Faculty of Science

Undirected Graphical Models

Advanced Topics in Data Modelling

Christian Igel

Department of Computer Science, University of Copenhagen

igel@diku.dk

Outline

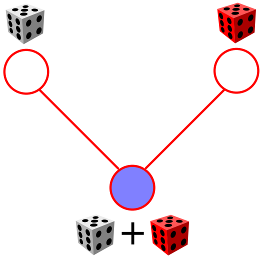
Outline

Graphical models

Probabilistic graphical models describe probability distributions by mapping conditional dependence and independence properties between random variables on a graph structure.

Why again graphical models?

1. Graphical models visualize the structure of a probabilistic model; they help to develop, understand and motivate probabilistic models.
2. Complex computations (e.g., marginalization) can derived efficiently using algorithms exploiting the graph structure.



Conditional independence

Two random variables a and b are conditionally independent given a random variable c if and only if

$$p(a, b \mid c) = p(a \mid c)p(b \mid c)$$

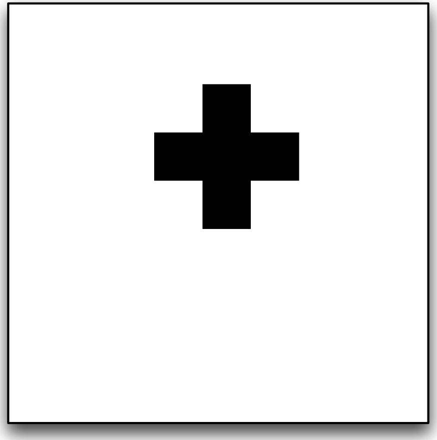
and we write $a \perp\!\!\!\perp b \mid c$.

Conditional independence of sets of variables A, B, C are defined analogously and we write $A \perp\!\!\!\perp B \mid C$.

Hoping to avoid the difficulties of using conditional probability, Thomas Jefferson writes the Declaration of Independence.

Outline

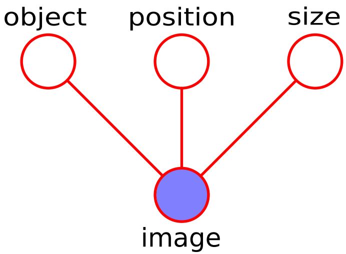
Latent variables: Motivation



200×200 pixels $\quad \rightarrow \quad 2^{40000} - 1$ parameters?

Latent variables

- Additional nodes, which do not directly correspond to observations, allow to describe complex distributions over the visible variables by means of simple conditional distributions.
- The corresponding random variables are called *hidden* or *latent* variables.



Outline

Unsupervised learning

Unsupervised learning means

- learning (important aspects of) a data distribution q,
- finding new *representations* of data that foster learning, generalization, and communication.

Maximum likelihood learning

- Given probabilistic model p with parameters $\boldsymbol{\theta}$ and training data $S = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, ..., \boldsymbol{x}_\ell\}$

- We want to compute/maximize the *likelihood function* given by

$$\mathcal{L}(\boldsymbol{\theta} \mid S) = p(S \mid \boldsymbol{\theta}) = \prod_{n=1}^{\ell} p(\boldsymbol{x}_n \mid \boldsymbol{\theta})$$

or its logarithm, the *log-likelihood*

$$\ln \mathcal{L}(\boldsymbol{\theta} \mid S) = \ln p(S \mid \boldsymbol{\theta}) = \sum_{n=1}^{\ell} \ln p(\boldsymbol{x}_n \mid \boldsymbol{\theta}) \ .$$

Notes on maximum likelihood learning

- For a model p with visible variables $\boldsymbol{X}$ and hidden variables $\boldsymbol{Z}$, the likelihood computation involves

  $$p(\boldsymbol{x}_n \,|\, \boldsymbol{\theta}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}_n, \boldsymbol{z} \,|\, \boldsymbol{\theta}) \ .$$

  This is difficult, especially because of the sum which prevents the logarithm to act directly on the joint distribution.

- If q is the true distribution underlying S, maximizing the logarithmic likelihood function corresponds to minimizing an empirical estimate of the Kullback-Leibler divergence $\mathrm{KL}(q \,\|\, p)$.

Remember: Kullback-Leibler divergence

Kullback-Leibler (KL) divergence between two distribution p and q over $\boldsymbol{X}$ is

- a (non-symmetric) measure of difference between p and q,
- always positive, zero iff the distributions are the same,

and defined as

$$\begin{aligned}\mathrm{KL}(q \,\|\, p) &= -\sum_{\boldsymbol{X}} q(\boldsymbol{X}) \ln \frac{p(\boldsymbol{X})}{q(\boldsymbol{X})} \\ &= \underbrace{-\sum_{\boldsymbol{X}} q(\boldsymbol{X}) \ln p(\boldsymbol{X})}_{\substack{\text{can be} \\ \text{approximated by} \\ \frac{1}{\ell}\sum_{n=1}^{\ell} \ln p(\boldsymbol{x}_n)}} + \underbrace{\sum_{\boldsymbol{X}} q(\boldsymbol{X}) \ln q(\boldsymbol{X})}_{\text{independent of } p}\end{aligned}$$

(sum turns to integral for continuous random variables).

Remember: Gradient-based optimization

- Goal: Maximization of likelihood!
- Consider learning by iteratively changing the parameters:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \Delta\boldsymbol{\theta}^{(t)}$$

- Simplest choice is (steepest) gradient ascent

$$\Delta\boldsymbol{\theta}^{(t)} = \eta \nabla_{\boldsymbol{\theta}} \ln \mathcal{L}(S \,|\, \boldsymbol{\theta}^{(t)})$$

  with learning rate $\eta > 0$.

- Often a *momentum term* is added to improve the performance

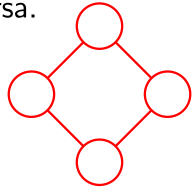
$$\Delta\boldsymbol{\theta}^{(t)} = \eta \ln \mathcal{L}(S \,|\, \boldsymbol{\theta}^{(t)}) + \mu \Delta\boldsymbol{\theta}^{(t-1)}$$

  with momentum parameter $\mu \geq 0$.

Outline

Undirected graphical models

- In undirected graphical models, also referred to as *Markov networks* or *Markov random fields* (MRFs), distributions are mapped to undirected graphs:
  - Random variables correspond to vertices
    (and we do not distinguish between random variable and corresponding vertex to ease the notation)
  - Edges encode conditional independence properties
- MRFs are an alternative to Bayesian networks and factor graphs. Some conditional independence properties can be expressed by MRFs that can not be expressed by Bayesian networks and vice versa.



Independence in Markov networks

Global Markov property

A joint probability distribution for a set of random variables V fulfills the *(global) Markov property w.r.t. an undirected graph*, where each random variable corresponds to a vertex, if for all disjoint sets of random variables $A, B, C \subset V$ it holds:

$A \perp\!\!\!\perp B \,|\, C$ if in the graph every path from a node in A to a node from B contains a node from C.

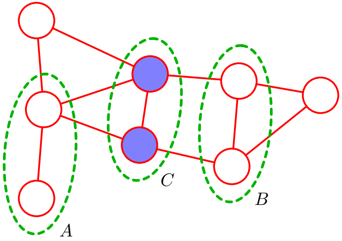
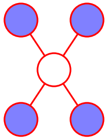


figure taken from Bishop: *Pattern Recognition and Machine Learning*, 2006

Markov blanket

- The set of nodes $\mathrm{MB}(a)$ is the Markov blanket of node a, if for any set of nodes B with $a \notin B$ we have $p(a \,|\, \mathrm{MB}(a), B) = p(a \,|\, \mathrm{MB}(a))$.
- In a Markov network, the Markov blanket $\mathrm{MB}(a)$ of a node a is its set of neighboring nodes:



Cliques

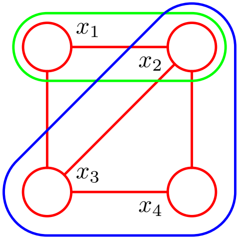


figure taken from Bishop: *Pattern Recognition and Machine Learning*, 2006

- A clique is a subset of nodes in which all nodes are pairwise connected. A clique is maximal if no node can be added such that the resulting set is still a clique.
- Let $\mathcal{C}$ denote the set of all maximal cliques of the undirected model graph.
- Let $\boldsymbol{x}_C$ denote the variables of a clique $C \in \mathcal{C}$. For example, if $C = \{2, 3, 4\}$ and $\boldsymbol{x} = (x_1, x_2, x_3, x_4)^\mathsf{T}$ then $\boldsymbol{x}_C = (x_2, x_3, x_4)^\mathsf{T}$.

Factorization

Factorization over a graph

The distribution p factorizes over an undirected graph G with maximal cliques $\mathcal{C}$ if

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C) \enspace .$$

- The functions $\psi_C(\boldsymbol{x}_C) \geq 0$ are called *potential functions*.
- The normalization constant

$$Z = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

is also called *partition function*.

Hammersley-Clifford Theorem

Hammersley-Clifford theorem

A strictly positive distribution p satisfies the (global) Markov property w.r.t. an undirected graph G if and only if p factorizes over G.

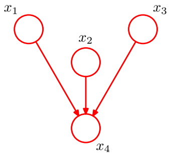
- Because $\psi_C(\boldsymbol{x}_C) > 0$ we can write

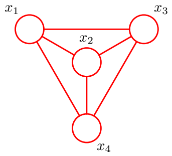
$$p(\boldsymbol{x}) = \frac{1}{Z}\prod_{C\in\mathcal{C}}\psi_C(\boldsymbol{x}_C) = \frac{1}{Z}e^{\sum_{C\in\mathcal{C}}\ln\psi_C(\boldsymbol{x}_C)} = \frac{1}{Z}e^{-E(\boldsymbol{x})}$$

- E is called energy function (the higher the energy of a state the lower its probability).

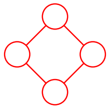
Undirected and directed graphical models

- Moralization: Connecting (marrying) the parents of the nodes in a directed graph gives the moral graph.



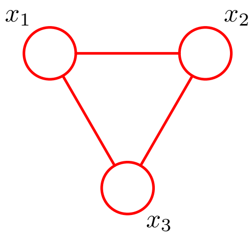


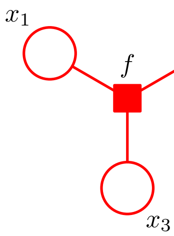
- The distribution described by the directed graph also factorizes over the moral graph, but some information is lost.
- Some conditional independence properties can be expressed by undirected graphical models that can not be expressed by Bayesian networks and vice versa.



Undirected graphical models & factor graphs

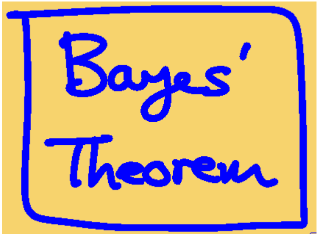
Conversion of an undirected graphical model to a factor graph is straightforward:

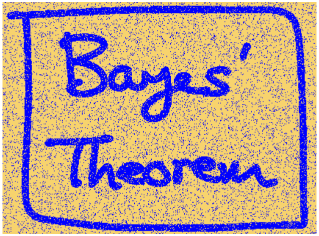




Outline

The problem

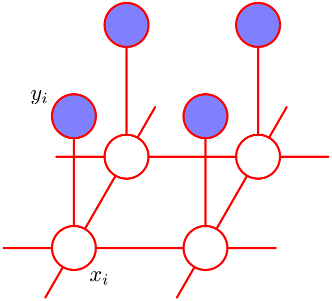




How to remove noise from a (binary) image?

The model

- Binary pixel values $x_i \in \{-1, 1\}$
- Observed pixels y_i have flipped signs with a certain error probability
- Assumption: Neighboring pixels are highly correlated
- Graphical model:



The energy function

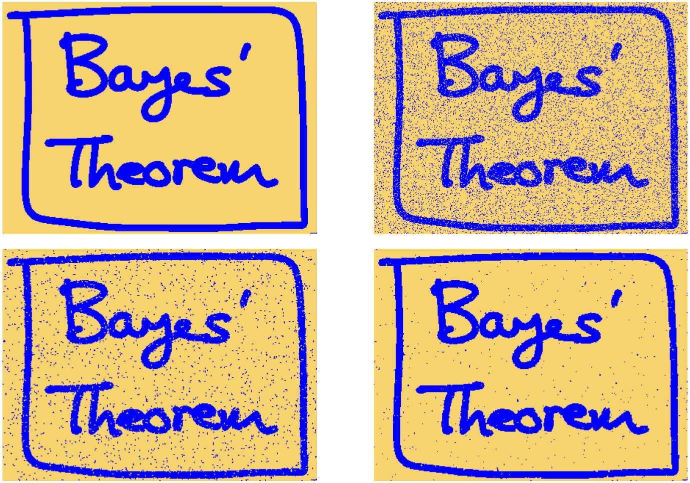
$$
\begin{aligned}
E(\boldsymbol{x},\boldsymbol{y}) &= h\sum_i x_i - \beta \sum_{\text{all edges } (x_i,x_j)} x_i x_j - \eta \sum_i x_i y_i \\
&= -\beta \sum_{\text{all edges } (x_i,x_j)} x_i x_j - \sum_i (\eta x_i y_i - h x_i)
\end{aligned}
$$

- $\eta > 0$: strength of coupling of x_i and y_i
- $\beta > 0$: influence of correlation assumption
- h: optimal bias towards a particular sign
- Colors indicate different types of clique

Algorithm

- Idea: The x_i minimizing the energy for given y_i correspond to the real image
- Optimal x_i can be found by (rather complicated) graph-cut algorithm
- Simple heuristic: Iterated Conditional Modes (ICM)
  - Init all x_i with y_i
  - Repeatedly loop through the x_i, flip single x_i and flip it back if energy does not decrease

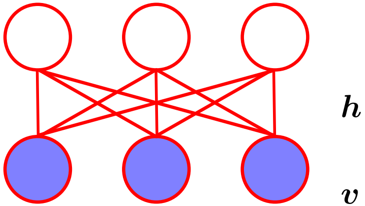
Sample solutions



top: original / corrupted image
bottom: reconstruction using ICM / graph cut

Outline

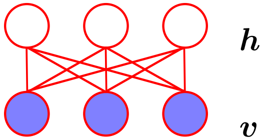
Restricted Boltzmann Machines (RBM)



P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*, pp. 194–281. MIT Press, 1986.

RBM structure and energy function

- m visible binary variables $\boldsymbol{v}$ and n hidden binary variables $\boldsymbol{h}$ form bipartite graph:

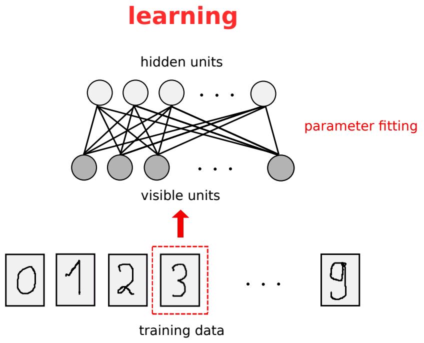


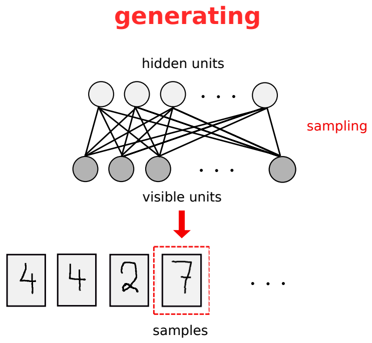
- The energy function reads

$$E(\underbrace{\boldsymbol{v},\boldsymbol{h}}_{\boldsymbol{x}}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij}h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$

with real-valued weights and bias parameters collected in $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c})$.

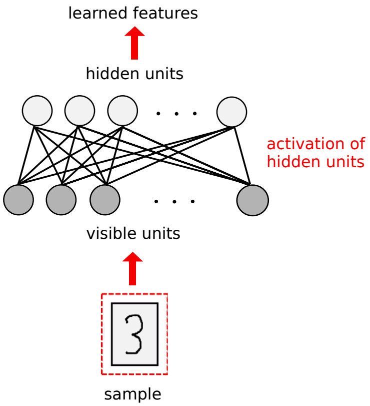
RBM usage I


learning
hidden units
parameter fitting
visible units
training data

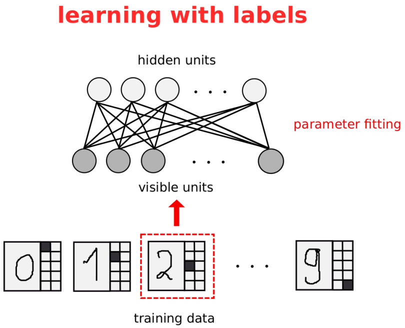

generating
hidden units
sampling
visible units
samples

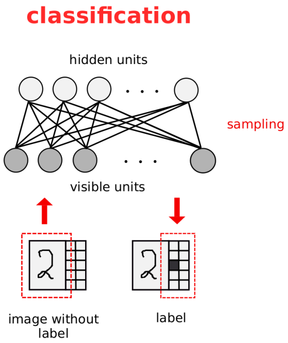
RBM usage II

feature mapping



RBM usage III





Outline

RBMs are stochastic neural networks

- Because of the structure it holds:

$$p(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_{i=1}^{n} p(h_i \mid \boldsymbol{v}) \text{ and } p(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_{i=1}^{m} p(v_i \mid \boldsymbol{h})$$

- The conditional probabilities can be interpreted as neural firing rates, because

$$p(h_i = 1 \mid \boldsymbol{v}) = \sigma\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right)$$

$$p(v_j = 1 \mid \boldsymbol{h}) = \sigma\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right)$$

with activation (aka. transfer of firing) function $\sigma(x) = 1/(1 + e^{-x})$.

Appearance of the sigmoid I

Let $\boldsymbol{v}_{-l}$ be all visible nodes except v_l. We have:

$$p(v_l \,|\, \boldsymbol{v}_{-l}, \boldsymbol{h}) = \frac{p(v_l, \boldsymbol{v}_{-l}, \boldsymbol{h})}{p(\boldsymbol{v}_{-l}, \boldsymbol{h})} = \frac{e^{-E(v_l, \boldsymbol{v}_{-l}, \boldsymbol{h})}}{\sum_{v_l' \in \{0,1\}} e^{-E(v_l', \boldsymbol{v}_{-l}, \boldsymbol{h})}}$$

Let's define:

$$\alpha(\boldsymbol{v}_{-l}, \boldsymbol{h}) = -\sum_{i=1}^{n} w_{il} h_i - b_l$$

$$\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) = -\sum_{i=1}^{n} \sum_{j=1, j\neq l}^{m} w_{ij} h_i v_j - \sum_{j=1, j\neq l}^{m} b_i v_i - \sum_{i=1}^{n} c_i h_i$$

Then $E(\boldsymbol{v}, \boldsymbol{h}) = \beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) + v_l \alpha(\boldsymbol{v}_{-l}, \boldsymbol{h})$, where $v_l \alpha(\boldsymbol{v}_{-l}, \boldsymbol{h})$ are all terms involving v_l.

Appearance of the sigmoid II

$$
\begin{aligned}
&p(v_l = 1 \,|\, \boldsymbol{v}_{-l}, \boldsymbol{h}) \\
&= \frac{\exp(-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 1 \cdot \alpha(\boldsymbol{v}_{-l}, \boldsymbol{h}))}{\exp(-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 1 \cdot \alpha(\boldsymbol{v}_{-l}, \boldsymbol{h})) + \exp(-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}) - 0 \cdot \alpha(\boldsymbol{v}_{-l}, \boldsymbol{h}))} \\
&= \frac{\exp(-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})) \cdot \exp(-\alpha(\boldsymbol{v}_{-l}, \boldsymbol{h}))}{\exp(-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})) \cdot \exp(-\alpha(\boldsymbol{v}_{-l}, \boldsymbol{h})) + \exp(-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h}))} \\
&= \frac{\exp(-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})) \cdot \exp(-\alpha(\boldsymbol{v}_{-l}, \boldsymbol{h}))}{\exp(-\beta(\boldsymbol{v}_{-l}, \boldsymbol{h})) \cdot (\exp(-\alpha(\boldsymbol{v}_{-l}, \boldsymbol{h})) + 1)} \\
&= \frac{\exp(-\alpha(\boldsymbol{v}_{-l}, \boldsymbol{h}))}{\exp(-\alpha(\boldsymbol{v}_{-l}, \boldsymbol{h})) + 1} = \frac{\frac{1}{\exp(\alpha(\boldsymbol{v}_{-l},\boldsymbol{h}))}}{\frac{1}{\exp(\alpha(\boldsymbol{v}_{-l},\boldsymbol{h}))} + 1} \\
&= \frac{1}{1 + \exp(\alpha(\boldsymbol{v}_{-l}, \boldsymbol{h}))} = \sigma(-\alpha(\boldsymbol{v}_{-l}, \boldsymbol{h}))
\end{aligned}
$$

Outline

Sampling from an RBM

Sampling from an RBM can be done using Gibbs sampling, a simple Markov chain Monte Carlo algorithm:

$$
\begin{aligned}
\boldsymbol{h}^{(0)} &\sim p(\boldsymbol{h} \,|\, \boldsymbol{v}^{(0)}) \\
\boldsymbol{v}^{(1)} &\sim p(\boldsymbol{v} \,|\, \boldsymbol{h}^{(0)}) \\
\boldsymbol{h}^{(1)} &\sim p(\boldsymbol{h} \,|\, \boldsymbol{v}^{(1)}) \\
\boldsymbol{v}^{(2)} &\sim p(\boldsymbol{v} \,|\, \boldsymbol{h}^{(1)}) \\
\boldsymbol{h}^{(2)} &\sim p(\boldsymbol{h} \,|\, \boldsymbol{v}^{(2)}) \\
&\dots \\
\boldsymbol{v}^{(k)} &\sim p(\boldsymbol{v} \,|\, \boldsymbol{h}^{(k-1)})
\end{aligned}
$$

Each step is easy since $p(h_i = 1 \,|\, \boldsymbol{v}) = \sigma\big(\sum_{j=1}^{m} w_{ij} v_j + c_i\big)$ and $p(v_j = 1 \,|\, \boldsymbol{h}) = \sigma\big(\sum_{i=1}^{n} w_{ij} h_i + b_j\big)$.

Outline

Learning

- Given i.i.d. training data $S = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_\ell\}$, we want to maximize the logarithmic likelihood

$$\ln \mathcal{L}(\boldsymbol{\theta} \,|\, S) = \ln \prod_{i=1}^{\ell} p(\boldsymbol{v}_i \,|\, \boldsymbol{\theta}) = \sum_{i=1}^{\ell} \ln p(\boldsymbol{v}_i \,|\, \boldsymbol{\theta})$$

with $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c})$.

- We do gradient ascent. It holds

$$\frac{\partial}{\partial \boldsymbol{\theta}} \ln \mathcal{L}(\boldsymbol{\theta} \,|\, S) = \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{i=1}^{\ell} \ln p(\boldsymbol{v}_i \,|\, \boldsymbol{\theta}) = \sum_{i=1}^{\ell} \frac{\partial}{\partial \boldsymbol{\theta}} \ln p(\boldsymbol{v}_i \,|\, \boldsymbol{\theta}) \ .$$

Likelihood gradient I

Log-likelihood of single pattern $\boldsymbol{v}$:

$$\begin{aligned}\ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v}) = \ln p(\boldsymbol{v} \,|\, \boldsymbol{\theta}) &= \ln \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \\ &= \ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} - \ln \sum_{\boldsymbol{v}',\boldsymbol{h}'} e^{-E(\boldsymbol{v}',\boldsymbol{h}')}\end{aligned}$$

Gradient:

$$\begin{aligned}\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \right) - \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{v}',\boldsymbol{h}'} e^{-E(\boldsymbol{v}',\boldsymbol{h}')} \right) \\ &= -\frac{1}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} \\ &\quad + \frac{1}{\sum_{\boldsymbol{v}',\boldsymbol{h}'} e^{-E(\boldsymbol{v}',\boldsymbol{h}')}} \sum_{\boldsymbol{v}',\boldsymbol{h}'} e^{-E(\boldsymbol{v}',\boldsymbol{h}')} \frac{\partial E(\boldsymbol{v}',\boldsymbol{h}')}{\partial \boldsymbol{\theta}}\end{aligned}$$

Likelihood gradient II

$$
\begin{aligned}
- \frac{1}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} \\
+ \frac{1}{\sum_{\boldsymbol{v}',\boldsymbol{h}'} e^{-E(\boldsymbol{v}',\boldsymbol{h}')}} \sum_{\boldsymbol{v}',\boldsymbol{h}'} e^{-E(\boldsymbol{v}',\boldsymbol{h}')} \frac{\partial E(\boldsymbol{v}',\boldsymbol{h}')}{\partial \boldsymbol{\theta}} \\
= - \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v}',\boldsymbol{h}'} p(\boldsymbol{v}',\boldsymbol{h}') \frac{\partial E(\boldsymbol{v}',\boldsymbol{h}')}{\partial \boldsymbol{\theta}} \\
= - \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v}'} p(\boldsymbol{v}') \sum_{\boldsymbol{h}'} p(\boldsymbol{h}' \,|\, \boldsymbol{v}') \frac{\partial E(\boldsymbol{v}',\boldsymbol{h}')}{\partial \boldsymbol{\theta}}
\end{aligned}
$$

Note: $p(\boldsymbol{h} \,|\, \boldsymbol{v}) = \frac{p(\boldsymbol{h},\boldsymbol{v})}{p(\boldsymbol{v})} = \frac{\frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\frac{1}{Z} \sum_{\boldsymbol{h}'} e^{-E(\boldsymbol{v},\boldsymbol{h}')}} = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\sum_{\boldsymbol{h}'} e^{-E(\boldsymbol{v},\boldsymbol{h}')}}$

Gradient with respect to a weight I

For example, for a weight w_{ij}

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial \boldsymbol{\theta}} =$$

$$-\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v}'} p(\boldsymbol{v}') \sum_{\boldsymbol{h}'} p(\boldsymbol{h}' \,|\, \boldsymbol{v}') \frac{\partial E(\boldsymbol{v}', \boldsymbol{h}')}{\partial \boldsymbol{\theta}}$$

is

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{c} \,|\, \boldsymbol{v})}{\partial w_{ij}} = \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j - \sum_{\boldsymbol{v}'} p(\boldsymbol{v}') \sum_{\boldsymbol{h}'} p(\boldsymbol{h}' \,|\, \boldsymbol{v}') h_i' v_j' \ .$$

Sums run over all values the respective variables can take.
This huge computational complexity can be reduced by clever factorization.

Factorization trick

$$
\begin{aligned}
\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j &= \sum_{\boldsymbol{h}} \prod_{k=1}^{n} p(h_k \,|\, \boldsymbol{v}) h_i v_j \\
&= \sum_{h_i} \sum_{\boldsymbol{h}_{-i}} p(h_i \,|\, \boldsymbol{v}) p(\boldsymbol{h}_{-i} \,|\, \boldsymbol{v}) h_i v_j \\
&= \sum_{h_i} p(h_i \,|\, \boldsymbol{v}) h_i v_j \underbrace{\sum_{\boldsymbol{h}_{-i}} p(\boldsymbol{h}_{-i} \,|\, \boldsymbol{v})}_{=1} \\
&= p(h_i = 1 \,|\, \boldsymbol{v}) v_j \\
&= \sigma\left(\sum_{j=1}^{m} w_{ij} v_j + c_i \right) v_j
\end{aligned}
$$

Gradient with respect to a weight II

Thus, for a weight w_{ij}

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial \boldsymbol{\theta}} = \\ -\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v}'} p(\boldsymbol{v}') \sum_{\boldsymbol{h}'} p(\boldsymbol{h}' \,|\, \boldsymbol{v}') \frac{\partial E(\boldsymbol{v}', \boldsymbol{h}')}{\partial \boldsymbol{\theta}}$$

is

$$\begin{aligned} \frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \,|\, \boldsymbol{v})}{\partial w_{ij}} &= p(h_i = 1 \,|\, \boldsymbol{v}) v_j - \sum_{\boldsymbol{v}'} p(\boldsymbol{v}') p(h_i = 1 \,|\, \boldsymbol{v}') v'_j \\ &= p(h_i = 1 \,|\, \boldsymbol{v}) v_j - \mathbb{E}_{p(\boldsymbol{v}')} \left[p(h_i = 1 \,|\, \boldsymbol{v}') v'_j \right] \quad . \end{aligned}$$

Gradients w.r.t. bias parameters can be computed similarly.

Common notation

Often authors write

$$\begin{aligned}\sum_{\boldsymbol{v}\in S}\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}\,|\,\boldsymbol{v})}{\partial w_{ij}} &= \sum_{\boldsymbol{v}\in S}\left[-\mathbb{E}_{p(\boldsymbol{h}\,|\,\boldsymbol{v})}\left[\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}}\right]+\mathbb{E}_{p(\boldsymbol{h}',\boldsymbol{v}')}\left[\frac{\partial E(\boldsymbol{v}',\boldsymbol{h}')}{\partial w_{ij}}\right]\right]\\ &= \sum_{\boldsymbol{v}\in S}\left[\mathbb{E}_{p(\boldsymbol{h}\,|\,\boldsymbol{v})}\left[v_i h_j\right]-\mathbb{E}_{p(\boldsymbol{h}',\boldsymbol{v}')}\left[v'_i h'_j\right]\right]\\ &= \ell\left[\langle v_i h_j\rangle_{p(\boldsymbol{h}\,|\,\boldsymbol{v})p_e(\boldsymbol{v})}-\langle v_i h_j\rangle_{p(\boldsymbol{h},\boldsymbol{v})}\right]\end{aligned}$$

with p_e denoting the empirical distribution. This gives the rule:

$$\sum_{\boldsymbol{v}\in S}\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}\,|\,\boldsymbol{v})}{\partial w_{ij}} \propto \langle v_i h_j\rangle_{\text{data}}-\langle v_i h_j\rangle_{\text{model}}$$

Still too complex. . .

In summary, for an RBM it holds

$$\frac{\partial \ln p(\boldsymbol{v} \,|\, \boldsymbol{\theta})}{\partial w_{ij}} = \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j - \underbrace{\sum_{\boldsymbol{v}'} p(\boldsymbol{v}') \sum_{\boldsymbol{h}'} p(\boldsymbol{h}' \,|\, \boldsymbol{v}') h'_i v'_j}_{\text{still exponential many terms}}$$

with

$$\sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j = \sigma\left(\sum_{j=1}^{m} w_{ij} v_j + c_i \right) v_j \quad .$$

If $m > n$ we factorize differently – why and how?

Contrastive Divergence Learning

- Approximation of the gradient by sample $\boldsymbol{v}'$ generated by Gibbs sampling:

$$\frac{\partial \ln p(\boldsymbol{v} \,|\, \boldsymbol{\theta})}{\partial w_{ij}} \approx \sum_{\boldsymbol{h}} p(\boldsymbol{h} \,|\, \boldsymbol{v}) h_i v_j - \sum_{\boldsymbol{h}'} p(\boldsymbol{h}' \,|\, \boldsymbol{v}') h'_i v'_j$$

- In k-step *Contrastive Divergence* learning the Markov chain in the Gibbs sampling procedure is iterated k steps – and often $k = 1$.
- Contrastive Divergence learning can diverge, because the approximation can be bad if the mixing rate of the chain is low.

Contrastive Divergence Learning Algorithm

Algorithm 1: k-step Contrastive Divergence

Input: RBM $(V_1, \dots, V_m, H_1, \dots, H_n)$, training set S
Output: gradient approximation Δw_{ij}, Δb_j and Δc_i for $i = 1, \dots, n$, $j = 1, \dots, m$

```
1  init $\Delta w_{ij} = \Delta b_j = \Delta c_i = 0$ for $i = 1, \dots, n$, $j = 1, \dots, m$
2  forall the $\boldsymbol{v} \in S$ do
3      $\boldsymbol{v}^{(0)} \leftarrow \boldsymbol{v}$
4      for $t = 0, \dots, k-1$ do
5          for $i = 1, \dots, n$ do sample $h_i^{(t)} \sim p(h_i \mid \boldsymbol{v}^{(t)})$
6
7          for $j = 1, \dots, m$ do sample $v_j^{(t+1)} \sim p(v_j \mid \boldsymbol{h}^{(t)})$
8
9      for $i = 1, \dots, n$, $j = 1, \dots, m$ do
10         $\Delta w_{ij} \leftarrow \Delta w_{ij} + p(h_i = 1 \mid \boldsymbol{v}^{(0)}) \cdot v_j^{(0)} - p(h_i = 1 \mid \boldsymbol{v}^{(k)}) \cdot v_j^{(k)}$
11         $\Delta b_j \leftarrow \Delta b_j + v_j^{(0)} - v_j^{(k)}$
12         $\Delta c_i \leftarrow \Delta c_i + p(h_i = 1 \mid \boldsymbol{v}^{(0)}) - p(h_i = 1 \mid \boldsymbol{v}^{(k)})$
```

Sampling or not?

For hidden units:

- The final update use probabilities instead of samples to avoid unnecessary sampling noise.
- All other steps rely on samples.

Heuristic for visible units:

- “Assuming the visible units use the logistic function, use real-valued probabilities for both the data and the reconstructions” in 1-step CD learning (in contrast to samples as in the pseudo code).

G. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. Technical Report *UTML TR 2010–003*, Department of Computer Science, University of Toronto, 2010

CD approximates log-likelihood gradient

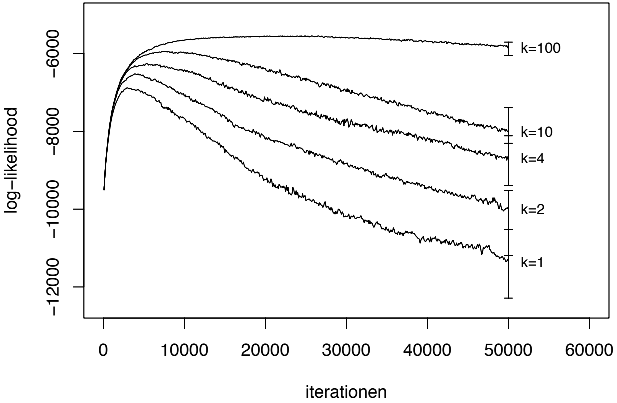
Theorem (Bengio & Dellalleau, Neural Comput. 21, 2009)

For a converging Gibbs chain $\boldsymbol{v}^{(0)} \Rightarrow \boldsymbol{h}^{(0)} \Rightarrow \boldsymbol{v}^{(1)} \Rightarrow \boldsymbol{h}^{(1)} \dots$ *starting at data point* $\boldsymbol{v}^{(0)}$*, it holds*

$$
\begin{aligned}
\frac{\partial}{\partial\boldsymbol{\theta}} \ln p(\boldsymbol{v}^{(0)}) = -\sum_{\boldsymbol{h}^{(0)}} p(\boldsymbol{h}^{(0)} \mid \boldsymbol{v}^{(0)}) \frac{\partial E(\boldsymbol{v}^{(0)}, \boldsymbol{h}^{(0)})}{\partial\boldsymbol{\theta}} \\
+ \mathbb{E}_{p(\boldsymbol{v}^{(k)} \mid \boldsymbol{v}^{(0)})}\left[\sum_{\boldsymbol{h}^{(k)}} p(\boldsymbol{h}^{(k)} \mid \boldsymbol{v}^{(k)}) \frac{\partial E(\boldsymbol{v}^{(k)}, \boldsymbol{h}^{(k)})}{\partial\boldsymbol{\theta}}\right] \\
+ \mathbb{E}_{p(\boldsymbol{v}^{(k)} \mid \boldsymbol{v}^{(0)})}\left[\frac{\partial \ln p(\boldsymbol{v}^{(k)})}{\partial\boldsymbol{\theta}}\right]
\end{aligned}
$$

and the final term converges to zero as k *goes to infinity.*

Example of divergence



A. Fischer & C. Igel. Empirical Analysis of the Divergence of Gibbs Sampling Based Learning Algorithms for Restricted Boltzmann Machines. *ICANN 2010*, LNCS 6354, pp. 208–217, Springer, 2010

Bounding CD bias I

Theorem (Fischer & Igel, Neural Comput. 23, 2011)

Given an RBM with m visible and n hidden neurons and joint probability distribution p. Let p_e be the empirical distribution defined by a set of samples $\boldsymbol{v}_1, \dots, \boldsymbol{v}_\ell$. Then an upper bound on the expectation of the error of the CD-k approximation of the log-likelihood gradient is given by

$$\left|\mathbb{E}_{p_e(\boldsymbol{v}^{(0)})}\left[\mathbb{E}_{p(\boldsymbol{v}^{(k)}|\boldsymbol{v}^{(0)})}\left[\frac{\partial \ln p(\boldsymbol{v}^{(k)})}{\partial \theta}\right]\right]\right| \leq \frac{1}{2}\left|\mu - p\right|\left(1 - e^{-(m+n)\Delta}\right)^k$$

[...]

Bounding CD bias II

[...] with a start distribution $\mu(\boldsymbol{v}, \boldsymbol{h}) = p(\boldsymbol{h} \,|\, \boldsymbol{v}) p_e(\boldsymbol{v})$ for the initial states of the chain and

$$\Delta = \max \left\{ \max_{l \in \{1,\dots,m\}} \vartheta_l, \max_{l \in \{1,\dots,n\}} \xi_l \right\} \quad ,$$

where (indicator function $\mathbb{I}_{\{A\}}$ is 1 if A is true and 0 otherwise)

$$\vartheta_l = \max \left\{ \left| \sum_{i=1}^{n} \mathbb{I}_{\{w_{il}>0\}} w_{il} + b_l \right|, \left| \sum_{i=1}^{n} \mathbb{I}_{\{w_{il}<0\}} w_{il} + b_l \right| \right\}$$

and

$$\xi_l = \max \left\{ \left| \sum_{j=1}^{m} \mathbb{I}_{\{w_{lj}>0\}} w_{lj} + c_l \right|, \left| \sum_{j=1}^{m} \mathbb{I}_{\{w_{lj}<0\}} w_{lj} + c_l \right| \right\} \quad .$$

Parallel Tempering

- Divergence is less prominent if parallel tempering (PT) is used for sampling.
- Parallel tempering introduces parallel *replica* Markov chains to foster faster mixing.
- That's what I would suggest to use (in combination with *flip-the-state* operator).

G. Desjardins, A. C. Courville, Y. Bengio, P. Vincent, and O. Delalleau. Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 145-152, 2010.

Outline

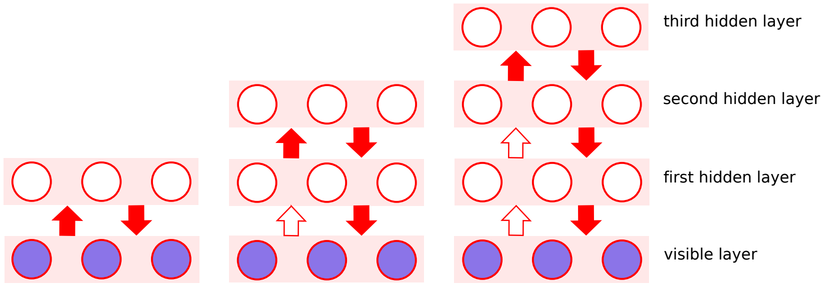
Motivation

`http://www.cs.toronto.edu/~hinton/adi/index.htm`

G. E. Hinton & R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science 313(5786):504–507, 2006

- A *deep belief network* (DBN) is a layered stochastic neural network composed of stacked RBMs.
- The idea is to learn features from features.
- DBNs are initially trained in a layer-wise fashion. Parameters of previous layes are fixed.
- Only the final layer remains fully stochastic. Previous layers turn to deterministic mappings for the "forward" path (from the visible units to the upper hidden layers), but are used stochastically when generating input pattern "backwards".

Schema of incremental training



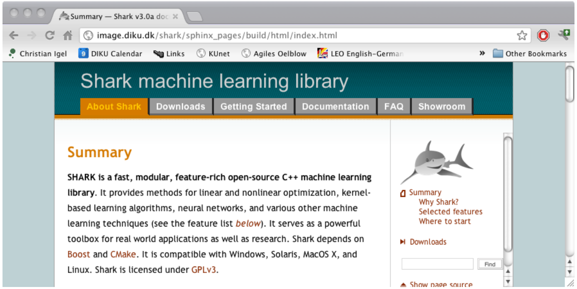
Summary

- Markov random fields (aka. undirected graphical models) play an important role in computer vision and pattern recognition, for instance for
  - image denoising and inpainting,
  - image segmentation,
  - object localization and recognition.
- Restricted Boltzmann Machines (RBM) are
  - models of stochastic neural networks,
  - undirected graphical models, and
  - building blocks of deep belief networks.
- The partition function makes computing the likelihood and its derivatives challenging.

Software

`image.diku.dk/shark`

Igel, Glasmachers, Heidrich-Meisner: Shark, *Journal of Machine Learning Research* 9:993–996, 2008



Summary

SHARK is a fast, modular, feature-rich open-source C++ machine learning library. It provides methods for linear and nonlinear optimization, kernel-based learning algorithms, neural networks, and various other machine learning techniques (see the feature list *below*). It serves as a powerful toolbox for real world applications as well as research. Shark depends on Boost and CMake. It is compatible with Windows, Solaris, MacOS X, and Linux. Shark is licensed under GPLv3.

Gold Prize at Open Source Software Wold Challenge 2011

More software and tutorials: `http://deeplearning.net`

References

Basic:

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006 (many images on the slides are from this book)

Y. Bengio. Learning Deep Architectures for AI, *Foundations and Trends in Machine Learning* 2(1): 1–127, 2009

A. Fischer and C. Igel. Training Restricted Boltzmann Machines: An Introduction. *Pattern Recognition*, 7:25-39, 2014

Advanced books:

S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996 (e.g., proof of Hammersley-Clifford theorem)

D. Koller & N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009 (very good and comprehensive book)

References to Applications I

Image classification, processing, and generation:

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Y. Tang, R. Salakhutdinov, and G. E. Hinton. Robust Boltzmann machines for recognition and denoising. In *CVPR*, pages 2264–2271. IEEE, 2012.

N. Le Roux, N. Heess, J. Shotton, and J. M. Winn. Learning a generative model of images by factoring appearance and shape. *Neural Computation*, 23(3):593–650, 2011.

J. Kivinen and C. Williams. Multiple texture Boltzmann machines. *JMLR W&CP: AISTATS 2012*, 22:638–646, 2012.

T. Schmah, G. E. Hinton, R. S. Zemel, S. L. Small, and S. C. Strother. Generative versus discriminative training of RBMs for classification of fMRI images. *NIPS 21*, pages 1409–1416, 2009.

H. Larochelle and Y. Bengio. Classification using discriminative restricted Boltzmann machines. *ICML*, pages 536–543. ACM, 2008.

References to Applications II

Learning movement patterns:

G. W. Taylor, E. H. G, and S. Roweis. Modeling human motion using binary latent variables. *NIPS 19*, pp. 1345–1352. MIT Press, 2007.

G. W. Taylor and G. E. Hinton. Factored conditional restricted Boltzmann Machines for modeling motion style. *ICML*, pp. 1025–1032. ACM, 2009.

A acoustic modeling:

A. Mohamed and G. E. Hinton. Phone recognition using restricted Boltzmann machines. *ICASSP*, pp. 4354–4357. IEEE Press, 2010.

References to Applications III

Collaborative filtering for movie recommendations:

R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted Boltzmann machines for collaborative filtering. *ICML*, pp. 791–798. ACM, 2007.

Extraction of semantic document representations:

R. Salakhutdinov and G. E. Hinton. Replicated softmax: an undirected topic model. *NIPS 22*, pp. 1607–1614, 2009.

P. V. Gehler, A. D. Holub, and M. Welling. The rate adapting poisson model for information retrieval and object recognition. *ICML*, pp. 337–344. ACM, 2006.

E. P. Xing, R. Yan, and A. G. Hauptmann. Mining associated text and images with dual-wing harmoniums. *UAI*. AUAI Press, 2005.