Discriminative Feature Extraction with Deep Neural Networks

André Stuhlsatz, Jens Lippel and Thomas Zielke

Abstract— **We propose a framework for optimizing Deep Neural Networks (DNN) with the objective of learning low-dimensional discriminative features from high-dimensional complex patterns.**

In a two-stage process that effectively implements a Nonlinear Discriminant Analysis (NDA), we first pretrain a DNN using stochastic optimization, partly supervised and unsupervised. This stage involves layer-wise training and stacking of single Restricted Boltzmann Machines (RBM).

The second stage performs fine-tuning of the DNN using a modified back-propagation algorithm that directly optimizes a Fisher criterion in the feature space spanned by the units of the last hidden-layer of the network.

Our experimental results show that the features learned by a DNN using the proposed framework greatly facilitate classification, even when the discriminative features constitute a substantial dimension reduction.

I. Introduction

Deep Neural Networks (DNN) are Multilayer Neural Networks (MLNN) with more than one hidden-layer and thousands, often millions of parameters. None of the well known training algorithms for MLNNs are capable of finding good solutions in such high-dimensional parameter spaces, except for chance. On the other hand, the high flexibility of DNNs enables the learning of discriminative features of low dimension from high-dimensional complex data, also known as Nonlinear Discriminant Analysis (NDA). In [1], [2], [3], [4], [5], [6] NDA with MLNNs is theoretically analyzed. However, empirical results were only reported using MLNNs with no more than two hidden-layers and few parameters [7], [8], because training of DNNs is infeasible due to overfitting and the existence of many poor local optima in the parameter space.

In this paper, we present a framework for pre-optimizing DNNs regarding NDA, and a subsequent fine-tuning using back-propagation with a Fisher discriminant criterion as objective function. We justify our approach by classification experiments on various real world datasets using linear classifiers as well as sophisticated nonlinear ones, like Support Vector Machines (SVM). Our results show that highly discriminative low-dimensional features can be learned using the proposed framework without particular assumptions about the underlying data distribution.

André Stuhlsatz is with the Department of Mechanical and Process Engineering, University of Applied Sciences, Duesseldorf, Germany. He is also a member of the Cognitive Systems Group at the Otto-von-Guericke University, Magdeburg, Germany. (email: andre.stuhlsatz@fh-duesseldorf.de)

Jens Lippel is post-graduate student in the Department of Mechanical and Process Engineering, University of Applied Sciences, Duesseldorf, Germany (email: jens.lippel@fh-duesseldorf.de).

Thomas Zielke is with the Department of Mechanical and Process Engineering, University of Applied Sciences, Duesseldorf, Germany (email: thomas.zielke@fh-duesseldorf.de).

II. Pre-Optimizing Deep Neural Networks

Randomly initializing free parameters and restarting until thrown into the basin of attraction of a good solution, where gradient descent works well, is practically useless for optimizing DNNs. To find a good set of initial parameters for DNNs used as autoencoder, in [9] a stochastic pre-optimization was proposed.

The basic idea of pre-optimizing DNNs for the autoencoder task is to stack layer-wise trained Restricted Boltzmann Machines (RBM) [10], and then to use the RBM's stochastically learned parameters as initial configuration for the full network (Fig. 1). The initial configuration is then hopefully near a good solution with respect to the objective function to be optimized.

For the purpose of layer-wise training [11], the first RBM of a stack is trained. Then the activations of the *hidden units* $\boldsymbol{h}^l$ are clamped as inputs $\boldsymbol{v}^{(l+1)}$ at the *visual units* for training the next RBM in the stack, and so forth until the top layer is trained (Fig. 1). It has been shown [12], that layer-wise training improves a variational bound of the logarithmic prior of the data, $\log p(\boldsymbol{v}^1)$, when layers with increasing number of states are added and the higher level weights are appropriately initialized. However, empirical investigations indicate that layer-wise training of RBMs yields very good results even when violating these assumptions, e.g. [11].

A. Training a Single Restricted Boltzmann Machine

Let be $\boldsymbol{\Theta} := (\mathbf{W}, \boldsymbol{b})$ with $\mathbf{W} \in \mathbb{R}^{N_v \times N_h}, \boldsymbol{b} := ((\boldsymbol{b}^v)^T, (\boldsymbol{b}^h)^T)^T, \boldsymbol{b}^v \in \mathbb{R}^{N_v}, \boldsymbol{b}^h \in \mathbb{R}^{N_h}$ the network parameters of a RBM. Then the equilibrium distribution of the visual and hidden states $\boldsymbol{s} := (\boldsymbol{v}^T, \boldsymbol{h}^T)^T, \boldsymbol{v} \in \{0,1\}^{N_v}, \boldsymbol{h} \in \{0,1\}^{N_h}$ of a RBM is modeled as follows:

$$P^\infty(\boldsymbol{s}; \boldsymbol{\Theta}) = \frac{1}{Z(\boldsymbol{\Theta})} \exp\left(-H(\boldsymbol{s}; \boldsymbol{\Theta})\right) \tag{1}$$

$$Z(\boldsymbol{\Theta}) := \sum_{\boldsymbol{s}} \exp\left(-H(\boldsymbol{s}; \boldsymbol{\Theta})\right) \tag{2}$$

with *energy function*

$$H(\boldsymbol{s}; \boldsymbol{\Theta}) := -\boldsymbol{v}^T \mathbf{W} \boldsymbol{h} - \boldsymbol{b}^T \boldsymbol{s}. \tag{3}$$

Training a RBM is performed by optimizing the parameters $\boldsymbol{\Theta}$ in such a way that the likelihood

$$P^\infty(\boldsymbol{v}; \boldsymbol{\Theta}) = \sum_{\boldsymbol{h}} P^\infty(\boldsymbol{s}; \boldsymbol{\Theta}) \tag{4}$$

is maximized [10]. Technically speaking, one minimizes the Kullback-Leibler divergence,

$$d(P^0 || P^\infty; \boldsymbol{\Theta}) := \sum_{\boldsymbol{v}} P^0(\boldsymbol{v}) \log\left(\frac{P^0(\boldsymbol{v})}{P^\infty(\boldsymbol{v}; \boldsymbol{\Theta})}\right), \tag{5}$$

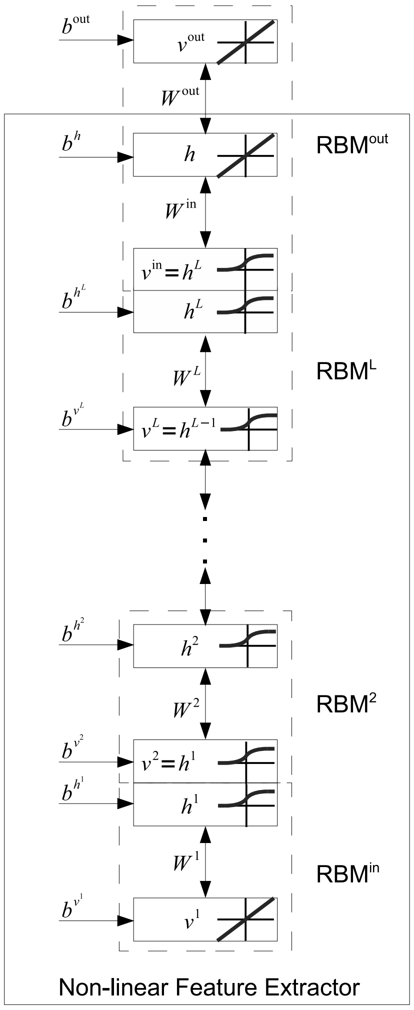


Fig. 1. Multilayer RBM stack composing a nonlinear feature extractor.

that is usually done by gradient descent with parameter updates according to

$$\Theta_{ij}^{(k+1)} := \Theta_{ij}^{(k)} - \eta \cdot \frac{\partial d(P^0||P^\infty;\mathbf{\Theta}^{(k)})}{\partial \Theta_{ij}^{(k)}} \tag{6}$$

given an appropriate learning rate η.

The partial derivatives are straightforward derived:

$$\frac{\partial d(P^0||P^\infty;\mathbf{\Theta})}{\partial \Theta_{ij}} = \left\langle \left\langle \frac{\partial H(\boldsymbol{s};\mathbf{\Theta})}{\partial \Theta_{ij}} \right\rangle_{P^\infty(\boldsymbol{h}|\boldsymbol{v};\mathbf{\Theta})} \right\rangle_{P^0(\boldsymbol{v})} - \left\langle \frac{\partial H(\boldsymbol{s};\mathbf{\Theta})}{\partial \Theta_{ij}} \right\rangle_{P^\infty(\boldsymbol{s};\mathbf{\Theta})}, \tag{7}$$

where $\langle\cdot\rangle_{P(\cdot)}$ denotes expectations with respect to a distribution P. Practically, these expectations are estimated using samples simulated from the distributions P^0 and P^∞ running a Markov chain to equilibrium via *Gibbs sampling* [13].

B. Contrastive Divergence (CD)

Training a RBM is prohibitively slow, because running a Markov chain to equilibrium requires a large number of sampling steps and the resulting stochastic derivatives (7) are usually very noisy.

Making RBM training practical, a heuristic was proposed in [14] in which the difference of two Kullback-Leibler distances,

$$CD_n(\mathbf{\Theta}) := d(P^0||P^\infty;\mathbf{\Theta}) - d(P^n||P^\infty;\mathbf{\Theta}), \tag{8}$$

is minimized instead of minimizing (5). Here, P^n denotes the distribution of visual states $\boldsymbol{v}$, in case the Markov chain for actually sampling from P^∞ is already stopped after a small number $n > 0$ of steps.

In analogy to Eq. (7), the approximated[1] partial derivatives of CD_n read as:

$$\frac{\partial CD_n(\mathbf{\Theta})}{\partial \Theta_{ij}} \approx \left\langle \left\langle \frac{\partial H(\boldsymbol{s};\mathbf{\Theta})}{\partial \Theta_{ij}} \right\rangle_{P^\infty(\boldsymbol{h}|\boldsymbol{v};\mathbf{\Theta})} \right\rangle_{P^0(\boldsymbol{v})} - \left\langle \frac{\partial H(\boldsymbol{s};\mathbf{\Theta})}{\partial \Theta_{ij}} \right\rangle_{P^n(\boldsymbol{s};\mathbf{\Theta})}. \tag{9}$$

The idea behind the CD_n-heuristics is that the distribution P^n is closer to the equilibrium distribution than P^0. Thus, $d(P^0||P^\infty;\mathbf{\Theta})$ is greater than $d(P^n||P^\infty;\mathbf{\Theta})$ unless P^n equals P^0. If all transitions of the Markov chain have nonzero probability, then $P^n = P^0$ yields $P^\infty = P^0$. Therefore, it holds $CD_n(\mathbf{\Theta}) = 0$ if the model is perfectly optimized.

Moreover, it is actually possible to simulate from P^1 using a 1-step Gibbs sampler for first simulate all hidden states according to

$$P^\infty(h_j = 1|\boldsymbol{v};\mathbf{\Theta}) = \frac{1}{1+\exp\left(-\sum_{i=1}^{N_v} W_{ij}v_i - b_j^h\right)} \tag{10}$$

and than simulate all visual states according to

$$P^\infty(v_i = 1|\boldsymbol{h};\mathbf{\Theta}) = \frac{1}{1+\exp\left(-\sum_{j=1}^{N_h} W_{ij}h_j - b_i^v\right)}. \tag{11}$$

III. Deep Neural Networks as Discriminative Feature Extractors

In Section II, we summarized the building blocks for pre-optimizing DNNs. In the following, we want to adapt these building blocks aiming at an initialization of DNNs for a subsequent fine-tuning as nonlinear discriminative feature extractors with respect to a discriminant criterion.

A. Nonlinear Discriminant Analysis

Consider a *target coding scheme* $\mathbf{\Lambda} := (\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_c) \in \mathbb{R}^{c\times c}$ for a c-class problem where code vectors $\boldsymbol{\lambda}_i \in \mathbb{R}^c$ uniquely represent a category ω_i. Further, let be $\boldsymbol{t}_n \in \{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_c\}$ a *target code* associated to the known category $\omega(\boldsymbol{x}_n)$ of a data point $\boldsymbol{x}_n \in \mathbb{R}^d$ and let be $\boldsymbol{v}^{out}(\boldsymbol{x}_n) \in \mathbb{R}^c$ the state of the output units.

[1]The remark about an approximation is due to the lack of the negligible term $\frac{\partial P^n}{\partial \Theta_{ij}}\frac{\partial d(P^n||P^\infty;\mathbf{\Theta})}{\partial P^n}$ compared to the exact derivative [14].

As is well-known, the Mean Squared Error (MSE) at the network outputs $\boldsymbol{v}^{out}$ w.r.t. a training sample $\mathcal{O}_N := \{(\boldsymbol{t}_1, \boldsymbol{x}_1), \ldots, (\boldsymbol{t}_N, \boldsymbol{x}_N)\}$ reaches asymptotically with probability one the true risk $R := \int_{\mathbb{R}^d} \|\boldsymbol{r}(\boldsymbol{x}) - \boldsymbol{v}^{out}(\boldsymbol{x})\|_2^2 \; p(\boldsymbol{x}) \; d\boldsymbol{x} + c$, i.e. it holds

$$\lim_{\substack{N \longrightarrow \infty \\ p}} \frac{1}{N} \sum_{n=1}^{N} \left\|\boldsymbol{t}_n - \boldsymbol{v}^{out}(\boldsymbol{x}_n)\right\|_2^2 = R, \tag{12}$$

where $\boldsymbol{r}(\boldsymbol{x}) := \boldsymbol{\Lambda} \cdot (P(\omega_1|\boldsymbol{x}), \ldots, P(\omega_c|\boldsymbol{x}))^T$.

Consequently, if a MLNN minimizes the MSE in the infinite sample case, then the network outputs are optimized to approximate the Bayes-risk vector $\boldsymbol{r}(\boldsymbol{x})$. In particular, if $\boldsymbol{\Lambda}$ equals the identity matrix the network outputs approximate the class posteriori probabilities $P(\omega_i|\boldsymbol{x})$. In [6] it has been proved, that minimizing the true risk R using a linear output network, i.e.

$$\boldsymbol{v}^{out}(\boldsymbol{x}) = \mathbf{W}\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{b} \tag{13}$$

with last hidden layer outputs $\boldsymbol{h}(\boldsymbol{x}) \in \mathbb{R}^m$, is equivalent to a maximization of a discriminant criterion $Q_{\boldsymbol{h}}$ evaluated in the m-dimensional space spanned by the last hidden layer outputs $\boldsymbol{h}(\boldsymbol{x})$. In particular for a finite number N of observations $\mathcal{O}_N$, the use of the target coding scheme

$$\boldsymbol{\Lambda}_{j,i} := \begin{cases} \frac{\sqrt{N_i/N}}{P(\omega_i)} & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}, \tag{14}$$

where N_i/N is the fraction of examples of class ω_i, approximates the well-known Fisher discriminant criterion

$$Q_{\boldsymbol{h}} = \mathit{trace}\left\{\mathbf{S}_T^{-1}\mathbf{S}_B\right\} \tag{15}$$

with the common *total scatter matrix* $\mathbf{S}_T$ and common *between-class scatter* matrix $\mathbf{S}_B$.

This result suggests to adapt the pre-optimization of DNNs for the learning of target codes yielding a nonlinear discriminant analysis.

B. Pre-Optimization Adapted to Fisher Objective

Minimizing the MSE between target codes (14) and output values of a DNN with linear output-layer yields (asymptotically) a maximization of a Fisher discriminant criterion (15) in the space spanned by the last hidden-layer outputs.

Keeping this result in mind, we carry out the adaptation of the pre-optimization related to (15) in two steps:

First, in Section III-B.1, we extend the output RBM with extra visual output units (Fig. 1). The output states $\boldsymbol{v}^{out}$ are modeled continuously, Gaussian-distributed to facilitate a regression of the desired target codes (14). Likewise, we modeled the hidden states $\boldsymbol{h}$ continuously and Gaussian-distributed in order to facilitate also a continuous feature extraction at the hidden-layer.

Second, in Section III-B.2, we adapt the CD-heuristic for the learning of a mapping from input states $\boldsymbol{v}^{in}$ to output states $\boldsymbol{v}^{out}$, i.e. we maximize the likelihood w.r.t a unknown conditional density $p^0(\boldsymbol{v}^{out}|\boldsymbol{v}^{in})$ instead of $P^0(\boldsymbol{v})$.

1) Continuous and Gaussian-distributed States: Continuously Gaussian-distributed states $\boldsymbol{s} := (\boldsymbol{v}^{in}, \boldsymbol{v}^{out}, \boldsymbol{h}) \in \{0,1\}^{N_{\boldsymbol{v}^{in}}} \times \mathbb{R}^{N_{\boldsymbol{v}^{out}}} \times \mathbb{R}^{N_h}$ can be modeled by modifying the energy functional (3) parameterized by $\boldsymbol{\Theta} := (\mathbf{W}^{in}, \mathbf{W}^{out}, \boldsymbol{b}^h, \boldsymbol{b}^{out}), \mathbf{W}^{in} \in \mathbb{R}^{N_{\boldsymbol{v}^{in}} \times N_h}, \mathbf{W}^{out} \in \mathbb{R}^{N_{\boldsymbol{v}^{out}} \times N_h}, \boldsymbol{b}^h \in \mathbb{R}^{N_h}, \boldsymbol{b}^{out} \in \mathbb{R}^{N_{\boldsymbol{v}^{out}}}$, with a quadratic energy term (cf. [15]):

$$\begin{aligned} H(\boldsymbol{s}; \boldsymbol{\Theta}) :=& \frac{1}{2}\left(\boldsymbol{v}^{out} - \boldsymbol{b}^{out}\right)^T (\boldsymbol{\Sigma}^{out})^{-1} \left(\boldsymbol{v}^{out} - \boldsymbol{b}^{out}\right) \\ &-(\boldsymbol{v}^{out})^T(\boldsymbol{\Sigma}^{out})^{-\frac{1}{2}}\mathbf{W}^{out}\boldsymbol{h} - (\boldsymbol{v}^{in})^T\mathbf{W}^{in}(\boldsymbol{\Sigma}^h)^{-\frac{1}{2}}\boldsymbol{h} \\ &+\frac{1}{2}\left(\boldsymbol{h} - \boldsymbol{b}^h\right)^T (\boldsymbol{\Sigma}^h)^{-1}\left(\boldsymbol{h} - \boldsymbol{b}^h\right) \end{aligned} \tag{16}$$

with diagonal covariance matrices

$$\boldsymbol{\Sigma}^{out} := \mathit{diag}\left((\sigma_1^{out})^2, \ldots, (\sigma_{N_{\boldsymbol{v}^{out}}}^{out})^2\right) \tag{17}$$

$$\boldsymbol{\Sigma}^h := \mathit{diag}\left((\sigma_1^h)^2, \ldots, (\sigma_{N_h}^h)^2\right). \tag{18}$$

Substituting the functional (16) into the equilibrium distribution (1), it is easy to see that

$$p^\infty(\boldsymbol{v}^{out}|\boldsymbol{v}^{in}, \boldsymbol{h}; \boldsymbol{\Theta}) = \prod_{k=1}^{N_{\boldsymbol{v}^{out}}} p^\infty(v_k^{out}|\boldsymbol{h}; \boldsymbol{\Theta}) \tag{19}$$

with Gaussian

$$p^\infty(v_k^{out}|\boldsymbol{h}; \boldsymbol{\Theta}) = \frac{1}{\sigma_k^{out}\sqrt{2\pi}} \exp\left(-\frac{(v_k^{out} - \mu_k^{out})^2}{2(\sigma_k^{out})^2}\right) \tag{20}$$

and mean $\mu_k^{out} := b_k^{out} + \sigma_k^{out}\mathbf{W}_{k(\cdot)}^{out}\boldsymbol{h}$.

Note, $\mathbf{W}_{i(\cdot)}$ denotes the ith row of the matrix $\mathbf{W}$, respectively $\mathbf{W}_{(\cdot)j}$ denotes the $j$the column.

Similarly, one obtains

$$p^\infty(\boldsymbol{h}|\boldsymbol{v}^{out}, \boldsymbol{v}^{in}; \boldsymbol{\Theta}) = \prod_{j=1}^{N_h} p^\infty(h_j|\boldsymbol{v}^{out}, \boldsymbol{v}^{in}; \boldsymbol{\Theta}) \tag{21}$$

with Gaussian

$$p^\infty(h_j|\boldsymbol{v}^{in}, \boldsymbol{v}^{out}; \boldsymbol{\Theta}) = \frac{1}{\sigma_j^h\sqrt{2\pi}} \exp\left(-\frac{\left(h_j - \mu_j^h\right)^2}{2(\sigma_j^h)^2}\right) \tag{22}$$

and mean

$$\mu_j^h := b_j^h + \sigma_j^h(\boldsymbol{v}^{in})^T\mathbf{W}_{(\cdot)j}^{in} + (\sigma_j^h)^2(\boldsymbol{v}^{out})^T(\boldsymbol{\Sigma}^{out})^{-\frac{1}{2}}\mathbf{W}_{(\cdot)j}^{out}.$$

2) CD-Learning of Input-Output Associations (CD^{IO}): For learning input-output associations with the output RBM efficiently, we modified the CD-heuristic (8) to be

$$CD_n^{IO}(\boldsymbol{\Theta}) := d(p^0||p^\infty; \boldsymbol{\Theta}) - d(p^n||p^\infty; \boldsymbol{\Theta}) \tag{23}$$

with

$$\begin{aligned} d(p^n||p^\infty; \boldsymbol{\Theta}) :=& \sum_{\boldsymbol{v}^{in}} \int_{\mathbb{R}^{N_{\boldsymbol{v}^{out}}}} P^0(\boldsymbol{v}^{in})p^n(\boldsymbol{v}^{out}|\boldsymbol{v}^{in}; \boldsymbol{\Theta}) \cdot \\ &\log\left(\frac{p^n(\boldsymbol{v}^{out}|\boldsymbol{v}^{in}; \boldsymbol{\Theta})}{p^\infty(\boldsymbol{v}^{out}|\boldsymbol{v}^{in}; \boldsymbol{\Theta})}\right) d\boldsymbol{v}^{out} \end{aligned} \tag{24}$$

and $p^0(\boldsymbol{v}^{out}|\boldsymbol{v}^{in}; \boldsymbol{\Theta}) := p^0(\boldsymbol{v}^{out}|\boldsymbol{v}^{in})$.

This modification yields the following approximated partial derivatives with respect to the parameters $\mathbf{\Theta}$:

$$\frac{\partial CD_n^{IO}(\mathbf{\Theta})}{\partial \Theta_{ij}} \approx \tag{25}$$
$$\left\langle \left\langle \frac{\partial H(\boldsymbol{s};\mathbf{\Theta})}{\partial \Theta_{ij}} \right\rangle_{p^{\infty}(\boldsymbol{h}|\boldsymbol{v}^{in},\boldsymbol{v}^{out};\mathbf{\Theta})} \right\rangle_{p^{0}(\boldsymbol{v}^{in},\boldsymbol{v}^{out})}$$
$$- \left\langle \left\langle \frac{\partial H(\boldsymbol{s};\mathbf{\Theta})}{\partial \Theta_{ij}} \right\rangle_{p^{\infty}(\boldsymbol{h}|\boldsymbol{v}^{in},\boldsymbol{v}^{out};\mathbf{\Theta})} \right\rangle_{p^{n}(\boldsymbol{v}^{out}|\boldsymbol{v}^{in};\mathbf{\Theta})P^{0}(\boldsymbol{v}^{in})}$$

While the first expectation in (25) is readily available, the costly estimation of the second expectation with free varying visual output states, is substantially reduced using CD_1^{IO}:

Simulate $\boldsymbol{h}^0 \sim p^{\infty}(\boldsymbol{h}|\boldsymbol{v}^{in}(\boldsymbol{x}_n),\boldsymbol{t}_n;\mathbf{\Theta})$ (Eq. 22) with clamped target $\boldsymbol{t}_n$ and associated training input $\boldsymbol{v}^{in}(\boldsymbol{x}_n)$. Then, simulate a reconstruction $\boldsymbol{v}^{rec} \sim p^{\infty}(\boldsymbol{v}^{out}|\boldsymbol{h}^0;\mathbf{\Theta})$ (Eq. 20) and update the hidden states according to $p^{\infty}(\boldsymbol{h}|\boldsymbol{v}^{in}(\boldsymbol{x}_n),\boldsymbol{v}^{rec};\mathbf{\Theta})$. Compute the empirical expectations by repeating this procedure for all training samples.

Given a finite training sample $\mathcal{O}_N$, empirical distributions $P^0(\boldsymbol{v}^{in}) = \frac{1}{N}\sum_{n=1}^{N}\delta(\boldsymbol{v}^{in} - \boldsymbol{x}_n)$ and $p^0(\boldsymbol{v}^{out}|\boldsymbol{v}^{in}(\boldsymbol{x}_n)) = \delta(\boldsymbol{v}^{out} - \boldsymbol{t}_n)$ are assumed. Thus, if the density $p^1(\boldsymbol{v}^{out}|\boldsymbol{v}^{in}(\boldsymbol{x}_n);\mathbf{\Theta}^*)$ reflects sufficiently well the empirical density $p^0(\boldsymbol{v}^{out}|\boldsymbol{v}^{in}(\boldsymbol{x}_n))$ for all $1 \leq n \leq N$ at a minimizer $\mathbf{\Theta}^*$ of (23), then this implies also a minimum MSE between the desired target codes $\boldsymbol{t}_n$ and a linear output model $\boldsymbol{\mu}^{out} = \boldsymbol{b}^{out} + (\mathbf{\Sigma}^{out})^{\frac{1}{2}}\mathbf{W}^{out}\boldsymbol{h}$.

Therefore, in virtue of the results presented in Section III-A, our proposed approach enables a pre-optimization of the network parameters with respect to the Fisher discriminant criterion (15) if using the target coding scheme (14).

C. Building and Fine-Tuning the Nonlinear Feature Extractor

Aiming at a layer-wise pre-optimization of DNNs as described in Section II, we trained three types of RBMs in a greedy layer-wise manner (Fig. 1):

- an input RBM with Gaussian visual states for continuous input data and binary hidden states trained via CD_1-heuristics.
- a couple of RBMs with binary visual and hidden states building up $(L-1)$ inter-layers each trained via CD_1-heuristics.
- an output RBM with binary visual input states, Gaussian hidden states and Gaussian visual output states trained via CD_1^{IO}-heuristics.

After pre-optimization, the output units of the output RBM are discarded to obtain a DNN up to the last hidden-layer (Fig. 1). Maintaining the parameter set $\mathbf{\Theta} := (\boldsymbol{b}^{h^1},\ldots,\boldsymbol{b}^{h^L},\boldsymbol{b}^{h},\mathbf{W}^1,\ldots,\mathbf{W}^L,\mathbf{W}^{in})$, the DNN is now appropriately initialized for a nonlinear feature extraction with respect to a Fisher discriminant criterion (15).

The last step in optimizing the feature extractor is a fine-tuning of the initialized DNN. For this purpose we used a modified back-propagation algorithm that maximizes (15) in the feature space defined by the last hidden layer of the network.

IV. Experiments

In this section, the evaluation of the classification and generalization performance of the NDA features produced by our feature extraction framework is discussed.

For the experiments, we used three databases, namely MNIST [16], Statlog's Satimage (Landsat Satellite) [17] and the Letter Recognition Dataset [17].

At first, we tested our framework on the popular MNIST database of handwritten digits, because it is established as benchmark database in the community. As additional performance references, we chose the Satimage database, as a very small dataset compared to MNIST, and the Letter Recognition Dataset which has much more classes than MNIST.

In order to quantify the discriminative power of the NDA features, we combined the feature extractors with linear classifiers based on Mahalanobis-distance, K-Nearest-Neighbor (KNN) classifiers, linear Support Vector Machines (SVM) and nonlinear SVMs. These classifiers were trained on the NDA features obtained by the feature extractors from their respective training sets. For each classifier and database, we report the classification performance on the raw data as well as on the NDA features. Some results on the raw data have been taken from the literature.

Using MNIST, we also performed experiments for the evaluation of the generalization ability of the NDA features. Additionally, the stability of the pre-optimization algorithm has been explored through cross-validation on MNIST.

A. Classification Results on the MNIST Database

The MNIST database of handwritten digits [16] contains grayscale images of handwritten digits with a 28x28 pixel resolution. The full database consists of a total of 60,000 training and 10,000 test images

We randomly divided the 60,000 images of the training set into a smaller training set of 50,000 images and a validation set of 10,000 images. The validation set was used to select the best network topology based on the validation error using a linear classifier. Several network topologies were evaluated with a fixed number of units in the last hidden layer (feature extraction layer). We observed that three hidden layers with 1000-2000-1000 units performed best. In order to significantly reduce the dimensionality of the raw data we explored different numbers of last hidden layer units (NDA feature dimensions) ranging from 2 to 100 units. We found that on MNIST the best number is 10 feature dimensions, i.e. the number of classes. Using less than 10 feature dimensions more and more decreases the performance due to a loss of information, while more than 10 feature dimensions do not influence the performance noticeably. In order to get the best classification performance at the optimal dimensionality reduction, we fixed the network to a 784-1000-2000-1000-10 topology with weights and biases corresponding to the lowest validation error w.r.t. a linear classification.

TABLE I
MNIST Dataset Classification Error

Classifier	classification errors in %		
	784dim.-raw data	10dim.-NDA	
	Test	Valid.	Test
linear	12.0 [16]	1.37	**1.58**
KNN	2.83 [16]	—	**1.49**
SVM lin.	—	1.37	1.46
SVM rbf	**1.4** [16]	1.31	1.47

TABLE II
Satimage Dataset Classification Error

Classifier	classification errors in %					
	36dim.-raw data		6dim.-LDA		6dim.-NDA	
	Valid.	Test	Valid.	Test	Valid.	Test
linear	—	17.35	—	17.35	8.51	**9.80**
KNN	—	**9.35**	—	11.85	—	11.70
SVM lin.	15.4	16.85	16.09	19.20	7.82	**9.90**
SVM RBF	6.67	10.15	10.57	12.65	7.36	**10.05**

The classification results on both the validation and the test data are shown in Table I in comparison to results reported in [16] on the raw MNIST data (left column). The right column shows results using the NDA features. The test error of the linear classifier and the KNN is substantially reduced to 1.58% and 1.49% respectively on the NDA features. There is no significant performance gain when using a linear SVM or a nonlinear SVM compared to the linear classifier. This indicates that the NDA features explicitly represent the discriminative information contained in the raw data. These results are competitive with the best results obtained on the raw MNIST data without incorporating any prior knowledge.

However, using our NDA features it is possible to reach nearly the same high classification performance but with a simple linear classifier in a 10-dimensional feature space. The low dimensional representation of the NDA features is particular beneficial in applications with very high dimensional data and the requirement of very fast classification.

B. Classification Results on the Satimage Dataset

The Satimage [17] dataset contains 4,435 training and 2,000 test samples of 6 different classes. A sample consists of multispectral values of pixels in 3x3 neighborhoods in a satellite image. The categories are associated to the central pixel in each neighborhood. As a very small database compared to the MNIST database, we used it to assess the performance of the feature extraction in cases where only a small dataset is available. As in the MNIST experiments, we set the feature extraction network to the learned weights and biases that had produced the lowest validation error w.r.t. a linear classification.

In Table II test results are shown for different classifiers that were applied on the raw data, on LDA extracted features and the NDA features. The number of LDA features space dimensions was chosen to be equal to the number of NDA dimensions (6 dimensions).

TABLE III
Letter Recognition Dataset Classification Error

Classifier	classification errors in %			
	16dim.-raw data		26dim.-NDA	
	Valid.	Test	Valid.	Test
linear	—	31.18	3.40	**3.15**
KNN	—	4.80	—	**3.58**
SVM lin.	49.50	82.13	1.68	**2.85**
SVM rbf	3.08	**2.73**	1.68	3.08

The NDA features are always better classified than LDA features. An impressive high improvement of classification performance can be observed for the linear classifier (9.80% compared to 17.35%) and the linear SVM method (9.90% compared to 16.85%). Again, the test errors using NDA features differ only slightly for different classification methods.

C. Classification Results on the Letter Recognition Dataset

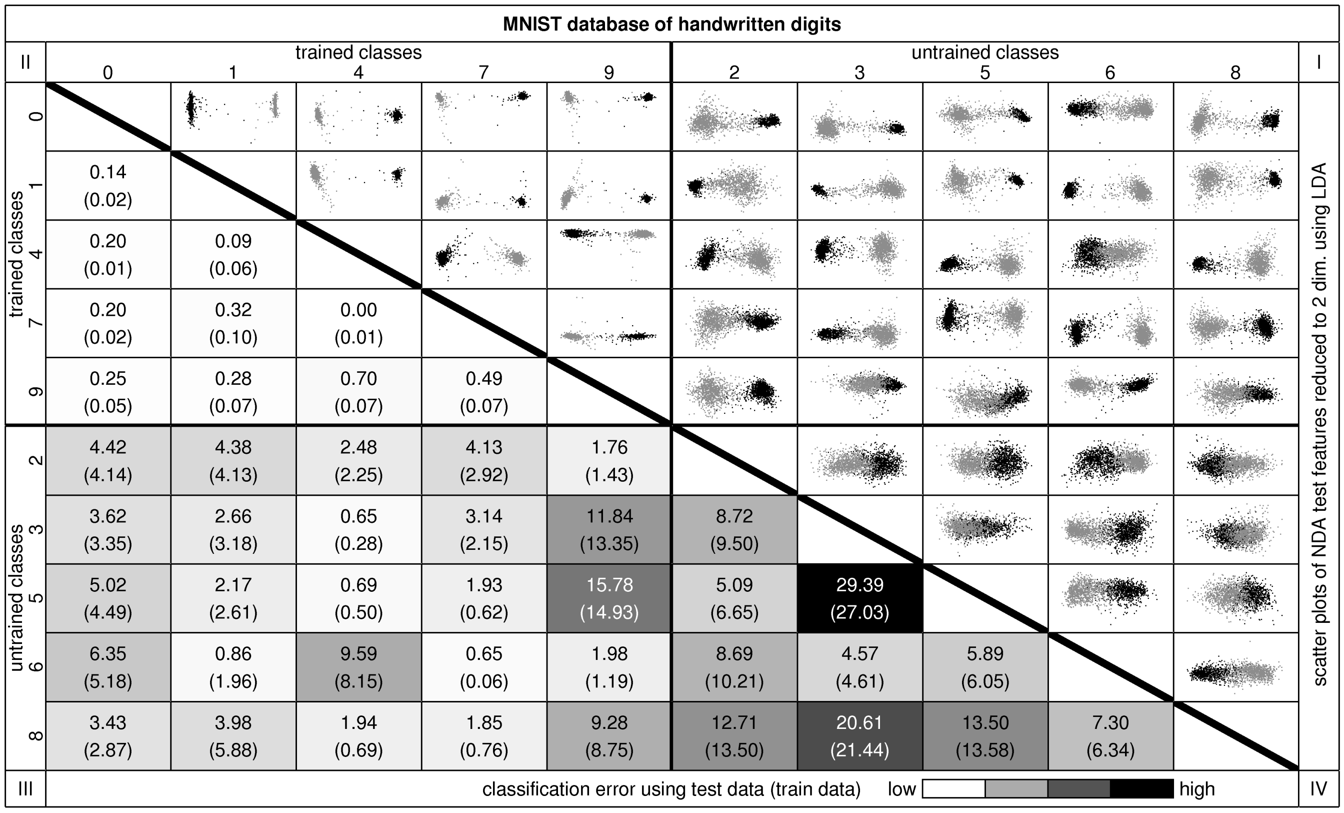
The Letter Recognition Dataset [17] contains 20,000 samples with 16 attributes describing the characteristics of randomly distorted images of the 26 capital letters of the English alphabet. The task is to classify the 26 different letters. We use the Letter Recognition Dataset as a third benchmark in order to study the behavior of our feature extractor on data where the number of classes is higher than the dimensionality of the data samples.

From the 20,000 available samples, we used the last 4,000 samples for testing. The remaining first 16,000 samples were divided into 12,000 samples for training and 4,000 samples for validation. Next, we sought the best network topology with respect to the validation error using a linear classifier. Fixing the feature extraction layer to 26 units, the network corresponding to the lowest validation error w.r.t. a linear classification was found to have the topology 16-250-125-75-26.

In Table III, classification errors using different classifiers on the raw data as well as on the NDA features are summarized. We did not perform LDA feature extraction due to the dimensionality expansion.

Again, the classification errors are substantially reduced in the NDA feature space. Although, the linear SVM classified the raw test data extremely bad, using the NDA features a high improvement was also obtained for the linear SVM virtually as good as for the SVM rbf. The classification performances of the different classifiers differ only slightly in the NDA space, but vary extremely in the original space.

The classification results on the Letter Recognition Dataset again show the discriminative power of the NDA features. Moreover, our feature extraction framework can also readily be used for expanding the data dimensionality. Therefore it can be used in cases where the LDA cannot be computed, for example.



	0	1	4	7	9	2	3	5	6	8
0										
1	0.14 (0.02)									
4	0.20 (0.01)	0.09 (0.06)								
7	0.20 (0.02)	0.32 (0.10)	0.00 (0.01)							
9	0.25 (0.05)	0.28 (0.07)	0.70 (0.07)	0.49 (0.07)						
2	4.42 (4.14)	4.38 (4.13)	2.48 (2.25)	4.13 (2.92)	1.76 (1.43)					
3	3.62 (3.35)	2.66 (3.18)	0.65 (0.28)	3.14 (2.15)	11.84 (13.35)	8.72 (9.50)				
5	5.02 (4.49)	2.17 (2.61)	0.69 (0.50)	1.93 (0.62)	15.78 (14.93)	5.09 (6.65)	29.39 (27.03)			
6	6.35 (5.18)	0.86 (1.96)	9.59 (8.15)	0.65 (0.06)	1.98 (1.19)	8.69 (10.21)	4.57 (4.61)	5.89 (6.05)		
8	3.43 (2.87)	3.98 (5.88)	1.94 (0.69)	1.85 (0.76)	9.28 (8.75)	12.71 (13.50)	20.61 (21.44)	13.50 (13.58)	7.30 (6.34)	

Fig. 2. The lower triangle shows binary classification errors obtained on the MNIST test data (training data) using linear classifiers trained in the associated NDA feature space. The NDA feature space was learned using the training data of randomly selected MNIST digits, namely 0, 1, 4, 7 and 9. To visualize the pair-wise situations in the 10-dimensional NDA feature space, the upper triangle shows 2-dimensional linear projections of the NDA features for all possible 2-class combinations of digits.

D. *Experiments - Generalization Ability*

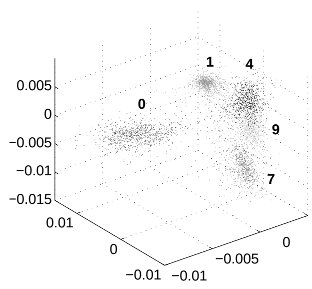
In the preceding sections, we present classification experiments on databases that provide training samples for all pattern classes in the test set. That is, there are a limited number of classes in the respective application domain and the feature extractor is trained by representative samples from all these classes. However, there are many applications where representative samples or even any samples are not available for all classes. One example is face recognition. Obviously, even if a training set contains face images of millions of people, we could by far not provide the training process with image samples of all existing faces. On the other hand, face images can be considered a pattern category. All patterns in that category usually have a somehow similar appearance and format but can be subdivided in classes consisting of face images of a single person only.

A good feature extractor should produce discriminative features for all classes of a pattern category even if training samples are missing for some or even many classes. We want a feature extractor to generalize from the available class information in the training process. The following experiments focus on how well the NDA features generalize in case of partially missing class information.

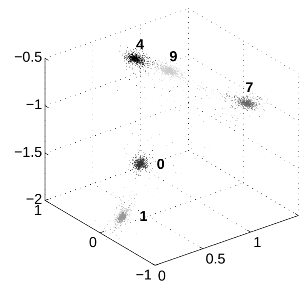
We trained a 784-1000-2000-1000-10 network in the same way as in section IV-A, but this time the training data contained only samples of 5 randomly selected classes of the MNIST database, namely of digits 0, 1, 4, 7, and 9. The training data of digits 2, 3, 5, 6 and 8 were never seen by the feature extractor during training. With the feature extractor obtained by this way the whole MNIST database (including the images of classes, which were ignored during the training process) was transformed to the 10 dimensional NDA feature space.

In order to explore the influence of the missing class information on the pair-wise discriminant power of the NDA features, we trained in the next step for each possible 2-class combination of digits a linear binary classifier using the associated NDA training features. In Figure 2, the lower triangle of the matrix summarizes resulting test errors (training errors) using the linear binary classifiers trained on the NDA features for each class pairing. The upper triangle shows 2-dimensional linear projections for visualization of the pair-wise situations in the 10-dimensional NDA feature space. Hence, if the clouds overlap in the 2D-projections then they overlap also in the NDA feature space and vice versa. Likewise, if the clouds scatter highly, then the corresponding NDA features do so too. Therefore we can analyze the generalization performance of the NDA features based on the 2D-projections and binary classification results

Figure 2 is separated in 4 quadrants. The upper left quad-



(a) LDA test features



(b) NDA test features

Fig. 3. Plots of different 3D features trained with data of MNIST digits 0, 1, 4, 7 and 9. The linear classification error of the NDA test features is 1.36% compared to 8.06% using the LDA test features.

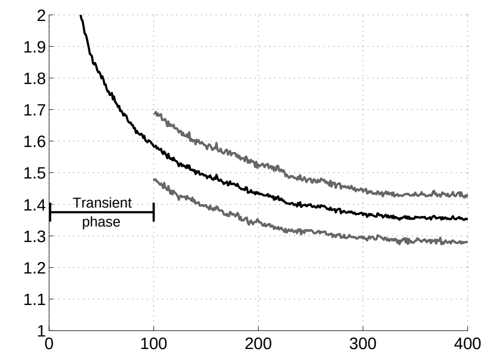


Fig. 4. MNIST mean cross-validation error and standard deviation monitored over 400 epochs of fine-tuning a 784-1000-2000-1000-10 NDA feature extractor.

rant II contains comparisons between classes for which data was used for training the NDA feature extractor. The lower right quadrant IV contains comparisons between classes that were omitted during training the extractor. The other quadrants, I and III, contain mixtures of both.

In quadrant II of Figure 2, very low error rates of less than 1% are reported for test data from the classes that were used for training the NDA feature extractor. The corresponding scatter plots also show well separated and focused clouds. Obviously, such a behavior has to be expected from a good feature extractor. In order to analyze the generalization properties of the features, the other quadrants are most interesting. The most challenging case is presented in quadrant IV. There the scatter plots show that the features of the digits 2, 3, 5, 6 and 8 still form fairly focused clouds, although the NDA feature extractor was never trained with samples of these digits. Moreover, the associated test errors are reasonably low, except for a few very difficult cases, e.g. 8/3 and 5/3. Plots of mixed known/unknown classes are shown in quadrant I and III. One can observe sharply focused clouds for the test data of trained classes (black dots) and less focussed clouds for the test data of untrained classes (gray dots). All features are well separated by a linear classifier (only for the pairs 3/9 and 5/9 the test errors are greater than 10%).

One important aspect of the generalization ability of a feature extractor is the quality of the compactness and the separation of the volumes that are occupied by the classes in the feature space. In order to visualize NDA feature volumes of MNIST data, we chose to train a 784-1000-2000-1000-3 network with the training data for the digits 0, 1, 4, 7 and 9. Although dealing with 5 classes in this case, we reduced the feature dimension to 3 for the purpose of visualization. Additionally, we computed a 3D-LDA transformation using the same raw MNIST training data. Figure 3 shows a scatter plot of the LDA features (a) in comparison to a scatter plot of the NDA features (b). Both plots show the features extracted from the test data, i.e. the volumes occupied by data samples that were never seen by the training process. It can clearly be seen that the NDA feature volumes are much more compact and better separated than the LDA feature volumes.

E. Experiments - Stability of the Pre-Optimization Algorithm

For practical applications of our NDA framework, time consuming training runs without usable results would cause significant problems. Therefore, we conducted experiments to study the stability of our pre-optimization algorithm with respect to variations of the training set and different stochastic runs (the seed for the random number generator was derived from the system time).

For this purpose, we divided the 60,000 training images of the MNIST database into 6 randomly selected cross-validation sets. Applying the NDA framework, we trained a network with topology 784-1000-2000-1000-10 but one for each training set of each cross-validation set. This procedure was repeated twice, resulting in a total of 12 different NDA feature extractors.

In Figure 4, the mean of the cross-validation errors for each of the 400 epochs is shown by the middle solid line. Likewise, the standard deviation of the cross-validation errors at each epoch is represented by the envelopes (after 100 epochs of transient phase). The mean standard deviation computed over the epochs 100 to 400 is only 0.0853% absolute error rate. This is surprisingly low, proving that the training process is neither sensitive to small fluctuations in the training data nor to the random numbers involved.

V. Conclusion

We address the problem of building feature extractors for raw data of arbitrary dimension and distribution. The features shall be highly discriminative with respect to objects for which we have representative training data. That is, a feature extractor shall generate a data representation that facilitates classification with even the simplest classifiers. At the same time, this data representation shall be of low dimension, although dimensionality reduction shall not be a requirement.

DNNs have been used for classification [18] or auto-association [9]. However, we have developed a DNN framework for building nonlinear discriminative feature extractors: In a two-stage training process, we first pre-optimize a DNN using stochastic learning. This stage involves layer-wise training and stacking of single RBMs. We adapt the input units of the first RBM and the output units of the last

RBM of the stack to real-valued data, as this is a requirement for many real world applications. The training of all but the topmost RBM is done using the well known Contrastive Divergence algorithm. This part of the pre-optimization process is unsupervised. In practical terms, it can be interpreted as an adaptation of the network to the pattern world of the particular data domain. For the topmost RBM, we use a novel modification of the Contrastive Divergence heuristics. Together with an appropriate target coding, the pre-optimization is finalized using class information, i.e. with a supervised training scheme aiming at the maximization of a Fisher criterion. This is a prerequisite for the success of the fine-tuning of the feature extractor using a modified back-propagation algorithm that maximizes directly the Fisher criterion in the feature space defined by the last hidden layer of the network.

Since the goal of our work has been to build feature extractors rather than classifiers, we have employed standard classifiers in extensive experiments with well established benchmark databases in order to test the discriminative power of the features generated by the feature extractors.

Our results on the MNIST database, the Statlog's Satimage database and the Letter Recognition Dataset are competitive with the best results reported in the literature. However, a remarkable result is that linear classifiers show about the same classification error compared to nonlinear ones, like SVMs, when applied to the discriminative NDA features generated from the raw data. This is exactly what should be expected from a data representation that makes explicit the discriminative information contained in the raw data. A simple linear classifier operating on low dimensional discriminative features is the only feasible solution in all large scale database search applications. The computational costs for generating the discriminative features have to be spent only once for every data reference or probe respectively. Most important is that the memory space for the features and the computational costs for each individual classification are very low.

Moreover, on the MNIST database, we have shown that a feature extractor trained only on a subset of classes in the whole object domain "handwritten digits" still produces useful discriminative features, even for digits that were not included in the training set. This is a desirable property of a feature extractor in applications where training data is not available for all individual object classes, e.g. face recognition.

In another series of experiments on the MNIST database, we have studied the stability of our pre-optimization algorithm with respect to variations of the training set. Despite the stochastic nature of this process only small fluctuations in the overall performance can be observed. This is an important property of a framework for building feature extractors as the ultimate goal is a construction process that does not require expert knowledge or trial-and-error cycles.

ACKNOWLEDGMENT

The authors would like to thank Hans-Günter Meier for his contribution to some theoretical discussions.

REFERENCES

[1] K. Fukunaga and R. D. Short, "A class of feature extraction criteria and its relation to the bayes risk estimate," *IEEE Trans. Inf. Theory*, vol. 26, pp. 59–65, 1980.

[2] H. Asoh and N. Otsu, "Nonlinear data analysis and multilayer perceptrons," in *Proceedings of International Joint Conference on Neural Networks*, vol. 2, 1989, pp. 411–415.

[3] A. R. Webb and D. Lowe, "The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis," *Neural Networks*, vol. 3, pp. 367–375, 1990.

[4] D. Lowe and A. R. Webb, "Optimized feature extraction and the bayes decision in feed-forward classifier networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, pp. 355–364, 1991.

[5] P. Gallinari, S. Thiria, F. Badran, and F. Fogelman-Soulie, "On the relations between discriminant analysis and multilayer perceptrons," *Neural Networks*, vol. 4, pp. 349–360, 1991.

[6] H. Osman and M. M. Fahmy, "On the discriminatory power of adaptive feed-forward layered networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, pp. 837–842, 1994.

[7] J. Mao and A. K. Jain, "Artificial neural networks for feature extraction and multivariate data projection," *IEEE Trans. Neural Netw.*, vol. 6, pp. 296–317, 1995.

[8] J.-H. Jiang, J.-H. Wang, X. Chu, and R.-Q. Yu, "Non-linear discriminant feature extraction using generalized back-propackagtion network," *Journal of Chemometrics*, vol. 10, pp. 281–294, 1996.

[9] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *SCIENCE*, vol. 313, pp. 504–507, 2006.

[10] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, vol. 9, pp. 147–169, 1985.

[11] G. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algortihm for deep belief nets," *Neural Computation*, vol. 18(7), pp. 1527–1554, 2006.

[12] R. M. Neal and G. E. Hinton, "A new view of EM algorithm that justifies incremental, sparse and other variants." in *Learning in Graphical Models*, M. I. Jordan, Ed. Kulwer Academic Publishers, 1998, pp. 355–368.

[13] J. S. Liu, *Monte Carlo Strategies in Scientific Computing*. Springer Verlag, 2001.

[14] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14(8), pp. 1771–1800, 2002.

[15] M. Welling, M. Rosen-Zvi, and G. Hinton, "Exponential family harmoniums with an application to information retrieval," *NIPS*, vol. 17, pp. 1481–1488, 2005.

[16] Y. LeCun and C. Cortes, *THE MNIST DATABASE of handwritten digits*, Std. [Online]. Available: http://yann.lecun.com/exdb/mnist

[17] A. Asuncion and D. J. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/ simmlearn/MLRepository.html

[18] H. Larochelle, Y. Bengio, J. Loradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *Journal of Machine Learning Research*, vol. 1, pp. 1–40, 2009.