-1}; y_1 = x_1;\ y_i = \sin(y_{i-1}) + x_i, i = 2, ..., d; -3.0 \le x_i \le 3.0$
- Sphere function: $f_s(\mathbf{x}) = \sum_{i=1}^{d} x_i^2; -20 \le x_i \le 20$
- Ackley function: $f_A(\mathbf{x}) = -20 \exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e; -20 \le x_i \le 20$
- Griewank function: $f_G(\mathbf{x}) = \frac{1}{4000d} \cdot \sum_{i=1}^{d} x^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1; -600 \le x_i \le 600$
- Rastrigin function: $f_{Ra}(\mathbf{x}) = 10d + \sum_{i=1}^{d}\left(x_i^2 - 10\cos(2\pi x_i)\right); -5.12 \le x_i \le 5.12$
- Rosenbrock function: $f_{Ro}(\mathbf{x}) = \sum_{i=2}^{d}\left(100 \cdot (x_i - x_{i-1}^2)^2 + (1 - x_{i-1})^2\right); -2.048 \le x_i \le 2.048$

Figure 4: Test functions used in our experiments

variables is represented in latent variables of Helmholtz machine.

By the way, one of difficult problems is how to decide the structure of Helmholtz machine, i.e. the number of latent variables and the number of layers. Small number of latent variables cannot learn the distribution well, but too large size will lead to overfitting and takes a lot of time for learning. For this problem, we use a constructive algorithm for structure learning in Helmholtz machine. A constructive algorithm starts with a small network and then adds additional hidden units and weights until a satisfactory solution is found, so it always searches for small networks to reduce the training time. But the remaining problem is to decide when to stop the addition of hidden units. We use a greedy approach to network construction; after adding a new hidden unit with small random initial values, we train the *whole* network by wake-sleep algorithm, not only the new hidden unit.

## 4 Experimental Results

Various functions have been tested to compare our methods with others. The test functions are shown in Figure 4. Test function 1, 4, and 5 are maximization functions, and other functions are minimization functions. All functions except Test function 2 and Rosenbrock function have their optimum at $\mathbf{x} = (\mathbf{0}, \ldots, \mathbf{0})$. Test function 2 and Rosenbrock function have their minimum at $\mathbf{x} = (\mathbf{1}, \ldots, \mathbf{1})$. The proposed methods can also be applied to the discrete space. However we focus on the continuous domain for demonstration purposes.

The parameters for the experiments are explained in Table 1 - Table 3. Table 1 shows the parameter settings of EDA with PPCA. For all test functions, termination criterion is based on the maximum number of evaluations. However, for last five benchmark functions like sphere function, Ackley function, and so on, EDAs with PPCA were stopped when the best fitness value is below $1.0 \times 10^{-13}$ or the variance $\sigma^2$ is below $1.0 \times 10^{-15}$. Table 2 describes the parameters for EDA with Helmholtz machine. Since we use constructive approach for Helmholtz machine construction, we initialize the Helmholtz machine with one latent variable. For test function 1 - 5, Helmholtz machine with 3 layers was used. The uppermost hidden layer has only one latent variable to capture the correlation of latent variables of first hidden layer. In constructive phase, latent variable of first hidden layer is added. For remaining test functions, two layer Helmholtz machine was used. Our latent variable model approaches use the elitist strategy. The parameters for real coded genetic algorithm is explained in Table 3. Real coded genetic algorithm use real vectors as a chromosome, and also apply the elitist strategy.

Table 4 displays the results for test functions 1, 2 and 3. Results obtained by all other methods except PPCA and Helmholtz machine are taken from [12] where Gaussian networks are used as the probabilistic model. One hundred ex-

Table 1: Characteristics of the functions and parameter settings for PPCA

| Function | Func 1 | Func 2 | Func 3 | Func 4 | Func 5 | Sphere | Ackley | Griewank | Rastrigin | Rosenbrock |
|---|---|---|---|---|---|---|---|---|---|---|
| Type | Max | Min | Min | Max | Max | Min | Min | Min | Min | Min |
| Optimum | $10^5$ | 0 | 0 | $10^7$ | $10^7$ | 0 | 0 | 0 | 0 | 0 |
| Dimension | 10 | | | 100 | | 50, 100 | | | | |
| # Max Eval | 300,000 | | | 200,000 | | 1,000,000 | | | | |
| Pop_Size | 10,000 | 2,000 | 750 | 2,000 | 2,000 | 200 | 200 | 5,000 | 5,000 | 200 |
| Sel_Rate | 0.01 | 0.1 | 0.2 | 0.1 | 0.2 | 0.5 | 0.5 | 0.1 | 0.1 | 0.5 |
| # Latent | 5 | 1 | 1 | 10 | 20 | 1 | 1 | 1 | 1 | 1 |

Table 2: Parameters for Helmholtz machine

| Parameters | Values |
|---|---|
| Maximum Generation | 100 |
| Population Size | 1000 |
| Learning Rate | 0.001 |
| Noise Decay Rate | 0.999 |
| Learning Iteration | 1000 |
| # Initial Latent Variables | 1 |
| Selection Rate | 0.5 |

Table 3: Parameters for real coded genetic algorithm

| Parameters | Values |
|---|---|
| Maximum Generation | $10^4$ |
| Population Size | 2000 |
| Crossover Rate | 0.9 |
| Mutation Rate | 0.01 |

Table 4: Mean values averaged on 100 runs for the test function 1, 2, and 3.

| | $\mathrm{UMDA}_c^G$ | $\mathrm{MIMIC}_c^G$ | $\mathrm{EGNA}_{ee}$ | $\mathrm{EGNA}_{BGe}$ | ES | PPCA | HM |
|---|---|---|---|---|---|---|---|
| Test 1 | 53460 | 58775 | 100000 | 100000 | 5910 | 94063.01 | 76646.08 |
| Test 2 | 0.13754 | 0.13397 | 0.09914 | 0.0250 | 0 | $3.68\times10^{-9}$ | $1.64\times10^{-8}$ |
| Test 3 | 0.011076 | 0.007794 | 0.008175 | 0.012605 | 0.034477 | $2.22\times10^{-18}$ | $7.90\times10^{-9}$ |

Table 5: Mean fitness values averaged on 20 runs for the test function 4 and 5 with the standard deviation.

| | (10+50)-ES | $\mathrm{PBIL}_c$ | IDEA | | PPCA | | HM |
|---|---|---|---|---|---|---|---|
| | Mean ± Stdev | Mean ± Stdev | Mean | RT | Mean ± Stdev | RT | Mean ± Stdev |
| Test 4 | $2.91 \pm 0.45$ | $4.76 \pm 0.78$ | 7.50 | 44.32 | $7.83 \pm 0.75$ | 6.91 | $366221.55 \pm 536372.64$ |
| Test 5 | $7.56 \pm 1.52$ | $11.18 \pm 1.36$ | 27.73 | 4.95 | $18.23 \pm 1.54$ | 2.34 | $237406.62 \pm 294739.05$ |

Table 6: Mean fitness values and standard deviations averaged on 20 runs for the test functions: 'S' is Sphere function, 'A' is Ackley function, 'G' is Griewank function, 'Ra' is Rastrigin function, and 'Ro' is Rosenbrock function.

| | | GA | | PPCA | | HM | |
|---|---|---|---|---|---|---|---|
| | N | Mean ± Stdev | # Eval | Mean ± Stdev | # Eval | Mean ± Stdev | # Eval |
| S | 50 | $17.43 \pm 1.17$ | 19,930,400 | $8.73\times10^{-14} \pm 6.89\times10^{-15}$ | 41,880 | $2.31\times10^{-5} \pm 2.28\times10^{-5}$ | 15,700 |
| | 100 | $20.37 \pm 1.00$ | 19,974,800 | $9.39\times10^{-14} \pm 5.28\times10^{-15}$ | 62,780 | $1.75\times10^{-5} \pm 1.92^{-5}$ | 20,200 |
| A | 50 | $1863.00 \pm 132.02$ | 19,919,800 | $1.01\times10^{-7} \pm 3.78\times10^{-9}$ | 43,780 | $1.48\times10^{-3} \pm 7.61\times10^{-4}$ | 16,400 |
| | 100 | $4549.99 \pm 280.06$ | 19,905,000 | $1.14\times10^{-7} \pm 3.25\times10^{-9}$ | 64,100 | $8.61\times10^{-4} \pm 4.90\times10^{-4}$ | 21,800 |
| G | 50 | $18.08 \pm 0.60$ | 19,931,000 | $5.10\times10^{-10} \pm 5.22\times10^{-10}$ | $10^6$ | $1.42\times10^{-6} \pm 2.44\times10^{-6}$ | 28,100 |
| | 100 | $20.81 \pm 0.90$ | 19,978,800 | $6.05\times10^{-7} \pm 3.71\times10^{-7}$ | $10^6$ | $2.70\times10^{-6} \pm 6.72\times10^{-6}$ | 28,800 |
| Ra | 50 | $221.72 \pm 21.57$ | 19,459,800 | $5.50\times10^{-12} \pm 5.87\times10^{-13}$ | 682,000 | $2.32\times10^{-3} \pm 2.65\times10^{-3}$ | 17,800 |
| | 100 | $447.28 \pm 22.70$ | 19,792,800 | $1.20\times10^{-11} \pm 9.96\times10^{-13}$ | 988,000 | $3.73\times10^{-4} \pm 7.33\times10^{-4}$ | 30,400 |
| Ro | 50 | $3941.472 \pm 431.93$ | 19,459,800 | $47.94 \pm 0.60$ | 255,240 | $49.23 \pm 0.47$ | 13,100 |
| | 100 | $8909.16 \pm 838.65$ | 18,958,600 | $97.24 \pm 0.76$ | 286,300 | $98.46 \pm 0.41$ | 20,200 |

periments for each function and algorithm were carried out. As shown in the Table 4, proposed methods show highly competitive results. Although EDA with latent variable models are slightly worse than EGNA for test function 1, they significantly outperform for other functions[1].

Table 5 summarizes the best fitness values averaged over 20 runs with standard deviations for the test functions 4 and 5. Here, the earlier reported results came from [17] and [4]. PPCA have better performances than those of the standard ES and the continuous version of PBIL. This implies that PPCA can detect the high order relations between variables by using the latent variables (or just one variable). The performance of PPCA is better than that of IDEA for the test function 4, although it is worse for the function 5. However, note that the relative time (RT) spent by PPCA is far smaller than that of IDEA for all functions. RT was introduced from [4] as a CPU independent fair running time comparison metric. Therefore we can expect that EDAs with PPCA find the nearly optimum solutions more efficiently than other EDAs. And EDA with Helmholtz machine significantly outperforms in test function 4 and 5. Especially, the major differences of test function 1 and 4 are the range of input variable and dimension of input variables. These difficulties cause poor results to other algorithms. But Helmholtz machine can find the solutions similar to test function 1. We speculate that Helmholtz machine can capture better correlation structure than other models such as PPCA and IDEA for more complex problems.

Table 6 shows the experimental results of other famous benchmark test functions. In all cases, latent variable models can find better solutions that simple genetic algorithms. In our approaches, PPCA has better performances that Helmholtz machines while PPCA requires more function evaluations to reach the optimal points. However, total running time of EDA with PPCA is generally less than that of Helmholtz machine. But if we optimize the parameters of Helmholtz machine for each function, we can get the better results with smaller running time.

## 5 Conclusions

We proposed two latent variable models, probabilistic principle component analysis (PPCA) and Helmholtz machine (HM), for the probability model of estimation of distribution algorithm (EDA). We tested the performance of the proposed methods on several continuous function optimization problems. Our empirical results show that EDAs with latent variable models are superior to simple real-coded genetic algorithm and other continuous estimation of distribution algorithms. Especially, PPCA-based EDA can detect the multivariate interactions with less total running time than other EDA methods. And EDA with Helmholtz machine can find good solutions in the case that the problems have complex correlations and a high dimension.

[1]We cannot find the criterion of zero for ES with test function 2. But we think our results are close to zero.

## Acknowledgments

This research was supported in part by the Korea Ministry of Science and Technology through KISTEP under Grant BR-2-1-G-06 and by the Ministry of Education under the BK21-IT Program.

## Bibliography

[1] S. Baluja and R. Caruana, "Removing the genetics from the standard genetic algorithm," *Proceedings of the 12th International Conference on Machine Learning*, pp. 38-46, Morgan Kaufmann, 1995.

[2] S. Baluja, and S. Davies, "Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space," *Proceedings of the 14th International Conference on Machine Learning*, pp. 30-38, Morgan Kaufmann, 1997.

[3] C. M. Bishop, "Latent Variable Models," *Learning in Graphical Models*, pp. 371-404, The MIT Press, 1999.

[4] P. A. N. Bosman and D. Thierens, "Expanding from Discrete to Continuous Estimation of Distribution Algorithms: The IDEA," *Parallel Problem Solving from Nature VI*, LNCS 1917, pp. 767-776, Springer, 2000.

[5] D.-Y. Cho and B.-T. Zhang, "Continuous Estimation of Distribution Algorithms with Probabilistic Principal Component Analysis," *The 2001 Congress on Evolutionary Computation* (CEC2001), Seoul, Korea, 2001 (submitted).

[6] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel, "The Helmholtz machine," *Neural Computation*, vol. 7, no. 5, pp. 889-904, 1995.

[7] J. S. De Bonet, C. L. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," *Advances in Neural Information Processing Systems*, vol. 9, pp. 424-430, The MIT Press, 1997.

[8] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, pp. 523-528, 1998.

[9] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The wake-sleep algorithm for unsupervised neural networks," *Science*, vol. 268, pp. 1158-1160, 1995.

[10] I. T. Jolliffe, *Principal Component Analysis*, Springer, 1986.

[11] P. Larrañga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Optimization by learning and simulation of Bayesian and Gaussian networks," *Technical Report EHU-KZAA-IK-4/99*, http://www.sc.edu.es/ccwbayes/postscript/kzaa-ik-04-99.ps, 1999.

[12] P. Larrañaga, R. Etxeberria, J. A. Lozano, and M. Peña, "Optimization in continuous domains by learning and simulation of Gaussian networks," *Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program*, pp. 201-204, 2000.

[13] Mühlenbein, H. and Paaß, G., "From recombination of genes to the estimation of distributions I: Binary parameters," *Parallel Problem Solving from Nature IV*, LNCS 1141, pp. 178-187, Springer, 1996.

[14] H. Mühlenbein and T. Mahnig "FDA - A scalable evolutionary algorithm for the optimization of additively decomposed functions," *Evolutionary Computation*, vol. 7, no. 4, pp. 353-376, 1999.

[15] M. Pelikan and H. Mühlenbein, "The bivariate marginal distribution algorithm," *Advances in Soft Computing - Engineering Design and Manufacturing*, pp. 521-535, Springer, 1999.

[16] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "Linkage Problem, Distribution Estimation, and Bayesian Networks," *Evolutionary Computation*, vol. 8, no. 3, pp. 311-340, 2000.

[17] M. Sebag and A. Ducoulombier, "Extending population-based incremental learning to continuous search spaces," *Parallel Problem Solving from Nature V*, LNCS 1498, pp. 418-427, Springer, 1998.

[18] S.-Y. Shin and B.-T. Zhang, "Bayesian Evolutionary Algorithms for Continuous Function Optimization," *The 2001 Congress on Evolutionary Computation* (CEC2001), Seoul, Korea, 2001 (submitted).

[19] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B*, vol. 61, no. 3, pp. 611-622, 1999.

[20] B.-T. Zhang and S.-Y. Shin, "Bayesian Evolutionary Optimization Using Helmholtz Machines," *Parallel Problem Solving from Nature VI*, LNCS 1917, pp. 827-836, Springer, 2000.