Training deep Autoencoders for collaborative filtering

Oleksii Kuchaiev & Boris Ginsburg
Motivation

Personalized recommendations
Key points (spoiler alert)

1. Deep autoencoder for collaborative filtering
   1. Improves generalization

2. Right activation function (SELU, ELU, LeakyRELU) enables deep architectures
   1. No layer-wise pre-training, or skip connections

3. Heavy use of dropout

4. *Dense re-feeding* for faster and better training

5. Beats other models on time-split Netflix data (RMSE of 0.9099 vs 0.9224)


Autoencoders & collaborative filtering

Effects of the activation types
Overfitting the data
Going deeper
Dropout
Dense re-feeding
Conclusions

Collaborative filtering

Rating prediction

\[ R(i, j) = k \text{ iff user } i \text{ gave item } j \text{ rating } k \]

\( m \) users

\( n \) items

\( r \) hidden factors

- Some of the most popular approaches:
  - Alternative Least Squares (ALS)

\( R = \text{Rating matrix} \)

\( X \)

\( r \) items
Autoencoder

Deep learning tool of choice for dimensionality reduction

\[ z = e_2 \]
\[ d_1 = f(W^2_d * z + b_3) \]
\[ d_2 = W^1_d * d_1 + b_4 \]

\[ e_1 = f(W^1_e * x + b_1) \]
\[ e_2 = f(W^2_e * e_1 + b_2) \]

\[ y = \text{decoder}(z), \text{reconstruction of } x \]
\[ y = \text{decoder}(\text{encoder}(x)) \]

Autoencoder can be thought of as generalization of PCA

“Constrained” if decoder weights are transpose of encoder
“De-noising” if noise is added to \( x \).

AutoEncoders for recommendations

User (item) based

Masked Mean Squared Error

$$MMSE = \frac{\sum_{i=0}^{n} m_i \cdot (r_i - y_i)^2}{\sum_{i=0}^{n} m_i}$$

$$m_i = 1 \text{ if } r_i \neq 0 \text{ else } m_i = 0.$$
# Dataset

**Netflix prize training data set**

Time split to predict *future* ratings

<table>
<thead>
<tr>
<th></th>
<th>Full</th>
<th>3 months</th>
<th>6 months</th>
<th>1 year</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training</strong></td>
<td>12/99-11/05</td>
<td>09/05-11/05</td>
<td>06/05-11/05</td>
<td>06/04-05/05</td>
</tr>
<tr>
<td>Users</td>
<td>477,412</td>
<td>311,315</td>
<td>390,795</td>
<td>345,855</td>
</tr>
<tr>
<td>Ratings</td>
<td>98,074,901</td>
<td>13,675,402</td>
<td>29,179,009</td>
<td>41,451,832</td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td>12/05</td>
<td>12/05</td>
<td>12/05</td>
<td>06/05</td>
</tr>
<tr>
<td>Users</td>
<td>173,482</td>
<td>160,906</td>
<td>169,541</td>
<td>197,951</td>
</tr>
<tr>
<td>Ratings</td>
<td>2,250,481</td>
<td>2,082,559</td>
<td>2,175,535</td>
<td>3,888,684</td>
</tr>
</tbody>
</table>

### Benchmark

**Netflix prize training data set**

\[
RMSE = \sqrt{\frac{\sum_{r_i \neq 0} (r_i - y_i)^2}{\sum_{r_i \neq 0} 1}}
\]

<table>
<thead>
<tr>
<th></th>
<th>PMF</th>
<th>I-AR</th>
<th>U-AR</th>
<th>T-SVD</th>
<th>RRN</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDb</td>
<td>2.3913</td>
<td>2.0521</td>
<td>2.0290</td>
<td>2.0037</td>
<td>1.9703</td>
</tr>
<tr>
<td>Netflix 6 months</td>
<td>0.9584</td>
<td>0.9778</td>
<td>0.9836</td>
<td>0.9589</td>
<td>0.9427</td>
</tr>
<tr>
<td>Netflix full</td>
<td>0.9252</td>
<td>0.9364</td>
<td>0.9647</td>
<td>0.9275</td>
<td>0.9224</td>
</tr>
</tbody>
</table>


Autoencoders & collaborative filtering

Effects of the activation types

Overfitting the data

Going deeper

Dropout

Dense re-feeding

Conclusions

Activation function matters
- We found that on this task ELU, SELU and LRELU perform much better than SIGMOID, RELU, RELU6, TANH and SWISH

Apparently important:
  a) non-zero negative part
  b) Unbounded positive part

Training RMSE per mini-batch. All lines correspond to 4-layers autoencoder (2 layer encoder and 2 layer decoder) with hidden unit dimensions of 128. Different line colors correspond to different activation functions.
Autoencoders & collaborative filtering
Effects of the activation types
Overfitting the data
Going deeper
Dropout
Dense re-feeding
Conclusions

Overfit your data

Wide layers generalize poorly

\[ y \]
\[ \begin{array}{cccc}
  \bullet & \bullet & \ldots & \bullet \\
\end{array} \]

\[ d_2 = W^1_d * d_1 + b_4 \]

\[ d \]
\[ \begin{array}{cccc}
  \bullet & \bullet & \ldots & \bullet \\
\end{array} \]

\[ e_1 = f(W^1_e * x + b_1) \]

\[ x \]
\[ \begin{array}{cccc}
  \bullet & \bullet & \ldots & \bullet \\
\end{array} \]

Evaluation RMSE > 1.1 on Netflix full
Autoencoders & collaborative filtering

Effects of the activation types

Overfitting the data

Going deeper

Dropout

Dense re-feeding

Conclusions

Deeper models
Generalize better

<table>
<thead>
<tr>
<th>Number of layers</th>
<th>Evaluation RMSE</th>
<th>params</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.146</td>
<td>4,566,504</td>
</tr>
<tr>
<td>4</td>
<td>0.9615</td>
<td>4,599,528</td>
</tr>
<tr>
<td>6</td>
<td>0.9378</td>
<td>4,632,552</td>
</tr>
<tr>
<td>8</td>
<td>0.9364</td>
<td>4,665,576</td>
</tr>
<tr>
<td>10</td>
<td>0.9340</td>
<td>4,698,600</td>
</tr>
<tr>
<td>12</td>
<td>0.9328</td>
<td>4,731,624</td>
</tr>
</tbody>
</table>

No layer-wise pre-training necessary!
Autoencoders & collaborative filtering
Effects of the activation types
Overfitting the data
Going deeper
**Dropout**
Dense re-feeding
Conclusions
Dropout

Helps wider models generalize

Evaluation RMSE

Dropout

512

512

dropout

1024

512

512

RMSE

0.91

0.93

0.95

0.97

1.01

1.03

1.05

1.07

1.09

0

20

40

60

80

100

Epoch
Autoencoders & collaborative filtering
Effects of the activation types
Overfitting the data
Going deeper
Dropout
Dense re-feeding
Conclusions

Dense re-feeding

Note that $x$ is sparse but $f(x)$ is dense.

For $x$, most of the loss is masked.

\[
\text{MMSE} = \frac{m_i \ast (r_i - y_i)^2}{\sum_{i=0}^{n} m_i}
\]

Imagine perfect $f$:

\[
\forall x_i \neq 0: f(x)_i = x_i
\]

If user later rates new item $k$ with rating $r$, then:

\[
f(x)_k = r
\]

By induction:

\[
f(f(x)) = f(x)
\]

Thus, $f(x)$ should be a fixed point of $f$ for every valid $x$.

Intuition: idealized scenario
Dense re-feeding

Attempt to enforce fixed point constraint

(very) sparse $x$ → Dense $f(x)$ → Dense $f(x)$ → Dense $f(f(x))$

Update with real data $x$

Update with synthetic data $f(x)$
Dense re-feeding
Together with bigger LR improves generalization
Results

Netflix time split data

<table>
<thead>
<tr>
<th>DataSet</th>
<th>I-AR</th>
<th>U-AR</th>
<th>RRN</th>
<th>DeepRec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netflix 3 months</td>
<td>0.9778</td>
<td>0.9836</td>
<td>0.9427</td>
<td>0.9373</td>
</tr>
<tr>
<td>Netflix Full</td>
<td>0.9364</td>
<td>0.9647</td>
<td>0.9224</td>
<td>0.9099</td>
</tr>
</tbody>
</table>


DeepRec is our 6 layer model
Conclusions

1. Autoencoders can replace ALS and be competitive with other methods
2. Deeper models generalize better
   1. No layer-wise pre-training is necessary
3. Right activation function enables deep architectures
   1. Negative parts are important
   2. Unbounded positive part
4. Heavy use of dropout is needed for wider models
5. Dense *re-feeding* further improves generalization

Code, docs and tutorial: https://github.com/NVIDIA/DeepRecommender