Experiments with a restricted Boltzmann machine

Telmo Amaral

21st December 2012

NNIG Technical Report No. 1/2012

Project “Reusable Deep Neural Networks: Applications to Biomedical Data”
(PDTC/EIA-EIA/119004/2010)

Instituto de Engenharia Biomédica (INEB)
Rua Dr. Roberto Frias, 4200-465, Porto, Portugal

Contents

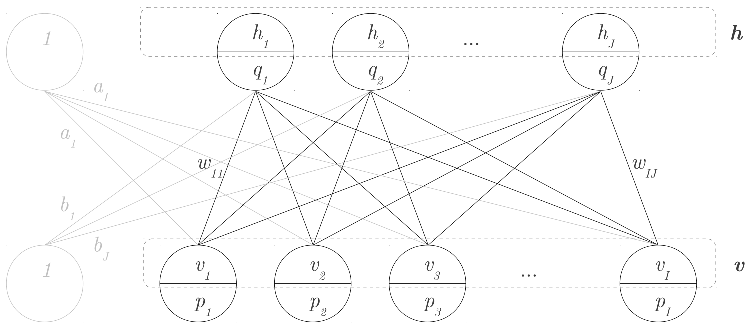


Figure 1: Architecture of an RBM.

1 Introduction

Restricted Boltzmann Machines (RBMs) [2] are an important concept in deep learning, since they can be used as a strategy for the unsupervised pre-training of hidden layers in deep networks. Thus it was decided that I should create a Matlab implementation of basic training algorithms for RBMs, featuring a friendly visualisation that would help to understand (and convey to the other project participants) the functioning of the training process.

The following sections describe the notation used in this report, the two alternative training algorithms that were implemented, the Matlab visualisation of the training process, and some experiments that were carried out.

2 Notation

Figure 1 depicts the architecture of an RBM, while Table 1 explains the notation used throughout this report.

3 Implemented training algorithms

The implemented algorithms were based on the RBM training procedures described in the 2010 technical report “A practical guide to training restricted Boltzmann machines” by Hinton [3]. Two alternative training algorithms were covered, namely using repeated Gibbs sampling and using contrastive divergence with one step of Gibbs sampling (CD-1). These are described in the following subsections.

3.1 Algorithm based on Gibbs sampling

All “for” loops over samples (n) or over visible or hidden units (i or j) can be parallel loops, i.e. the iterations that make up a loop are independent and can be executed concurrently.

Table 1: Notation used in this report.

symbol	meaning
I	number of visible units
J	number of hidden units
w_{ij}	weight of connection between ith visible unit and jth hidden unit
a_i	bias of ith visible unit
b_j	bias of jth hidden unit
$\mathbf{v}$	vector of visible states
v_i	state of ith visible unit
p_i	$p(v_i = 1\|\mathbf{h}) = \sigma(a_i + \sum_j h_j w_{ij})$ $\sigma(x) = 1/(1 + \exp(-x))$
$\mathbf{h}$	vector of hidden states
h_j	state of jth hidden unit
q_j	$p(h_j = 1\|\mathbf{v}) = \sigma(b_j + \sum_i v_i w_{ij})$
N	number of data samples
$\mathbf{x}^{(n)}$	nth data sample
N_m	number of samples from model
ϵ	learning rate
m	learning momentum
d	weight decay

The sequence of actions marked with "*" corresponds to one step of Gibbs sampling.

An epoch (i.e. a training iteration):

```
1. for $n = 1...N$
    (a) set visible states to $n$th data sample: $\mathbf{v}^{(n)} = \mathbf{x}^{(n)}$
    (b) for all $j$ compute hidden probabilities $q_j^{(n)}$
2. for all $i$ and $j$ compute expectations over data distribution
    (a) $\langle v_i q_j \rangle_{data} = \frac{1}{N} \sum_n v_i^{(n)} q_j^{(n)}$
    (b) $\langle v_i \rangle_{data} = \frac{1}{N} \sum_n v_i^{(n)}$
    (c) $\langle q_j \rangle_{data} = \frac{1}{N} \sum_n q_j^{(n)}$
3. for $n = 1...N_m$Training iteration $t$:
    (a) set visible vector $\mathbf{v}^{(n)}$ to random states
    (b) for many times, do a Gibbs sampling step:
        i. * for all $j$ compute hidden probability $q_j^{(n)}$
        ii. * for all $j$ sample hidden state $h_j^{(n)}$ from $q_j^{(n)}$
        iii. * for all $i$ compute visible probability $p_i^{(n)}$
        iv. * for all $i$ sample visible state $v_i^{(n)}$ from $p_i^{(n)}$
    (c) for all $j$ compute hidden probability $q_j^{(n)}$
4. for all $i$ and $j$ compute expectations over model distribution
    (a) $\langle v_i q_j \rangle_{model} = \frac{1}{N_m} \sum_n v_i^{(n)} q_j^{(n)}$
    (b) $\langle v_i \rangle_{model} = \frac{1}{N_m} \sum_n v_i^{(n)}$
    (c) $\langle q_j \rangle_{model} = \frac{1}{N_m} \sum_n q_j^{(n)}$
5. for all $i$ and $j$ compute changes in weights and biases
    (a) $\Delta w_{ij} = m \Delta w_{ij} + \epsilon(\langle v_i q_j \rangle_{data} - \langle v_i q_j \rangle_{model} - d w_{ij})$
    (b) $\Delta a_i = m \Delta a_i + \epsilon(\langle v_i \rangle_{data} - \langle v_i \rangle_{model})$
    (c) $\Delta b_j = m \Delta b_j + \epsilon(\langle q_j \rangle - \langle q_j \rangle_{model})$
6. for all $i$ and $j$ apply changes in weights and biases
    (a) $w_{ij} = w_{ij} + \Delta w_{ij}$
    (b) $a_i = a_i + \Delta a_i$
    (c) $b_j = b_j + \Delta b_j$
```

3.2 Algorithm based on CD-1 (contrastive divergence with one Gibbs step)

An epoch:

```
1. for $n = 1...N$
    (a) set visible states to $n$th data sample: $\mathbf{v}^{(n.0)} = \mathbf{x}^{(n)}$
    (b) * for all $j$ compute hidden probability $q_j^{(n.0)}$
    (c) * for all $j$ sample hidden state $h_j^{(n.0)}$ from $q_j^{(n.0)}$
    (d) * for all $i$ compute reconstructed visible probability $p_i^{(n.1)}$
    (e) * for all $i$ sample a reconstructed visible state $v_i^{(n.1)}$ from $p_i^{(n.1)}$
    (f) for all $j$ compute hidden probability $q_j^{(n.1)}$
2. for all $i$ and $j$ compute expectations over data distribution
    (a) $\langle v_i q_j \rangle_{data} = \frac{1}{N} \sum_n v_i^{(n.0)} q_j^{(n.0)}$
    (b) $\langle v_i \rangle_{data} = \frac{1}{N} \sum_n v_i^{(n.0)}$
    (c) $\langle q_j \rangle_{data} = \frac{1}{N} \sum_n q_j^{(n.0)}$
3. for all $i$ and $j$ compute expectations over reconstructions
    (a) $\langle v_i q_j \rangle_{recon} = \frac{1}{N} \sum_n v_i^{(n.1)} q_j^{(n.1)}$
    (b) $\langle v_i \rangle_{recon} = \frac{1}{N} \sum_n v_i^{(n.1)}$
    (c) $\langle q_j \rangle_{recon} = \frac{1}{N} \sum_n q_j^{(n.1)}$
4. for all $i$ and $j$ compute changes in weights and biases
    (a) $\Delta w_{ij} = m\Delta w_{ij} + \epsilon(\langle v_i q_j \rangle_{data} - \langle v_i q_j \rangle_{recon} - d w_{ij})$
    (b) $\Delta a_i = m\Delta a_i + \epsilon(\langle v_i \rangle_{data} - \langle v_i \rangle_{recon})$
    (c) $\Delta b_j = m\Delta b_j + \epsilon(\langle q_j \rangle - \langle q_j \rangle_{recon})$
5. for all $i$ and $j$ apply changes in weights and biases
    (a) $w_{ij} = w_{ij} + \Delta w_{ij}$
    (b) $a_i = a_i + \Delta a_i$
    (c) $b_j = b_j + \Delta b_j$
```

4 Matlab visualisation

Figure 2 shows an aspect of the created visualisation, during the training of an RBM with four visible units and five hidden units. In this example, the probability of the third hidden unit (i.e. the value shown on the bottom half of the circle representing the unit) has been updated by forward propagation,

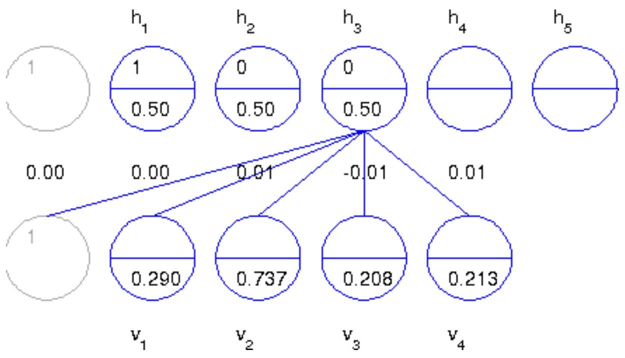


Figure 2: Aspect of the visualisation of an RBM being trained.

based on the values of the four visible units and visible bias and on the weights of the five associated connections. The actual value of the hidden unit was then sampled from the updated probability.

Via the use of breakpoints, this visualisation allows to follow the process of updating the probability and sampling the value of each unit at a time, for both hidden units (forward propagation) and visible units (backward propagation). Any numbers of hidden and visible units can be defined, as long as they fit reasonably within the visualisation window.

5 Experiments

Experiments were carried using different types of data (e.g. binary and real-valued, visual and non-visual), to produce plots that helped to understand how the generative nature of the RBM model allows it to develop the ability to reconstruct the training data as training progresses. The three most important experiments are described in the following subsections, named after the type of data involved.

5.1 Binary 3×3 patterns

This experiment is coded in `script1.m`.

Parameters

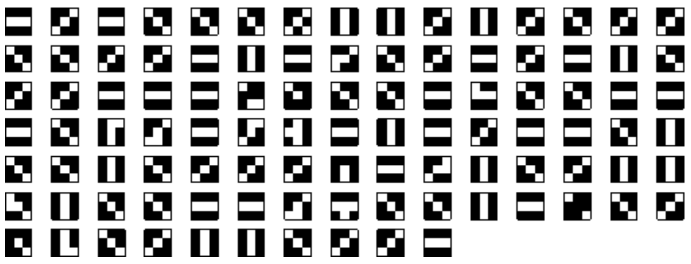
		see note
I	9	
N	100	
J	8	1
standard deviation. for initialisation of w_{ij}	0.01	
initial a_i		2
number of epochs	300	
N_m	100	
number of Gibbs sampling steps	10	
ϵ	0.0930	3
m	0.9	4
d	0.0002	

Notes:

1. J computed from $I + J + I \times J = I \times N/10$, based on Hinton's recipe: "estimate how many bits it would take to describe each data-vector (...). Then multiply that estimate by the number of training cases and use a number of parameters that is about an order of magnitude smaller."
2. a_i initialised based on Hinton's recipe: "It is usually helpful to initialize the bias of visible unit i to $log(p_i/(1 - p_i))$ where p_i is the proportion of training vectors in which unit i is on."
3. ϵ set on first iteration so that mean absolute weight change is about 1/100 of mean absolute weight. This is a simplification of Hinton's recipe: "A good rule of thumb for setting the learning rate (...) is to look at a histogram of the weight updates and a histogram of the weights. The updates should be about 10^{-3} times the weights (to within about an order of magnitude)."
4. The setting of m could be improved (maybe interactively) based on Hinton's recipe: "Start with a momentum of 0.5. Once the large initial progress in the reduction of the reconstruction error has settled down to gentle progress, increase the momentum to 0.9."

Training data

Shown below is a set of 100 3×3 binary visual patterns used to train the RBM, corresponding to horizontal, vertical, and diagonal patterns. These data were created adding some "noise": the probability of any given bit (pixel) being inverted was 0.02.

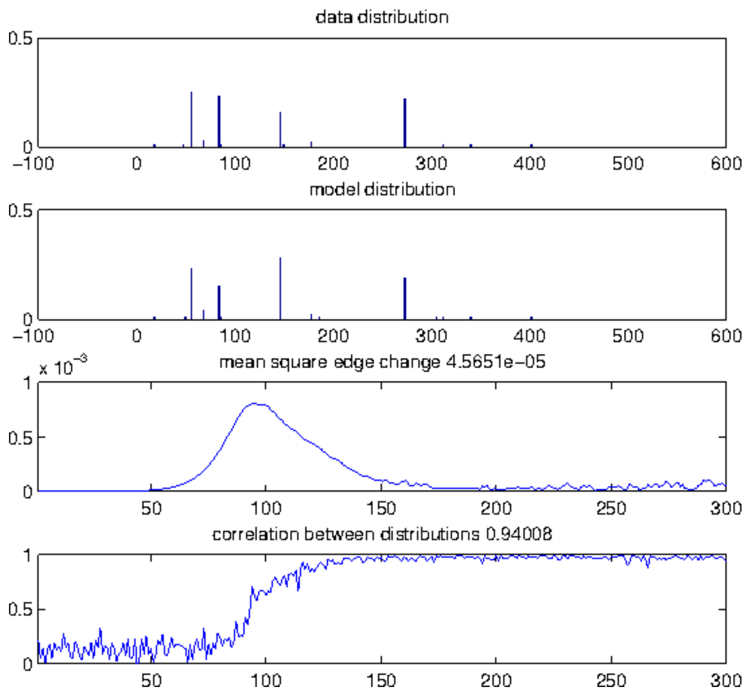


Training based on Gibbs sampling

The first plot blow is a histogram of training data vectors. The second plot is a histogram of model samples after the 300 training epochs (iterations), i.e. the distribution of reconstructed data. Since the patterns are binary and formed by 9 pixels each, they can be represented as binary 9-bit words. So, there are $2^{I=9} = 512$ possible data vectors / model samples. It is apparent that the distribution of reconstructed data converged towards the distribution of the training data.

The third plot shows how the mean square change on weights of the connections between visible and hidden units of the RBM evolves as a function of the training epoch. The fourth plot shows how the correlation between the data and model distributions evolves as a function of the epoch. It can be seen that the connection weights changed mostly between epochs 50 and 150, when the accuracy of reconstruction increased more sharply.

The mean duration of an epoch was 1.0593 seconds.

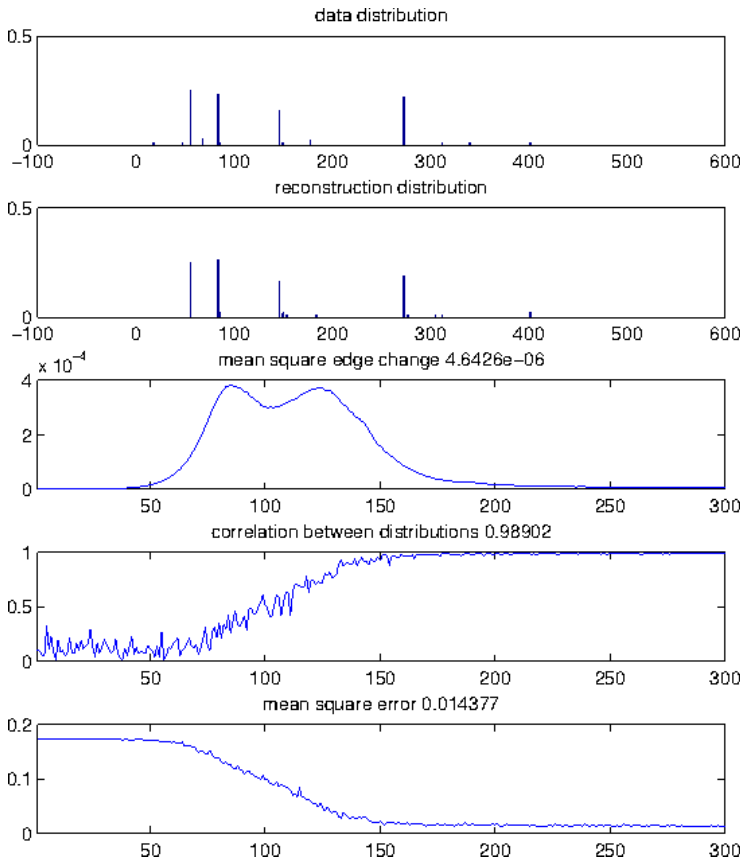


Training based on CD-1

The second plot below is now a histogram over data vector reconstructions.

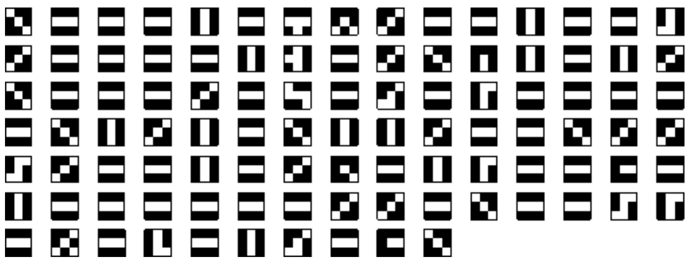
The fifth plot shows how the mean square error (between data bits and reconstructed bit probabilities) evolves as a function of the epoch.

The mean duration of an epoch was 0.1464 seconds (seven times faster than using Gibbs sampling with 10 steps).



Generated data

Shown below are 100 data samples generated (i.e. synthesised) by the RBM once it was trained. Each sample was generated after 10 Gibbs sampling steps.



5.2 Grey-level 3×3 patterns

The implemented version of RBM is more appropriate for use with binary data. Nevertheless, it can be trained with real-valued data.

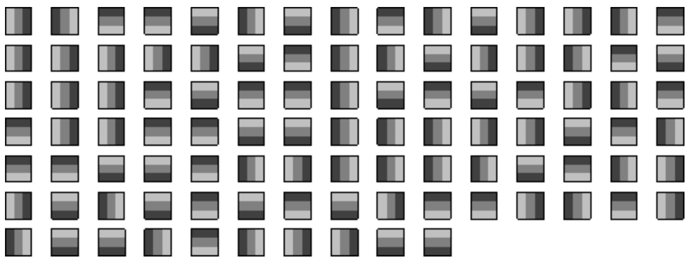
This experiment is coded in `script2.m`.

Parameters

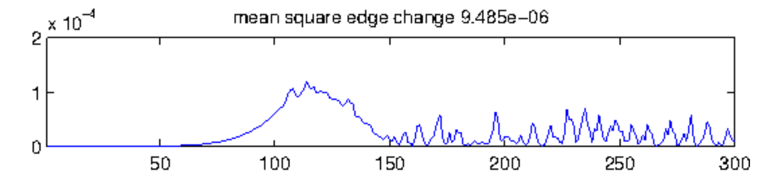
		see note
I	9	
N	100	
J	8	1
standard deviation for initialisation of w_{ij}	0.01	
initial a_i		2
number of epochs	300	
ϵ	0.0505	3
m	0.9	4
d	0.0002	

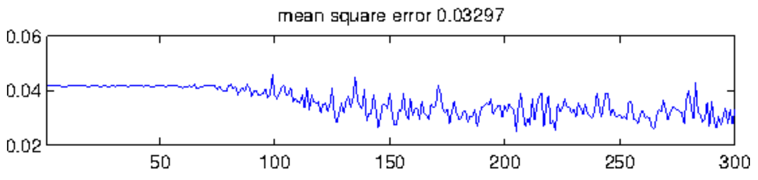
Training data

Shown below are 100 real-valued 3×3 grey-level visual patterns used to train the RBM. These data were created without any noise.



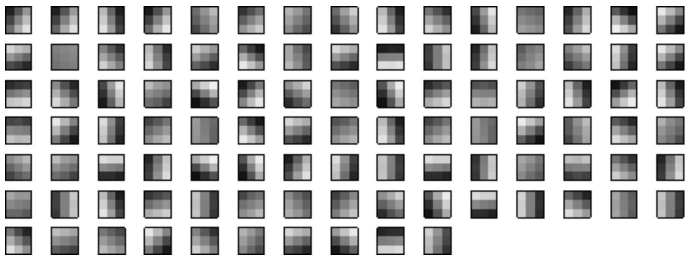
Training based on CD-1





Generated data

Each sample shown below was generated after 10 Gibbs sampling steps. It can be seen that the generated data roughly approximate the patterns present in the training data, though not as successfully as in the binary case.



5.3 Iris data

This experiment is coded in `script3.m`.

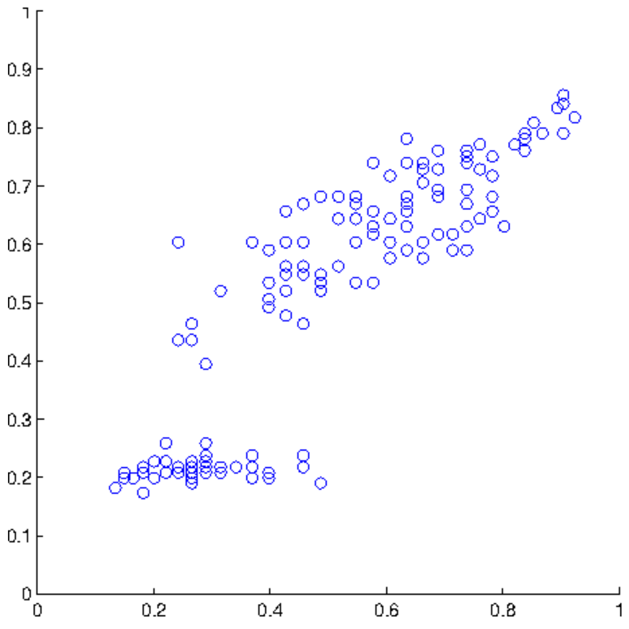
Parameters

		see note
I	4	
N	150	
J	11	1
standard deviation for initialisation of w_{ij}	0.01	
initial a_i		2
number of epochs	300	
ϵ	0.0480	3
m	0.9	4
d	0.0002	

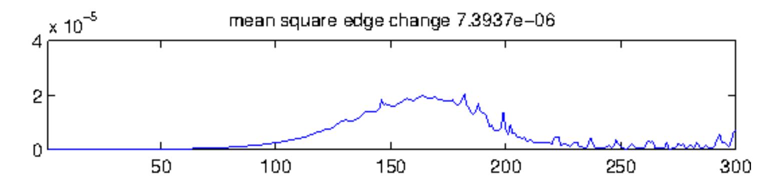
Training data

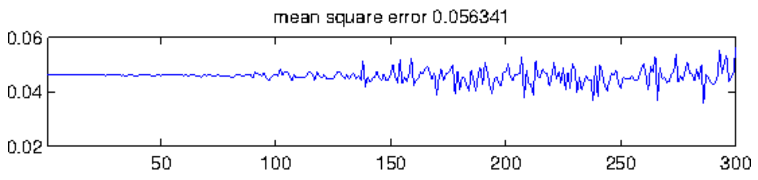
The original data were normalised to zero mean and unit variance, then squashed between 0 and 1, via a sigmoid function.

This is the scatter plot over features 1 (sepal length, horizontal axis) and 3 (petal length, vertical axis):



Training based on CD-1



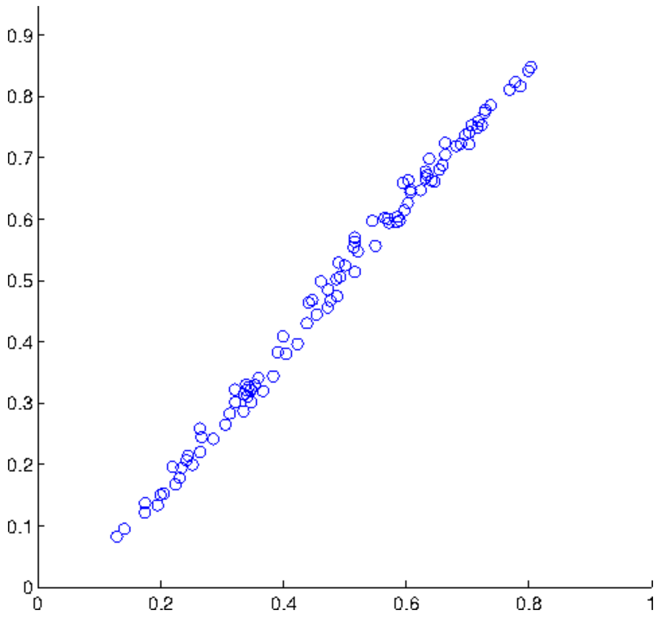


Generated data

Each sample was generated after 10 Gibbs sampling steps.

It is clear that, in this case, the RBM was not able to properly learn the data distribution.

During my visit to the LISA lab at the University of Montreal, Prof. Yoshua Bengio offered some comments on these experiments, which are included in a separate activities report Amaral [1].



References

[1] T. Amaral. Activities report from August 2012 to April 2013, April 2013. 14

[2] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002. 3

[3] G. Hinton. A practical guide to training restricted boltzmann machines. Technical report, University of Toronto, 2010. 3