Deep Learning for Recommender Systems

Nick Pentreath
Principal Engineer

@MLnick
About

@MLnick on Twitter & Github

Principal Engineer, IBM

Spark Technology Center & Cognitive Open
Technologies

Machine Learning & AI

Apache Spark committer & PMC

Author of *Machine Learning with Spark*

Various conferences & meetups
Agenda

Recommender systems overview

Deep learning overview

Deep learning for recommendation

Challenges and future directions
Recommender Systems
Recommender Systems

Users and Items

```
{
    "user_id": "1",
    "name": "Joe Bloggs",
    "created_date": 1476884080,
    "updated_date": 1476946916,
    "last_active_date": 1476946962,
    "age": 32,
    "country": "US",
    "city": "New York",
    ...
}
```

```
{
    "item_id": "10",
    "name": "LOL Cats",
    "description": "catstasticats",
    "category": ["Cat Videos", "Humour", "Animals"],
    "tags": ["cat", "lol", "funny", "cats", "felines"],
    "created_date": 1476884080,
    "updated_date": 1476884080,
    "last_played_date": 1476946962,
    "likes": 100000,
    "author_id": "321",
    "author_name": "ilikecats",
    "channel_id": "CatVideoCentral",
    ...
}
```
Events

Implicit preference data
- Online – page view, click, app interaction
- Commerce – cart, purchase, return
- Media – preview, watch, listen

Explicit preference data
- Ratings, reviews

Intent
- Search query

Social
- Like, share, follow, unfollow, block
Context

```
{
  "user_id": "1",
  "item_id": "10",
  "event_type": "page_view",
  "timestamp": 1476884080,
  "referrer": "http://spark.tc",
  "ip": "123.12.12.12",
  "device_type": "Smartphone",
  "user_agent_os": "Android",
  "user_agent_type": "Mobile Browser",
  "user_agent_family": "Chrome Mobile",
  "geo": 
    "-73.935, 40.731"
}
```
Prediction

Prediction is ranking

- Given a user and context, rank the available items in order of likelihood that the user will interact with them.
Deep Learning
Deep Learning

Overview

Original theory from 1940s; computer models originated around 1960s; fell out of favor in 1980s/90s

Recent resurgence due to

- Bigger (and better) data; standard datasets (e.g. ImageNet)
- Better hardware (GPUs)
- Improvements to algorithms, architectures and optimization

Leading to new state-of-the-art results in computer vision (images and video); speech/text; language translation and more

Modern Neural Networks

Deep (multi-layer) networks

Computer vision
- Convolution neural networks (CNNs)
- Image classification, object detection, segmentation

Sequences and time-series
- Recurrent neural networks (RNNs)
- Machine translation, text generation
- LSTMs, GRUs

Embeddings
- Text, categorical features

Deep learning frameworks
- Flexibility, computation graphs, auto-differentiation, GPUs

Source: Stanford CS231n
Deep Learning for Recommendations
Deep Learning for Recommendations

Evolution of Recommendation Models

- Netflix prize created significant progress (not unlike ImageNet)
- Matrix factorization techniques became SotA
- Restricted Boltzmann Machine (a class of NN) was one of the strongest single models
- Mostly tweaks until introduction of Factorization Machines
- Initial work on applying DL focused on feature extraction for content
- Huge momentum in DL techniques over the past two years

Scalable models
Amazon's item-to-item collaborative filtering

Factorization Machines
General factor models

Gaining momentum
DLRS Workshops @ RecSys Conferences

<table>
<thead>
<tr>
<th>Year</th>
<th>Netflix Prize</th>
<th>Deep content-based models</th>
</tr>
</thead>
<tbody>
<tr>
<td>2003</td>
<td>The rise of matrix factorization</td>
<td>Music recommendation</td>
</tr>
<tr>
<td>2006-2009</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2013</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2016/2017</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Prelude: Matrix Factorization

The *de facto* standard model

- Represent user ratings as a user-item matrix
- Find two smaller matrices (called the *factor* matrices) that approximate the full matrix
- Minimize the reconstruction error (i.e. rating prediction / completion)

- Efficient, scalable algorithms
  - Gradient Descent
  - Alternating Least Squares (ALS)
- Prediction is simple
- Can handle implicit data
Deep Learning for Recommendations

Prelude: Matrix Factorization

Can be re-created easily in Deep Learning frameworks, e.g. Keras

– Use raw weight matrices in low-level frameworks
– Or use embeddings for user and item identifiers
– Loss function over dot product
– Can add user and item biases using single-dimensional embeddings
– What about implicit data?

```python
from keras.layers import Embedding, Dot, Flatten, Input
from keras.models import Model

user_input = Input((1, ))
item_input = Input((1, ))
user_embedding = Flatten()(Embedding(num_users, k, input_length=1)(user_input))
item_embedding = Flatten()(Embedding(num_items, k, input_length=1)(item_input))
loss = Dot(axes=1)([user_embedding, item_embedding])

model = Model([user_input, item_input], loss)
model.compile(loss="mse", optimizer="adam")
```
Deep Learning for Recommendations

Matrix Factorization for Implicit Data

Flexibility of deep learning frameworks means we can deal with implicit data in many ways

- Frame it as a classification problem and use the built-in loss functions
- Frame it as a ranking problem and create custom loss functions (typically with negative item sampling)
- Frame it as a weighted regression / classification problem (similar to weighted ALS for implicit data)

```
from keras import backend as K

def bpr_triplet_loss(X):
    pos_item_embedding, neg_item_embedding, user_embedding = X

    loss = 1.0 - K.sigmoid(
        K.sum(user_embedding * pos_item_embedding, axis=-1, keepdims=True) -
        K.sum(user_embedding * neg_item_embedding, axis=-1, keepdims=True))

    return loss

pos_item_embedding = Flatten()
    item_embedding(pos_item_input))
neg_item_embedding = Flatten()
    item_embedding(neg_item_input))

loss = merge(
    [pos_item_embedding, neg_item_embedding, user_embedding],
    mode=bpr_triplet_loss,
    output_shape=(1, ))
```
Deep content-based music recommendation

Example of using a neural network model to act as feature extractor for item content / metadata

- CNN with audio spectogram as input data
- Filters capture lower-level audio characteristics, progressing to high-level features (akin to image problems)
- Max pooling and global pooling
- Fully connected layers with ReLU activations
- Output layer is the factor vector for the track from a trained collaborative filtering model
- Models trained separately

Source: Spotify / Sander Dieleman
Deep Learning for Recommendations

Wide and Deep Learning

Combines the strengths of “shallow” linear models with deep learning models

– Used for Google Play app store recommendations
– Linear model captures sparse feature interactions, while deep model captures higher-order interactions

– Sparsity of the deep model handled by using embeddings
– Both models trained at the same time

Source: Google Research
DeepFM

Wide and Deep architecture aiming to leverage strengths of Factorization Machines for the linear component

– Models trained together and both parts share the same weights

– More flexible handling of mixed real / categorical variables

– Reduces need for manually engineering interaction features
Deep Learning for Recommendations

Content2Vec

Specialize content embeddings for recommendation

- Combine modular sets of feature extractors into one item embedding
- e.g. CNN-based model for images (AlexNet)
- e.g. Word2Vec, sentence CNN, RNN for text
- e.g. Prod2Vec for embedding collaborative filtering (co-occurrences)
- Modules mostly pre-trained in some form
- Final training step then similar to “transfer learning”
- Use pair-wise item similarity metric (loss)

Source: Criteo Research
Session-based recommendation

Apply the advances in sequence modeling from deep learning

- RNN architectures trained on the sequence of user events in a session (e.g. products viewed, purchased)

- Adjustments for domain
  - Item encoding
  - Parallel mini-batch processing
  - Ranking losses

Source: Hidasi, Karatzoglou, Baltrunas, Tikk
Contextual Session-based models

Add contextual data to the RNN architecture

– Context included time, time since last event, event type

– Combine context data with input / output layer

– Also combine context with the RNN layers

– About 3-6% improvement (in Recall@10 metric) over simple RNN baseline

– Importantly, model is even better at predicting sales (vs view, add to cart events) and at predicting new / fresh items (vs items the user has already seen)
Deep Learning for Recommendations

Challenges

Challenges particular to recommendation models

- Data size and dimensionality
- Extreme sparsity (embeddings help)
- Wide variety of specialized settings
- Model serving is difficult – ranking, large number of items, computationally expensive
- Evaluation – model accuracy and its relation to real-world outcomes and behaviors
- Combining content, context and preference data
- Need for standard, open, large-scale, datasets that are content- and context-rich
Model Serving

Large item sets and ranking-based prediction make model serving at scale very challenging

- *Two-phase* approach is common
  - 1st phase generates candidates
  - 2nd phase re-ranks candidate list

- Candidate generation phase options include
  - Simpler form of model (e.g. MF, linear, deep) and/or simpler feature set
  - Approximate nearest neighbor search (e.g. LSH)

- Re-ranking may use more complex model, different objective function

Figure 2: Recommendation system architecture demonstrating the “funnel” where candidate videos are retrieved and ranked before presenting only a few to the user.
Future Directions

Most recent and future directions in research & industry

– Improved RNNs
  • Cross-session models
  • Further research on contextual models, as well as content and metadata
– Combine sequence and historical models (long- and short-term user behavior)
– Improving and understanding user and item embeddings

– Applications at scale
  • Dimensionality reduction techniques (e.g. Bloom embeddings for large input/output spaces)
  • Distributed training
  • Efficient model serving for complex architectures
Summary

Deep Learning for Recommendations

DL for recommendation is just getting started (again)

– Huge increase in interest, research papers. Already many new models and approaches
– DL approaches have generally yielded incremental % gains
  • But that can translate to significant $$$
  • More pronounced in e.g. session-based
– Cold start scenarios benefit from multi-modal nature of DL models
– Flexibility of DL frameworks helps a lot

– Benefits from advances in DL for images, video, NLP etc
– Open-source libraries appearing (e.g. Spotlight)
– Check out DLRS workshops & tutorials @ RecSys 2016 / 2017
Thank you!

Nick Pentreath
Principal Engineer
—
nickp@za.ibm.com
@MLnick
ibm.com
Links & References

Wikipedia: Perceptron

Amazon’s Item-to-item Collaborative Filtering Engine

Matrix Factorization for Recommender Systems (Netflix Prize)

Deep Learning for Recommender Systems Workshops @ RecSys

Deep Learning for Recommender Systems Tutorial @ RecSys 2017

Fashion MNIST Dataset

Deep Content-based Music Recommendation

Google’s Wide and Deep Learning Model

Spotlight: Recommendation models in PyTorch
Links & References

Stanford CS231n Convolutional Neural Networks for Visual Recognition

Restricted Boltzmann Machines for Collaborative Filtering

Specializing Joint Representations for the task of Product Recommendation

Session-based Recommendations with Recurrent Neural Networks

Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks

DeepFM: A Factorization-Machine based Neural Network for CTR Prediction

Getting deep recommenders fit: Bloom embeddings for sparse binary input/output networks

Deep Neural Networks for YouTube Recommendations

Ranking loss with Keras