# Deep Boltzmann Machines and the Centering Trick

Grégoire Montavon[1] and Klaus-Robert Müller[1,2]

[1] Technische Universität Berlin, 10587 Berlin, Germany,
Machine Learning Group
[2] Korea University, Anam-dong, Seongbuk-gu, Seoul 136-713, Korea,
Department of Brain and Cognitive Engineering

{gregoire.montavon,klaus-robert.mueller}@tu-berlin.de

Originally published in: G. Montavon, G. B. Orr, K.-R. Müller (Eds.), Neural Networks: Tricks of the Trade, 2nd edn, Springer LNCS 7700, 2012

**Abstract.** Deep Boltzmann machines are in theory capable of learning efficient representations of seemingly complex data. Designing an algorithm that effectively learns the data representation can be subject to multiple difficulties. In this chapter, we present the "centering trick" that consists of rewriting the energy of the system as a function of centered states. The centering trick improves the conditioning of the underlying optimization problem and makes learning more stable, leading to models with better generative and discriminative properties.

**Keywords:** Deep Boltzmann machine, centering, reparameterization, unsupervised learning, optimization, representations.

## 1 Introduction

Deep Boltzmann machines are undirected networks of interconnected units that learn a joint probability density over these units by adapting connections between them. They are in theory capable of learning statistically and computationally efficient representations of seemingly complex data distributions.

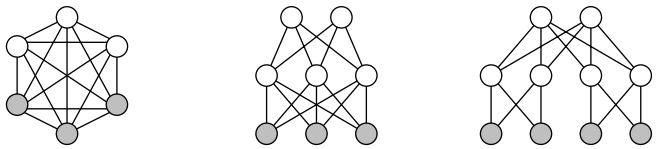
Designing an algorithm that effectively learns the data representation can be subject to multiple difficulties. Deep Boltzmann machines are sensitive to the parameterization of their energy function. In addition, the gradient of the optimization problem is not directly accessible and must instead be approximated stochastically by continuously querying the model throughout training.

In this chapter, we present the "centering trick" that consists of rewriting the energy function of the deep Boltzmann machine as a function of centered states. We argue that centering improves the conditioning of the optimization problem and facilitates the emergence of complex structures in the deep Boltzmann machine.

We demonstrate on the MNIST dataset that the centering trick allows mid-sized deep Boltzmann machines to be trained faster and to produce a solution which is a good generative model of data but also distills interesting discriminative features in the top layer.

## 2 Boltzmann Machines

In this section, we give some background on the Boltzmann machine (Hinton and Sejnowski, 1986). We will use the following notation: The sigmoid function is defined as $\text{sigm}(x) = \frac{e^x}{e^x+1}$, $x \sim \mathcal{B}(p)$ denotes that the variable $x$ is drawn randomly from a Bernoulli distribution of parameter $p$ and $\langle\cdot\rangle_P$ denotes the expectation operator with respect to a probability distribution $P$. All these operations apply element-wise to the input if the latter is a vector.



fully connected Boltzmann machine | deep Boltzmann machine (DBM) | locally connected DBM (LC-DBM)

**Fig. 1.** Example of Boltzmann machines used in practice with visible units depicted in gray and hidden units depicted in white. The layered structure of a DBM is interesting because a particular representation of data forms at each layer, possibly enabling the emergence of interesting statistics.

A Boltzmann machine is a network of $M_x$ interconnected binary units that associates to each state $x \in \{0,1\}^{M_x}$ the probability

$$p(x;\theta) = \frac{e^{-E(x;\theta)}}{\sum_\xi e^{-E(\xi;\theta)}}.$$

The term in the denominator is the called the partition function and makes probabilities sum to one. The function

$$E(x;\theta) = -\frac{1}{2}x^\top W x - x^\top b.$$

is the energy of the state $x$ given the model parameters $\theta = (W, b)$. From these equations, we can interpret a good model of data as a model $\theta$ that has low energy in regions of high data density and high energy elsewhere. The matrix $W$ of size $M_x \times M_x$ is symmetric and contains the connection strengths between units. The vector $b$ of size $M_x$ contains the biases associated to each unit. The diagonal of $W$ is constrained to be zero. Units are either visible units (representing the

sensory input) or hidden units (representing latent variables that are not directly observable but contribute to explaining data).

From the equations above, we can derive the conditional probability of each unit being activated given the other units

$$p(x_i = 1|x_{-i}; \theta) = \text{sigm}(b_i + \sum_{j \neq i} W_{ij} x_j)$$

where $x_{-i}$ denotes the set of all units but $x_i$. The gradient of the data log-likelihood with respect to model parameters $W$ and $b$ takes the form

$$\frac{\partial}{\partial W} \langle \log p(x_{\text{vis}}; \theta) \rangle_{\text{data}} = \langle x x^\top \rangle_{\text{data}} - \langle x x^\top \rangle_{\text{model}} \tag{1}$$

$$\frac{\partial}{\partial b} \langle \log p(x_{\text{vis}}; \theta) \rangle_{\text{data}} = \langle x \rangle_{\text{data}} - \langle x \rangle_{\text{model}} \tag{2}$$

where $x_{\text{vis}}$ are the visible units (i.e. the subset of units that represent the sensory input). The terms $\langle \cdot \rangle_{\text{data}}$ and $\langle \cdot \rangle_{\text{model}}$ are respectively the data-dependent expectations (obtained by conditioning the joint distribution on the observed state of the visible units) and the data-independent expectations obtained by sampling freely from the joint probability distribution.

### 2.1 Deep Boltzmann Machines

It is often desirable to incorporate some predefined structure to the Boltzmann machine. The most common way to achieve this is to remove certain connections in the network, that is, forcing parts of the matrix $W$ to zero. Example of Boltzmann machines with different structures are shown in Figure 1. For example, in the deep Boltzmann machine (DBM) (Salakhutdinov and Hinton, 2009), units are organized in a deep layered manner where only adjacent layers communicate and where units within the same layer are disconnected. Locally connected deep Boltzmann machines add further constraints to the model by forcing the modeling of the interaction between remote parts of the input to take place in the top layer.

The special layered structure of the DBM and its multiple variants has two advantages: First, particular statistics can emerge at each layer that may capture interesting features of data. Second, the layered structure of the DBM can be folded into a bipartite graph (one side containing odd layers and the other side containing even layers) where each side of the graph is conditionally independent given the other side.

In the case of the deep Boltzmann machine shown in Figure 2 (left) with $M_x$, $M_y$ and $M_z$ units at each layer, the energy associated to each state $(x, y, z) \in \{0, 1\}^{M_x + M_y + M_z}$ is

$$E(x, y, z; \theta) = -\, y^\top W x - z^\top V y - x^\top a - y^\top b - z^\top c$$

where $\theta = \{W, V, a, b, c\}$ groups the model parameters. The bipartite graph structure of the deep Boltzmann machine implies that an efficient alternating

Gibbs sampler can be derived:

$$\begin{aligned} x &\sim \mathcal{B}(\text{sigm}(W^\top y + a)) \\ z &\sim \mathcal{B}(\text{sigm}(Vy + c)) \\ y &\sim \mathcal{B}(\text{sigm}(Wx + V^\top z + b)). \end{aligned} \tag{3}$$

A similar alternating Gibbs sampler can be used for sampling states when the input units $x$ are clamped to the data:

$$\begin{aligned} z &\sim \mathcal{B}(\text{sigm}(Vy + c)) \\ y &\sim \mathcal{B}(\text{sigm}(Wx_{\text{data}} + V^\top z + b)). \end{aligned} \tag{4}$$

These alternating Gibbs samplers are illustrated in Figure 2 (right) and allow us to collect the data-independent and data-dependent statistics that intervene in the computation of the gradient (see Equation 1 and 2).

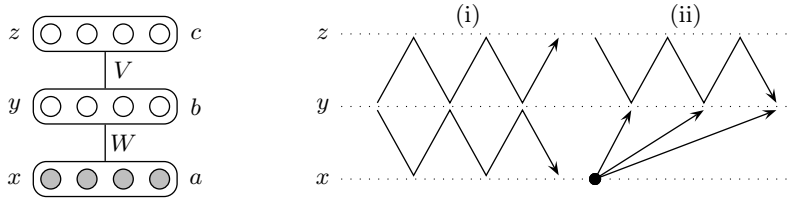



**Fig. 2.** On the left, diagram of a two-layer deep Boltzmann machine along with its parameters. On the right, different sampling methods: (i) the path followed by the alternating Gibbs sampler and (ii) the path followed by the alternating Gibbs sampler when the input is clamped to data.

## 3 Training Boltzmann Machines

While Equation 1 and 2 provide an exact gradient for minimizing the log-likelihood of data, keeping track of data statistics and model statistics is computationally demanding. The mixing rate of the model (i.e. the speed at which the alternating Gibbs sampler converges to the model's true distribution) is typically slow and implies that we need to resort to some approximation.

Collecting *data-dependent statistics* is relatively easy as the complexity of the distribution is reduced by the clamping of visible units to the data. In the case where hidden units are independent when conditioned on the visible units, sampling can be achieved exactly in only one pass of the Gibbs sampler. This is the case of the restricted Boltzmann machine (Hinton, 2002, 2012). In practice, when the number of hidden-to-hidden connections is relatively low or the connections are not particularly strong, reasonable approximations can be obtained by running a few steps of the alternating Gibbs sampler.

Collecting *data-independent statistics* is much harder and typically requires hundreds or thousands of iterations before converging to the true probability distribution. A workaround to the problem is to approximate these statistics by a small set of persistent chains (or "free particles") $\{x_1, \ldots, x_n\}$ that are continuously updated throughout training. This idea called *persistent contrastive divergence* has been proposed by Tieleman (2008).

The intuition behind persistent contrastive divergence is the following: let's first remember that the minima of the energy function correspond to high probability states and that the free particles are therefore inclined to descend the energy function. As the model is trained, the energy of the free particles is raised under the effect of the gradient update and free particles are encouraged to slide down the "bump" created by the gradient update. The higher the learning rate, the higher the bumps, and the faster the particles are descending the energy function. This implies that free particles are mixing much faster under the effect of training than in a static setting.

When training a deep Boltzmann machine, at least two sources of instability can be identified: (1) *Approximation instability:* The stochastic and approximate nature of the learning algorithms described above implies that the estimation of the gradient is noisy. The noise comes in part from the stochastic gradient descent procedure, but principally from the approximate sampling procedure that may cause systematically biased estimates of the gradient. (2) *Structural instability:* As it has been identified by Cho et al. (2011), in standard Boltzmann machines, the weight matrix $W$ tends to model in the first steps of the learning algorithm a global bias instead of co-dependencies between each units as we would expect. This is particularly problematic in the case of a Boltzmann machine with hidden-to-hidden connections such as the DBM, because hidden units tend to conglomerate and form a bias that may speed up learning initially but that eventually destroys the learning signal between pairs of hidden units.

### 3.1 The Centering Trick

The centering trick attempts to mitigate these sources of instability by ensuring that units activations intervening in the computation of the gradient are centered. Centering aims to produce a better conditioned optimization problem that is more robust to the noise of the learning procedure and to avoid the use of units as a global bias. Centering was already advocated by LeCun et al. (1998) and Schraudolph (1998) in the context of backpropagation networks. Centering can be achieved by rewriting the energy of the Boltzmann machine as a function of centered states:

**Center the Boltzmann machine**

$$E(x;\theta) = -\frac{1}{2}(x-\beta)^\top W(x-\beta) - (x-\beta)^\top b$$

The new variable $\beta$ represents the *offset* associated to each unit of the network and must be set to the mean activation of $x$. Setting $\beta = \text{sigm}(b_0)$ where $b_0$

is the initial bias ensures that units are initially centered. A similar parameterization of the energy function has been proposed by Tang and Sutskever (2011) where the offset parameters were restricted to visible units. As we will see later, the Boltzmann machine also benefits from centering the hidden units. From this new energy function, we can derive the conditional probability of each unit in the centered Boltzmann machine:

$$p(x_i = 1|x_{-i}; \theta) = \text{sigm}(b_i + \sum_{j \neq i} W_{ij}(x - \beta)_j).$$

Similarly, the gradient of the model log-likelihood with respect to $W$ and $b$ now takes the form:

$$\frac{\partial}{\partial W}\langle \log p(x_{\text{vis}}; \theta)\rangle_{\text{data}} = \langle (x-\beta)(x-\beta)^\top \rangle_{\text{data}} - \langle (x-\beta)(x-\beta)^\top \rangle_{\text{model}} \quad (5)$$

$$\frac{\partial}{\partial b}\langle \log p(x_{\text{vis}}; \theta)\rangle_{\text{data}} = \langle x-\beta \rangle_{\text{data}} - \langle x-\beta \rangle_{\text{model}}. \quad (6)$$

These gradients are similar to the *enhanced gradients* proposed by Cho et al. (2011) and to those arising from the parameterization proposed by Arnold et al. (2011) at the difference that our gradients do not account for the possibility that offsets $\beta$ deviate from the mean activations of units throughout training. If the latter effect is problematic, it is possible to reparameterize the network continuously or at regular intervals so that the offsets correspond to the new expected means $\langle x \rangle_{\text{data}}$. The reparameterization $\theta \to \theta'$ must leave the energy function invariant up to a constant, that is, $E(x; \theta) = E(x; \theta') + \text{const}$. Solving the equation under the new centering constraints leads to the update equations $W' = W$, $b' = b + W(\langle x \rangle_{\text{data}} - \beta)$ and $\beta' = \langle x \rangle_{\text{data}}$.

**Update biases and offsets at regular intervals**

$$b' = b + W(\langle x \rangle_{\text{data}} - \beta)$$
$$\beta' = \langle x \rangle_{\text{data}}$$

Similar derivations can be made for the deep Boltzmann machine. The energy function of the deep Boltzmann machine becomes $E(x, y, z; \theta) = -(y-\beta)^\top W(x-\alpha) - (z-\gamma)^\top V(y-\beta) - (x-\alpha)^\top a - (y-\beta)^\top b - (z-\gamma)^\top c$ where $\alpha$, $\beta$ and $\gamma$ are the offsets associated to the units in each layer. A basic algorithm for training a centered deep Boltzmann machine is given in Figure 3.

### 3.2 Understanding the Centering Trick

We look at the effect of centering on the stability of learning in a Boltzmann machine. We argue that when the Boltzmann machine is centered, the optimization problem is better conditioned (see Figure 5), more precisely, the ratio between the highest and the lowest eigenvalue of the Hessian $\boldsymbol{H}$ is smaller. We define $\xi$ as

**Training a centered deep Boltzmann machine**

$W, V = 0, 0$
$a, b, c = \mathrm{sigm}^{-1}(\langle x \rangle_{\mathrm{data}}), b_0, c_0$
$\alpha, \beta, \gamma = \mathrm{sigm}(a), \mathrm{sigm}(b), \mathrm{sigm}(c)$
initialize free particle $(x_m, y_m, z_m) = (\alpha, \beta, \gamma)$
**loop**
  initialize data particle $(x_d, y_d, z_d) = (\mathrm{pick}(\mathrm{data}), \beta, \gamma)$
  **loop**
    $y_d \sim \mathcal{B}(\mathrm{sigm}(W(x_d - \alpha) + V^\top(z_d - \gamma) + b))$
    $z_d \sim \mathcal{B}(\mathrm{sigm}(V(y_d - \beta) + c))$
  **end loop**
  $y_m \sim \mathcal{B}(\mathrm{sigm}(W(x_m - \alpha) + V^\top(z_m - \gamma) + b))$
  $x_m \sim \mathcal{B}(\mathrm{sigm}(W^\top(y_m - \beta) + a))$
  $z_m \sim \mathcal{B}(\mathrm{sigm}(V(y_m - \beta) + c))$
  $W = W + \eta \cdot [(y_d - \beta)(x_d - \alpha)^\top - (y_m - \beta)(x_m - \alpha)^\top]$
  $V = V + \eta \cdot [(z_d - \gamma)(y_d - \beta)^\top - (z_m - \gamma)(y_m - \beta)^\top]$
  $a = a + \eta \cdot (x_d - x_m) \; + \; \nu \cdot W^\top(y_d - \beta)$
  $b = b + \eta \cdot (y_d - y_m) \; + \; \nu \cdot W(x_d - \alpha) + \nu \cdot V^\top(z_d - \gamma)$
  $c = c + \eta \cdot (z_d - z_m) \; + \; \nu \cdot V(y_d - \beta)$
  $\alpha = (1 - \nu) \cdot \alpha + \nu \cdot x_d$
  $\beta = (1 - \nu) \cdot \beta + \nu \cdot y_d$
  $\gamma = (1 - \nu) \cdot \gamma + \nu \cdot z_d$
**end loop**

**Fig. 3.** Basic algorithm for training a two-layer centered deep Boltzmann machine. The algorithm is based on persistent contrastive divergence and is kept minimal for the sake of simplicity. The variable $\eta$ is the learning rate and the variable $\nu$ is the rate of the moving average necessary for the reparameterization. A Python implementation of the algorithm is available at `http://gregoire.montavon.name/code/dbm.py`.

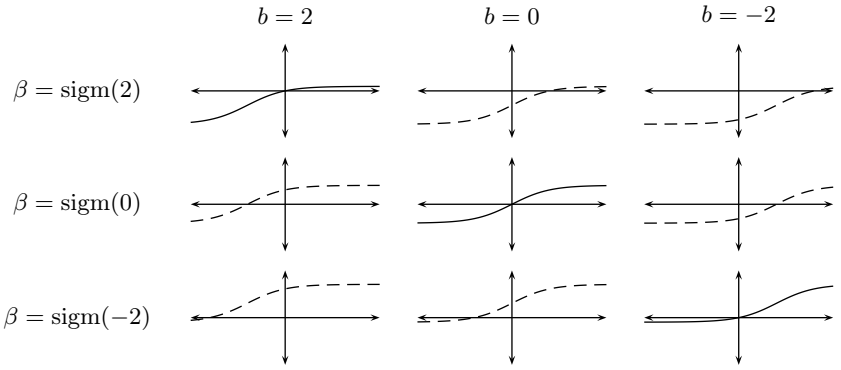



**Fig. 4.** Example of sigmoids $f(x) = \mathrm{sigm}(x + b) - \beta$ with different biases $b$ and offsets $\beta$. This figure illustrates how setting $\beta_0 = \mathrm{sigm}(b_0)$ ensures that the sigmoid crosses the origin initially and do not contribute to modeling a bias component.

the centered state $\xi = x - \beta$. Substituting $x - \beta$ by $\xi$ in Equation 5, the derivative of the data log-likelihood with respect to the weight parameter becomes

$$\frac{\partial}{\partial W} \langle \log p(x;\theta) \rangle_{\text{data}} = \langle \xi \xi^\top \rangle_{W,\text{data}} - \langle \xi \xi^\top \rangle_W$$

where $\langle \cdot \rangle_W$ denotes the expectation with respect to the probability distribution associated to a model of weight parameter $W$ and $\langle \cdot \rangle_{W,\text{data}}$ denotes the expectation of the same model with visible units clamped to data. Using the definition of the directional derivative, the second derivative with respect to a random direction $V$ (which is equal to the projected Hessian $\boldsymbol{H}V$) can be expressed as:

$$\begin{aligned}
\boldsymbol{H}V &= \frac{\partial}{\partial V} \left( \frac{\partial}{\partial W} \langle \log p(x;W) \rangle_{\text{data}} \right) \\
&= \lim_{h \to 0} \frac{1}{h} \left( \frac{\partial}{\partial W} \langle \log p(x;W+hV) \rangle_{\text{data}} - \frac{\partial}{\partial W} \langle \log p(x;W) \rangle_{\text{data}} \right) \\
&= \lim_{h \to 0} \frac{1}{h} \left( (\langle \xi \xi^\top \rangle_{W+hV,\text{data}} - \langle \xi \xi^\top \rangle_{W+hV}) - (\langle \xi \xi^\top \rangle_{W,\text{data}} - \langle \xi \xi^\top \rangle_W) \right) \\
&= \lim_{h \to 0} \frac{1}{h} \left( \langle \xi \xi^\top \rangle_{W+hV,\text{data}} - \langle \xi \xi^\top \rangle_{W,\text{data}} \right) - \lim_{h \to 0} \frac{1}{h} \left( \langle \xi \xi^\top \rangle_{W+hV} - \langle \xi \xi^\top \rangle_W \right)
\end{aligned}$$

From the last line, we can see that the Hessian can be decomposed into a data-dependent term and a data-independent term. A remarkable fact is that in absence of hidden units, the data-dependent part of the Hessian is zero, because the model—and therefore, the perturbation of the model—have no influence on the states. The conditioning of the optimization problem can therefore be analyzed exclusively from a model perspective. The data-dependent term is likely to be small even in the presence of hidden variables due to the sharp reduction of entropy caused by the clamping of visible units to data.

We can think of a well-conditioned model as a model for which a perturbation of the model parameter $W$ in any direction $V$ causes a well-behaved perturbation of state expectations $\langle \xi \xi^\top \rangle_W$. Pearlmutter (1994) showed that in a Boltzmann machine with no hidden units, the projected Hessian can be further reduced to

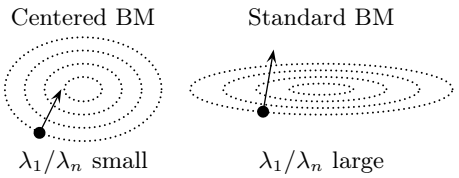
$$\boldsymbol{H}V = \langle \xi \xi^\top \rangle_W \cdot \langle D \rangle_W - \langle \xi \xi^\top D \rangle_W \qquad \text{where} \quad D = \frac{1}{2} \xi^\top V \xi \tag{7}$$

thus, getting rid of the limit and leading to numerically more accurate estimates. LeCun et al. (1998) shows that the stability of the optimization problem can be quantified by the *condition number* defined as the ratio between the largest eigenvalue $\lambda_1$ and the smallest eigenvalue $\lambda_n$ of $\boldsymbol{H}$. A geometrical interpretation of the condition number is given in Figure 5 (left). A low rank approximation of the Hessian can be obtained as

$$\hat{\boldsymbol{H}} = \boldsymbol{H}(V_0 | \dots | V_n) = (\boldsymbol{H}V_0 | \dots | \boldsymbol{H}V_n) \tag{8}$$

where the columns of $(V_0 | \dots | V_n)$ form a basis of independent unit vectors that projects the Hessian on a low-dimensional random subspace. The condition number can then be estimated by performing a singular value decomposition of the

projected Hessian $\hat{\boldsymbol{H}}$ and taking the ratio between the largest and smallest resulting eigenvalues.




| $\lambda_1/\lambda_n$ | $b=2$ | $b=0$ | $b=-2$ |
|---|---|---|---|
| $\beta=\text{sigm}(2)$ | **1.98** | 12.65 | 51.42 |
| $\beta=\text{sigm}(0)$ | 22.97 | **1.82** | 22.20 |
| $\beta=\text{sigm}(-2)$ | 52.72 | 13.40 | **1.94** |

**Fig. 5.** Left: relation between the condition number $\lambda_1/\lambda_n$ and the shape of the optimization problem. Gradient descent is easier to achieve when the condition number is small. Right: condition number obtained for centered Boltzmann machines (shown in bold on the diagonal) and for non-centered deep Boltzmann machines (off-diagonal elements). It can clearly be seen that the condition number is much smaller when the Boltzmann machine is centered.

We estimate the condition number $\lambda_1/\lambda_n$ of a fully connected Boltzmann machine of 50 units at initial state ($W=0$) for different bias and offset parameters $b$ and $\beta$ using Equation 7 and 8. The condition numbers are obtained by drawing 100 random unit vectors for the projection matrix $V$ and for each of them, estimating the statistics by sampling 1000 independent states $\xi$. Numerical estimates are given in Figure 5 (right) and clearly exhibit the better conditioning occurring when the Boltzmann machine is centered (i.e. when $\beta=\text{sigm}(b)$).

## 4 Evaluating Boltzmann Machines

In this section, we present two complementary approaches to evaluating a Boltzmann machine. The first method consists of looking at the discriminative components built in different portions of the Boltzmann machine (e.g. layers of a DBM) using kernels. The analysis is based on the work of Montavon et al. (2011, 2012) that characterizes the representation that emerges from the learning algorithm at each layer of a neural network. Second, we present a method introduced by Salakhutdinov (2008) that measures the generative performance of the Boltzmann machine in terms of log-likelihood of test data.

### 4.1 Discriminative Analysis

When Boltzmann machines incorporate a special structure, for example, via restricted connectivity, it can be useful to measure the discriminative capability of the representations emerging in specific portions of the network. Measuring the discriminative capability of a group of units is non-trivial, because (1) the meaningful information is not carried in a symbolic form but instead, distributed across the multiple units of the group and (2) the mapping of each data point onto these units is not deterministic but instead corresponds to the conditional

distribution over the units in the group given the input $x$. We present here a method that exploits the insight that the projection of the input distribution onto the group of units forms an kernel (with a well-defined feature space (Schölkopf et al., 1999)) that can be analyzed with respect to certain properties (or labels) $t$. The method was first introduced by Montavon et al. (2011) in the context of backpropagation networks. The method is based on the observation that leading kernel principal components can be approximated up to high accuracy from a finite, typically small set of samples (Braun et al., 2008). We propose a family of kernels for representing the top layer of a DBM:

**A family of kernels for deep Boltzmann machines:**

$$k(x, x') = \mathbb{E}_{z,z'}[f(z|x, z'|x')]$$

where $z|x$ and $z'|x'$ denote the random top-layer activities conditioned respectively on data points $x$ and $x'$ and where the function $f(z, z')$ is a similarity metric between $z$ and $z'$. Typical choices for $f$ are:

$$\begin{aligned} \text{linear:} \quad & f(z, z') = \langle z, z' \rangle \\ \text{radial basis function:} \quad & f(z, z') = \exp\Big(-\frac{||z - z'||}{\sigma}\Big) \\ \text{equality:} \quad & f(z, z') = 1_{\{z=z'\}} \end{aligned}$$

These kernels are able to gracefully deal with multimodal posteriors in the top-level distribution as the expectation operator lies outside the "detection function" $f$ and therefore, accounts for the all possible modalities of $z|x$ and $z'|x'$. Note that since the units of the Boltzmann machine are binary, norms $||\cdot||_1^1, \ldots, ||\cdot||_p^p$ are equivalent. Also, the linear and equality functions correspond up to some normalization to the extremal cases of the radial basis function kernel (with $\sigma$ very large or very small). Once a kernel has been chosen, the analysis proceeds in four steps:

1. Collect a small test set $X$ of size $n \times m$ and its associated label matrix $T$ of size $n \times c$ and compute the empirical kernel $K$ of size $n \times n$. The kernel can be built iteratively by running a Gibbs sampler on each data point and taking the average of all kernels. Alternatively a moving average of the kernel matrix can be maintained throughout training in order to keep track of the discriminative performance.
2. Perform an eigendecomposition of the kernel matrix $K = U\Lambda U^\top$ where $\Lambda$ is a diagonal matrix representing the eigenvalues of $K$ sorted by decreasing order, and where the columns of $U$ represent the eigenvectors associated to each eigenvalue. These eigenvectors are the kernel principal components of $X$ with respect to the kernel $k$ and form a non-linear subspace that spans the main directions of variation in the data (Schölkopf et al., 1998).
3. The representation is then evaluated by looking at how many kernel principal components are capturing the task $T$. Let $U_{1..d}$ and $\Lambda_{1..d}$ be the matrices containing respectively the $d$ leading eigenvectors and eigenvalues. Compute

the projected outputs $Y_d = U_{1..d}U_{1..d}^\top T$. These predictions are the optimal fit in the least square sense based on the $d$ kernel principal components[3].

4. Compute the residuals curve $e(d)$ and the AUC:

$$e(d) = ||T - Y_d||_{\mathrm{F}}^2 \qquad \mathrm{AUC} = \frac{1}{n}\sum_{d=1}^{n} e(d) \tag{9}$$

These two quantities serve as metrics for evaluating how well the task is represented by the kernel. An interpretation of residuals curves $e(d)$ is given in Figure 6.

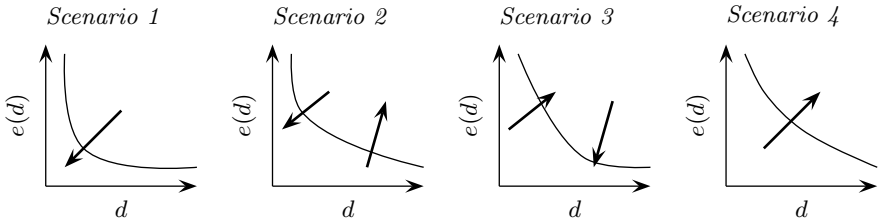



**Fig. 6.** Cartoon showing how to interpret residuals curves yield by various kernels on a certain task. *Scenario 1*: the kernel contains all label-relevant information in its principal components. This is the optimal case. *Scenario 2*: a large amount of label-relevant information is contained in the leading components, but remaining information is missing. *Scenario 3*: the task relevant information is contained in a large number of principal components but can be predicted accurately. *Scenario 4*: the kernel is not suited for the task of interest. Note that although Scenario 2 and 3 have similar AUC, their residuals curves are qualitatively very different.

### 4.2 Generative Analysis

The generative performance, measured in terms of data likelihood is what the Boltzmann machine is optimized for. Unfortunately, data likelihood can not be measured easily as it involves the estimation of the partition function that is generally intractable. As a consequence, we must recourse to sophisticated approximation schemes. We present an analysis introduced by Salakhutdinov and Murray (2008) that estimates the likelihood of the learned Boltzmann machine based on annealed importance sampling (AIS) (Neal, 2001). We describe here the basic analysis. Salakhutdinov (2008) introduced more elaborate procedures for particular types of Boltzmann machines such as restricted, semi-restricted and deep Boltzmann machines.

[3] Alternatively, if we would like to focus on the discrimination boundary between classes, a logistic model of type $\max_\beta \mathrm{trace}(\log(\mathrm{softmax}(\Phi_{1..d}\cdot\beta)\cdot T^\top))$ can be fitted, where $\Phi_{1..d} = U_{1..d}\Lambda_{1..d}^{0.5}$ is the empirical feature space and $\beta$ of size $d \times c$ contains the regression parameters.

As we have seen in Section 2, a deep Boltzmann machine associates to each input $x$ a probability

$$p(x;\theta) = \frac{\Psi(\theta,x)}{Z(\theta)}$$

$$\text{where} \quad \Psi(\theta,x) = \sum_{y,z} p^\star(x,y,z;\theta) \quad \text{and} \quad Z(\theta) = \sum_{x,y,z} p^\star(x,y,z;\theta)$$

and where $p^\star(x,y,z;\theta) = e^{-E(x,y,z;\theta)}$ is the unnormalized probability of state $(x,y,z)$. Computing $\Psi(\theta,x)$ and $Z(\theta)$ analytically is intractable because of the exponential number of elements involved in the sum. We must therefore resort to approximation procedures. Let us first rewrite the ratio of partition functions as follows:

$$p(x;\theta) = \frac{\Psi(\theta,x)}{Z(\theta)} = \frac{\frac{\Psi(\theta,x)}{\Psi(0,x)}}{\frac{Z(\theta)}{Z(0)}} \cdot \frac{\Psi(0,x)}{Z(0)} \tag{10}$$

It can be noticed that the ratio of base-rate partition functions ($\theta = 0$) is easy to compute as $\theta = 0$ makes all units independent. It has the analytical form

$$\frac{\Psi(0,x)}{Z(0)} = \frac{1}{2^{M_x}}. \tag{11}$$

The two other ratios in Equation 10 can be estimated using annealed importance sampling. The annealed importance sampling method proceeds as follows:

**Annealed importance sampling (AIS) (Neal, 2001):**

1. Generate a sequence of states $\xi_1, \ldots, \xi_T$ using a sequence of transition operators $\mathcal{T}(\xi,\xi';\theta_0), \ldots, \mathcal{T}(\xi,\xi';\theta_K)$ that leave $p(\xi)$ invariant, that is,
   - Draw $\xi_0$ from the base model (e.g. a random vector of zero and ones)
   - Draw $\xi_1$ given $\xi_0$ using $\mathcal{T}(\xi,\xi';\theta_1)$
   - ...
   - Draw $\xi_K$ given $\xi_{K-1}$ using $\mathcal{T}(\xi,\xi';\theta_K)$
2. Compute the importance weight

$$\omega_{\text{AIS}} = \frac{p^\star(\xi_1;\theta_1)}{p^\star(\xi_1;\theta_0)} \cdot \frac{p^\star(\xi_2;\theta_2)}{p^\star(\xi_2;\theta_1)} \cdot \ldots \cdot \frac{p^\star(\xi_K;\theta_K)}{p^\star(\xi_K;\theta_{K-1})}$$

It can be shown that if the sequence of models $\theta_0, \theta_1, \ldots, \theta_K$ where $\theta_0 = 0$ and $\theta_K = \theta$ evolves slowly enough, the importance weight obtained with the annealed importance sampling procedure is an estimate for the ratio between the partition function of the model $\theta$ and the partition function of the base rate model.

In our case, $\xi$ denotes the state $(x, y, z)$ of the DBM and the transition operator $\mathcal{T}(\xi, \xi'; \theta)$ corresponds to the alternating Gibbs samplers defined in Equation 3 and 4. The sequence of parameters $\{\theta_0, \ldots, \theta_K\}$ can, for example, lie on the line between 0 and $\theta$, that is, $\theta_k = \alpha_k \cdot \theta_K$ where $\alpha_0 < \cdots < \alpha_K$. Alternatively, the sequence of parameters can be those that are observed throughout training. In that case, maintaining a moving average of the parameter throughout training is necessary as the learning noise creates unnecessarily large variations between two adjacent parameters.

We can now compute the two ratios of partition functions of Equation 10 as

$$\frac{Z(\theta)}{Z(0)} \approx \mathrm{E}[\omega_{\mathrm{AIS}}] \quad \text{and} \quad \frac{\Psi(\theta, x)}{\Psi(0, x)} \approx \mathrm{E}[\nu_{\mathrm{AIS}}(x)] \tag{12}$$

where $\omega_{\mathrm{AIS}}$ is the importance weight resulting from the annealing process with the freely running Gibbs sampler and $\nu_{\mathrm{AIS}}$ is the importance weight resulting from the annealing with input units clamped to the data point. Substituting Equation 11 and 12 into Equation 10, we obtain

$$p(x; \theta) \approx \frac{\mathrm{E}[\nu_{\mathrm{AIS}}(x)]}{\mathrm{E}[\omega_{\mathrm{AIS}}]} \cdot \frac{1}{2^{M_x}}$$

and therefore, the log-likelihood of the model is estimated as:

$$\mathrm{E}_X[\log(p(x; \theta))] \approx \mathrm{E}_X[\log \mathrm{E}[\nu_{\mathrm{AIS}}(x)]] - \log \mathrm{E}[\omega_{\mathrm{AIS}}] - M_x \log(2). \tag{13}$$

Generally, computing an average of the importance weight $\nu_{\mathrm{AIS}}$ for each data point $x$ can take a long time. In practice, we can approximate it with a single AIS run for each data point. In that case, it follows from Jensen's inequality that

$$\mathrm{E}_X[\log \nu_{\mathrm{AIS}}(x)] - \log \mathrm{E}[\omega_{\mathrm{AIS}}] \leq \mathrm{E}_X[\log \mathrm{E}[\nu_{\mathrm{AIS}}(x)]] - \log \mathrm{E}[\omega_{\mathrm{AIS}}]. \tag{14}$$

Consequently, this approximation tends to produce slightly pessimistic estimates of the model log-likelihood, however the variance of $\nu_{\mathrm{AIS}}$ is low compared to the variance of $\omega_{\mathrm{AIS}}$ because the clamping of visible units to data points reduces the diversity of AIS runs.

## 5 Experiments

In this section, we present a few experiments that demonstrate the effectiveness of the centering trick for learning deep Boltzmann machines. We use the MNIST handwritten digits recognition dataset that consists of 60000 training samples and 10000 test samples. Each sample is a $28 \times 28$ grayscale image representing a handwritten digit along with its label. Grayscale values (between 0 and 1) are treated as probabilities.

*Architectures:* We consider a deep Boltzmann machine (DBM) made of $28 \times 28$ input units, 200 intermediate units and 25 top units and a locally-connected DBM (LC-DBM) made of $28 \times 28$ input units, 400 intermediate units that connect to random input patches of size $6 \times 6$ and 100 top units. These architectures are illustrated in Figure 7 (left). In the DBM, the modeling load is concentrated in the first layer with the top layer serving merely to model global digit-like features. On the other hand, in the LC-DBM, most of the modeling load is postponed to the second layer and the first layer serves essentially as a low-level local preprocessor. The initial offsets and biases for visible units are set to $\alpha = \langle x \rangle_{\text{data}}$ and $a_0 = \text{sigm}^{-1}(\alpha)$. We consider different initial biases ($b_0, c_0 = -2$, $b_0, c_0 = 0$ and $b_0, c_0 = 2$) and offsets ($\beta, \gamma = \text{sigm}(-2)$, $\beta, \gamma = \text{sigm}(0)$ and $\beta, \gamma = \text{sigm}(2)$) for the hidden units. These offsets and initial biases correspond to the sigmoids plotted in Figure 4.

*Learning:* We use persistent contrastive divergence (Tieleman, 2008) to train the network and keep track of 25 free particles in background of the learning procedure. We use a Gibbs sampler to collect *both* the data-independent and data-dependent statistics. At each iteration of the learning procedure, we run 3 iterations of the alternating Gibbs sampler for collecting the data-dependent statistics (from a minibatch of 25 data points) and one iteration for updating the data-independent statistics. We use a learning rate $\eta = 0.0004$ per sample for each layer.

*Evaluation:* Evaluating the DBM in an online fashion requires to keep track of the model parameters throughout training. We reduce the learning noise by maintaining a moving average of the sequence of parameters observed during learning. The moving average is tuned to remember approximately 10% of the training history. We keep track of 500 data-dependent chains running on the smoothed sequence of parameters and from which top-layer statistics $k(z, z')$ and ratios $\Psi(\theta, x)/\Psi(0, x)$ are estimated. We also keep track of 100 data-independent chains on the same sequence of parameters and from which the ratio $Z(\theta)/Z(0)$ is estimated. Discriminative performance is measured as the projection residuals of the labels on the kernel principal components and the area under the error curve (see Equation 9) using an exponential RBF kernel with $\sigma$ set to the mean of pairwise distances between $z$ and $z'$. Generative performance is measured in terms of data log-likelihood (see Equation 13).

*Results:* Figure 7 summarizes the results of our analysis and corroborates the importance of centering for obtaining a better discriminative and generative model of data. The centered DBM systematically produces better top-layer AUC errors and has higher log-likelihood. The importance of centering for improving generative models is particularly marked for the locally-connected DBM (LC-DBM) where the top-layer is crucial for modeling long-range dependencies. These results suggest that the centering trick is particularly useful when dealing with hierarchical architectures where global statistics are handled only in the deep layers of the network. Figure 8 (left) shows that the centered DBM yields a

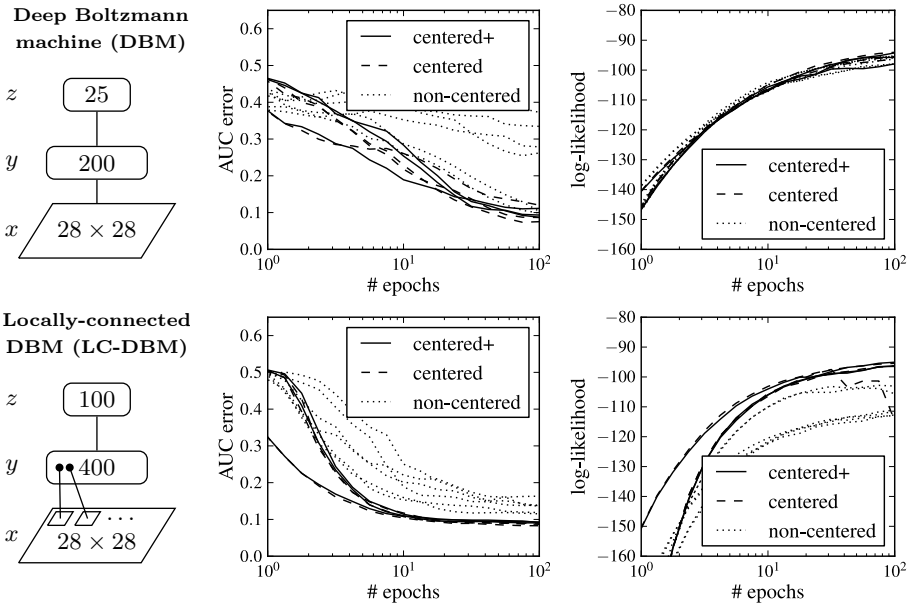



**Fig. 7.** Evolution of the AUC error and log-likelihood throughout training. "Centered+" designates deep Boltzmann machines that are continuously recentered throughout training. In the DBM, reasonable generative performance can be achieved without centering as the top layer is simply ignored by the rest of the model. In the LC-DBM, centering is important for both generative and discriminative performance as the top layer is required for modeling long-range dependencies. Continuously recentering yields the most robust performance.

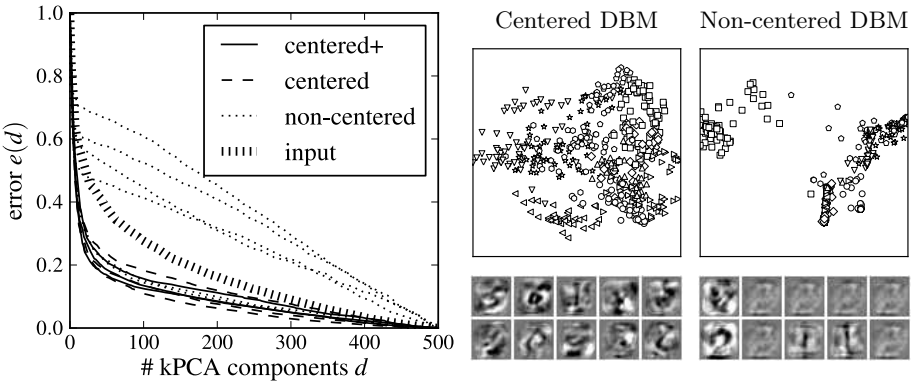



**Fig. 8.** Comparison of the top-layer representation produced by centered and non-centered DBMs. Left: error residuals produced by centered DBMs and non-centered DBMs. Right: 2D-PCA (with a linear kernel) and second-layer filters. Results suggest richer top-layer representations for centered DBMs than for non-centered DBMs.

kernel that contains most of the label information in its leading components and has a low level of noise in the remaining components. This corresponds to Scenario 1 of Figure 6. On the other hand, in the case of the non-centered DBM, the labels span a few leading components of the kernel, but the remaining components have a high level of noise. This corresponds to Scenario 2 of Figure 6. As a comparison, the simple input-layer RBF kernel exhibits high dimensionality and low noise and thus, corresponds to Scenario 3 of Figure 6. The importance of centering for producing good top-layer kernels is further confirmed by looking at the second layer filters, visualized in Figure 8 (right) using a linear back-projection, and observing that they are much richer for the centered DBM than for the non-centered one.

## 6 Conclusion

Learning deep Boltzmann machines is a difficult optimization problem that can be highly sensitive to the parameterization of its energy function. In this chapter, we propose the *centering trick* that consists of rewriting the energy as a function of centered states. The centering trick improves the stability of deep Boltzmann machines and allows to learn models that exhibit both advantageous discriminative and generative properties.

Our experiments have been most successful on mid-scale models (in the range of a few hundred hidden units). The high representational power of deep Boltzmann machines makes it hard to extend our experiments to larger scale models (of thousands of units) without using an explicit regularizer such as layer-wise pretraining or limited connectivity. We believe that applying the centering trick to large-scale models should be made in conjunction with a strong regularizer that limits the effective dimensionality of the model.

*Acknowledgements:* The authors thank Mikio Braun and the multiple reviewers for their useful comments. This work was supported by the World Class University Program through the National Research Foundation of Korea funded by the Ministry of Education, Science, and Technology, under Grant R31-10008. The authors also acknowledge partial support by DFG (MU 987/17-1).

# Bibliography


Ludovic Arnold, Anne Auger, Nikolaus Hansen, and Yann Ollivier. Information-geometric optimization algorithms: A unifying picture via invariance principles, 2011. arXiv:1106.3708.

Mikio L. Braun, Joachim Buhmann, and Klaus-Robert Müller. On relevant dimensions in kernel feature spaces. *Journal of Machine Learning Research*, 9:1875–1908, Aug 2008.

KyungHyun Cho, Tapani Raiko, and Alexander Ilin. Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines. In *Proceedings of the 28th International Conference on Machine Learning*, pages 105–112, 2011.

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

Geoffrey E. Hinton. A practical guide to training restricted Boltzmann machines. In G. Montavon, G. B. Orr, and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade, Reloaded*. Springer LNCS 7700, 611–631, 2012.

Geoffrey E. Hinton and Terrence J. Sejnowski. *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1*, chapter Learning and relearning in Boltzmann machines, pages 282–317. MIT Press, 1986.

Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient Backprop. in G. B. Orr, K.-R. Müller, editors, *Neural Networks, Tricks of the Trade*, Springer LNCS 1524, 9–50, 1998.

Grégoire Montavon, Mikio L. Braun, and Klaus-Robert Müller. Kernel analysis of deep networks. *Journal of Machine Learning Research*, 12:2563–2581, 2011.

Grégoire Montavon, Mikio L. Braun, and Klaus-Robert Müller. Deep Boltzmann machines as feed-forward hierarchies. *Journal of Machine Learning Research - Proceedings Track*, volume 22, pages 789–804, 2012.

Radford M. Neal. Annealed importance sampling. *Statistics and Computing*, 11 (2):125–139, 2001.

Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.

Ruslan Salakhutdinov and Geoffrey E. Hinton. Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455, 2009.

Ruslan Salakhutdinov. Learning and Evaluating Boltzmann Machines. Technical Report UTML TR 2008-002, Dept. of Computer Science, University of Toronto, 2008.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 872–879, 2008.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10 (5):1299–1319, 1998.

Bernhard Schölkopf, Sebastian Mika, Christopher J. C. Burges, Phil Knirsch, Klaus-Robert Müller, Gunnar Rätsch, and Alexander J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks*, 10(5): 1000–1017, 1999.

Nicol N. Schraudolph. Centering Neural Network Gradient Factors. in G. B. Orr, K.-R. Müller, editors, *Neural Networks, Tricks of the Trade*, Springer LNCS 1524, 207–226, 1998.

Yichuan Tang and Ilya Sutskever. Data normalization in the learning of restricted Boltzmann machines. Technical Report UTML-TR-11-2, Department of Computer Science, University of Toronto, 2011.

Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1064–1071, 2008.