Restricted Boltzmann Machines for Collaborative Filtering

paper by Ruslan Salakhutdinov, Andriy Mnih, Geoffrey Hinton

presentation by Andreas Töschener

1. Introduction

2. Restricted Boltzmann Machines

3. The Netflix Prize

4. RBM for the Netflix Prize
Collaborative Filtering (CF)
- collects taste information from many users (collaborative)
- uses this information to make predictions (filtering)
- also known as Recommender Systems

online shops are very interested in good CF algorithms
there are many standard algorithms for CF
the use of RBM for CF is very new and promising
Restricted Boltzmann Machines
Introduction

A Boltzmann machine can be seen as a recurrent neural network with probabilistic units.

A RBM is a special case of a Boltzmann Machine.

Only connections between visible and hidden units.

Much faster training than normal Boltzmann Machines.

The goal of a RBM is to learn the distribution $P(V)$. 
Let $v$ and $h$ be binary variables.

- $P(h_j = 1|v) = \sigma(b_j + \sum_i w_{ij}v_i)$
- $P(v_i = 1|h) = \sigma(a_i + \sum_j w_{ij}h_j)$
- $\sigma(z) = \frac{1}{1 + \exp(-z)}$
Trainingsalgorithm

\[ D_{KL}(P_{data}(v) \| P_{model}(v)) \rightarrow \min \]

log-likelihood solution:
- \( \Delta W_{ij} = \epsilon (\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}) \)
- \( \Delta a_i = \epsilon (\langle v_i \rangle_{data} - \langle v_i \rangle_{model}) \)
- \( \Delta b_j = \epsilon (\langle h_j \rangle_{data} - \langle h_j \rangle_{model}) \)

- use gradient ascent with the above gradients
Contrastive Divergence solution:

- \( \Delta W_{ij} = \epsilon \left( \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_T \right) \)
- \( \Delta a_i = \epsilon \left( \langle v_i \rangle_{data} - \langle v_i \rangle_T \right) \)
- \( \Delta b_j = \epsilon \left( \langle h_j \rangle_{data} - \langle h_j \rangle_T \right) \)
The Netflix Prize
Netflix is a big online DVD rental shop
October 2006, start of a $1 Mio. challenge
the goal of the challenge is to improve the performance of their recommender system by 10%
at the moment there are over 23,000 registered teams from 164 different countries [www.netflixprize.com/leaderboard]
currently Jährer Michael and I are on place 67
Details

- Netflix provides over 100 Mio. ratings
  - 500,000 users
  - 18,000 movies
- the ratings are discrete and range from 1 to 5
- there are no additional informations like:
  - movie genre
  - actors
  - director
  - interests of the users
  - ...
Measuring the Performance

- the performance of an algorithm is measured with the RMSE (root mean square error) on the qualifying set (2.8 Mio. ratings)
- Netflix own recommender system archives an RMSE of 0.9514
- the goal of the contest is an improvement of 10%, that corresponds to an RMSE of 0.8563
- a single RBM can archive an RMSE of 0.9100
RBM for the Netflix Prize
Introduction
Restricted Boltzmann Machines
The Netflix Prize
RBM for the Netflix Prize

- one RBM for each user
- each RBM has only one training case
- each RBM has the same number of hidden units
- the weights are shared between all RBMs
use a conditional Bernoulli distribution for the visible units

\[
P(V_i^k = 1 | \mathbf{h}) = \frac{\exp(a_i^k + \sum_{j=1}^{F} h_j W_{ij}^k)}{\sum_{l=1}^{5} \exp(b_l^i + \sum_{j=1}^{F} h_j W_{ij}^l)}
\]

\[
P(h_j = 1 | \mathbf{V}) = \sigma(b_j + \sum_{i=1}^{m} \sum_{k=1}^{5} V_i^k W_{ij}^k)
\]

\[
\sigma(x) = \frac{1}{1 + \exp(-x)}
\]
Contrastive Divergence Learning

- \( \Delta W^k_{ij} = \epsilon (\langle v^k_i h_j \rangle_{data} - \langle v^k_i h_j \rangle_T) \)
- \( \Delta a^k_i = \epsilon (\langle v^k_i \rangle_{data} - \langle v^k_i \rangle_T) \)
- \( \Delta b_j = \epsilon (\langle h_j \rangle_{data} - \langle h_j \rangle_T) \)
Making predictions

1st Method:
- run the RBM for a long time
- take the expected rating

2nd Method:
- init the visible units to the given ratings
- calc the probabilities for the hidden units (don't apply the sampling step)
- calc the probabilities for the visible units, based on the probabilities of the hidden units
- this method is much faster than the 1st one
there is an additional source of information in the data: user × movie pairs with unknown rating

this is a very valuable information for users with few ratings

the usage of this information shows good results for this contest

the practical usage is debatable
Conditional RBM

\[ P(h_j = 1|V, r) = \sigma(b_j + \sum_{i=1}^{m} \sum_{k=1}^{5} V_i^k W_{ij}^k + \sum_{i=1}^{M} r_i D_{ij}) \]

\[ \Delta D_{ij} = \epsilon(\langle h_j \rangle_{data} - \langle h_j \rangle_T) r_i \]
Conditional RBM vs. RBM

- the RBM and CRBM use 100 hidden units
- my implementation needs 30 minutes for one epoch with $T=1$ on a 3.6 GHz Intel Core 2
Conditional Factored RBM

- a lot of hidden units $\implies$ very big weight matrix
- the solution is a factorization of the weight matrix
- $W_{ij}^k = \sum_{c=1}^{C} A_{ic}^k B_{cj}$

example:
- 18,000 visible units, 5 ratings, 500 hidden units $\implies$ 45 Mio. trainable weights
- with a factorization of $W$, with $C=30$ $\implies$ 2.7 Mio. trainable weights
the Conditional RBM uses 100 hidden units
the Conditional Factored RBM uses 500 hidden units and $C=30$
normal RBMs are easy to implement
these RBMs with weight sharing, are more tricky to implement
the biggest problem on the netflix dataset is the big dataset itself
the calculation on the visible and hidden units can be done in parallel
use SMP-Machines, GPU, ...
on Intel machines the use of modern multicore processors gives no speedup (here the limiting factor is the memory bandwidth)
all experiments are done on a 3.6 GHz Intel Core 2 with 4 GB RAM

CRBM with 50 hidden units
- $T=1$, 27 min per epoch
- $T=3$, 43 min per epoch
- $T=5$, 57 min per epoch

CRBM with 100 hidden units
- $T=1$, 74 min per epoch
a RBM is a good method to automatical model a data distribution

RBMs show really good results for CF

computational very expensive

speedup through parallization is posible
Thank you for your attention

References:

- Restricted Boltzmann Machines for Collaborative Filtering, by Ruslan Salakhutdinov, Andrity Mnih, Geoffrey Hinton