Deep Learning for Recommender Systems

Nick Pentreath
Principal Engineer

@MLnick

IBM CODE

About

@MLnick on Twitter & Github

Principal Engineer, IBM

CODAIT - Center for Open-Source Data & AI Technologies

Machine Learning & AI

Apache Spark committer & PMC

Author of *Machine Learning with Spark*

Various conferences & meetups

Center for Open Source Data and AI Technologies

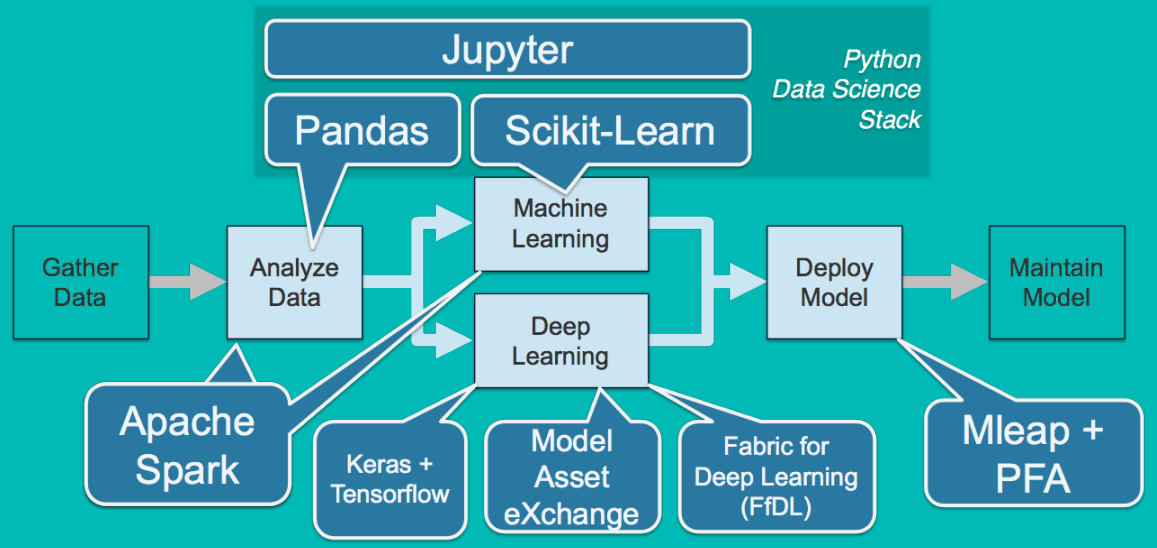
CODAIT

codait.org

CODAIT aims to make AI solutions dramatically easier to create, deploy, and manage in the enterprise

Relaunch of the Spark Technology Center (STC) to reflect expanded mission

Improving Enterprise AI Lifecycle in Open Source



Agenda

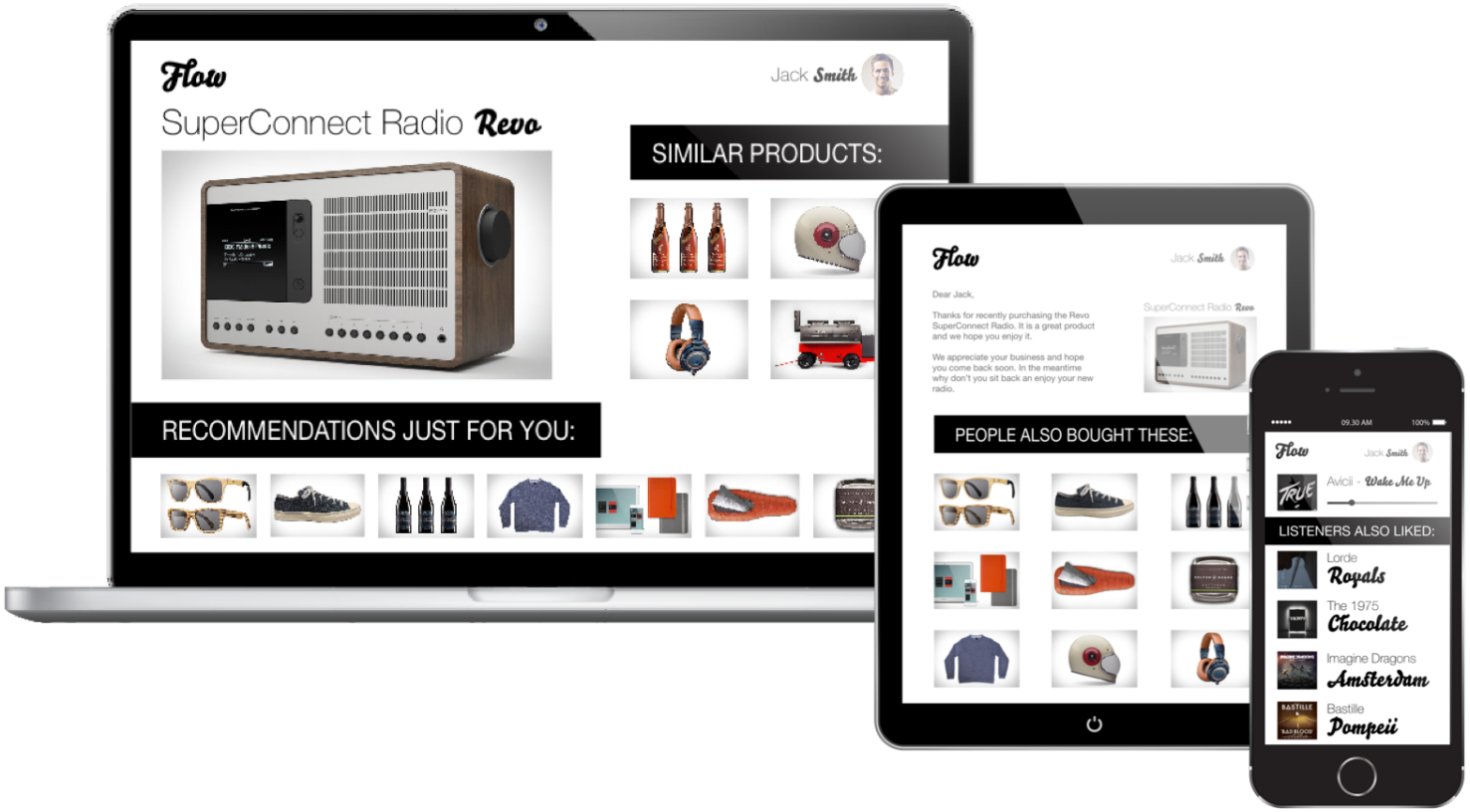
Recommender systems overview

Deep learning overview

Deep learning for recommendations

Challenges and future directions

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": "San Francisco",
  ...
}
```

```
{
  "item_id": "10",
  "name": "LOL Cats",
  "description": "catscatscats",
  "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

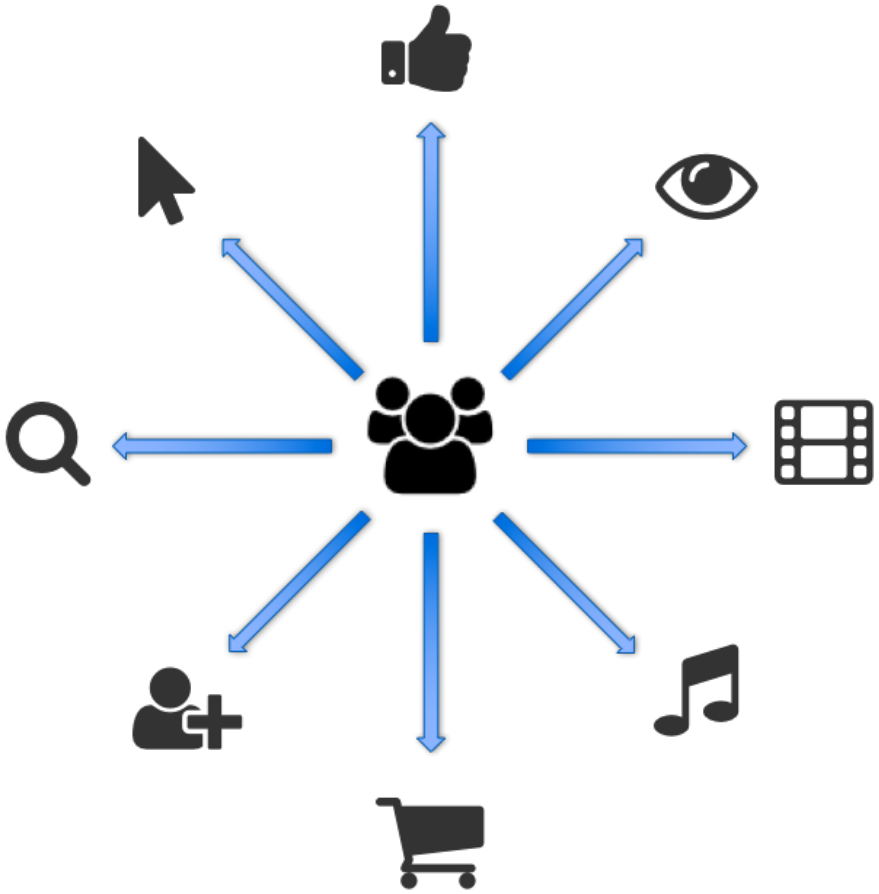
- Ratings, reviews

Intent

- Search query

Social

- Like, share, follow, unfollow, block



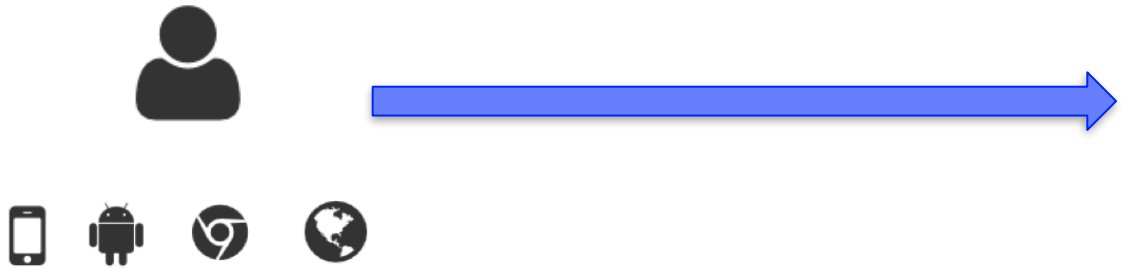
Context

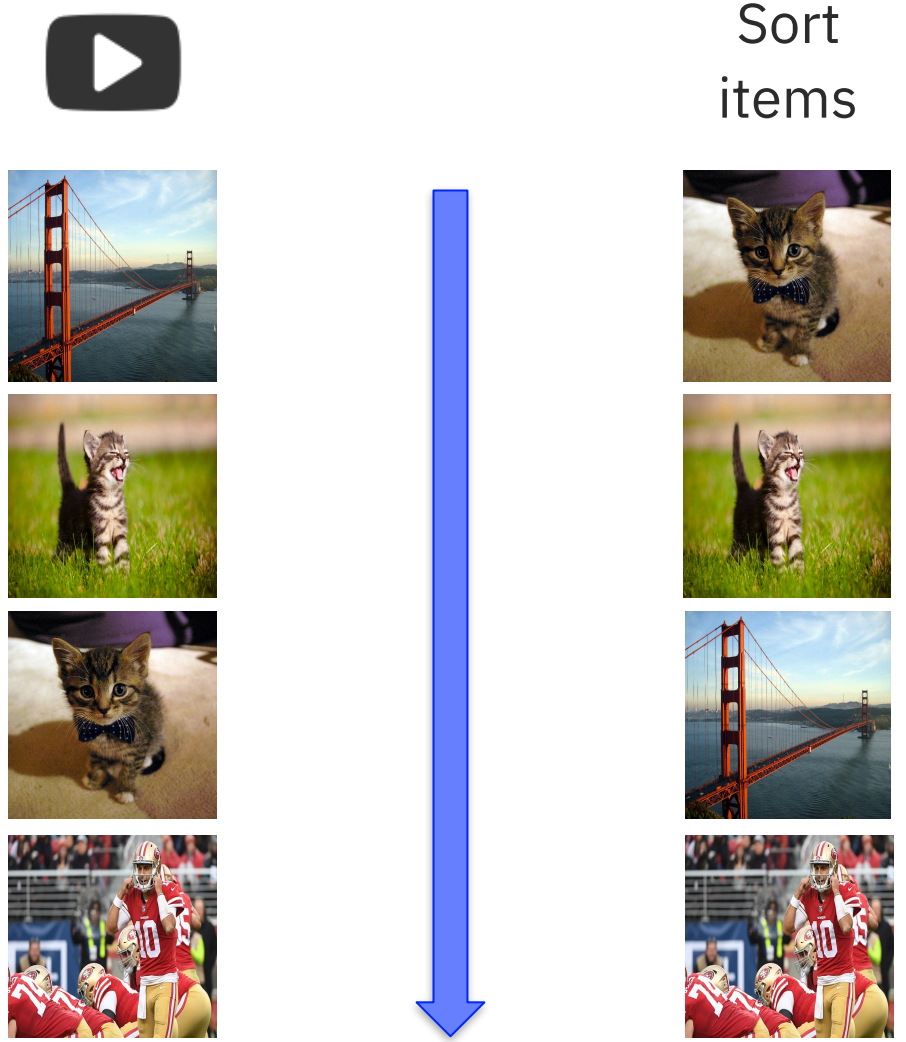
```
{
  "user_id": "1",
  "item_id": "10",
  "event_type": "page_view",
  "timestamp": 1476884080,
  "referrer": "http://codait.org",
  "ip": "123.12.12.12",
  "device_type": "Smartphone",
  "user_agent_os": "Android",
  "user_agent_type": "Mobile Browser",
  "user_agent_family": "Chrome Mobile",
  "geo":"37.784", "-122.401"
  ...
}
```

Prediction

Prediction is ranking

– Given a user and context, rank the available items in order of likelihood that the user will interact with them





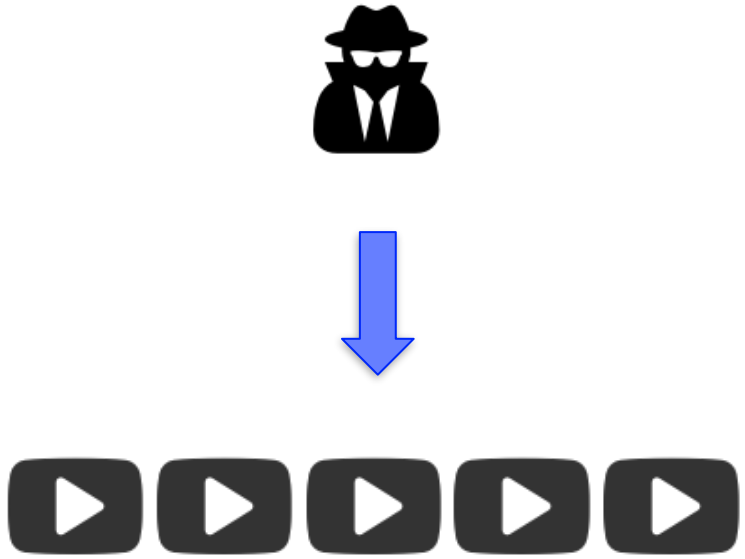
Cold Start

New items

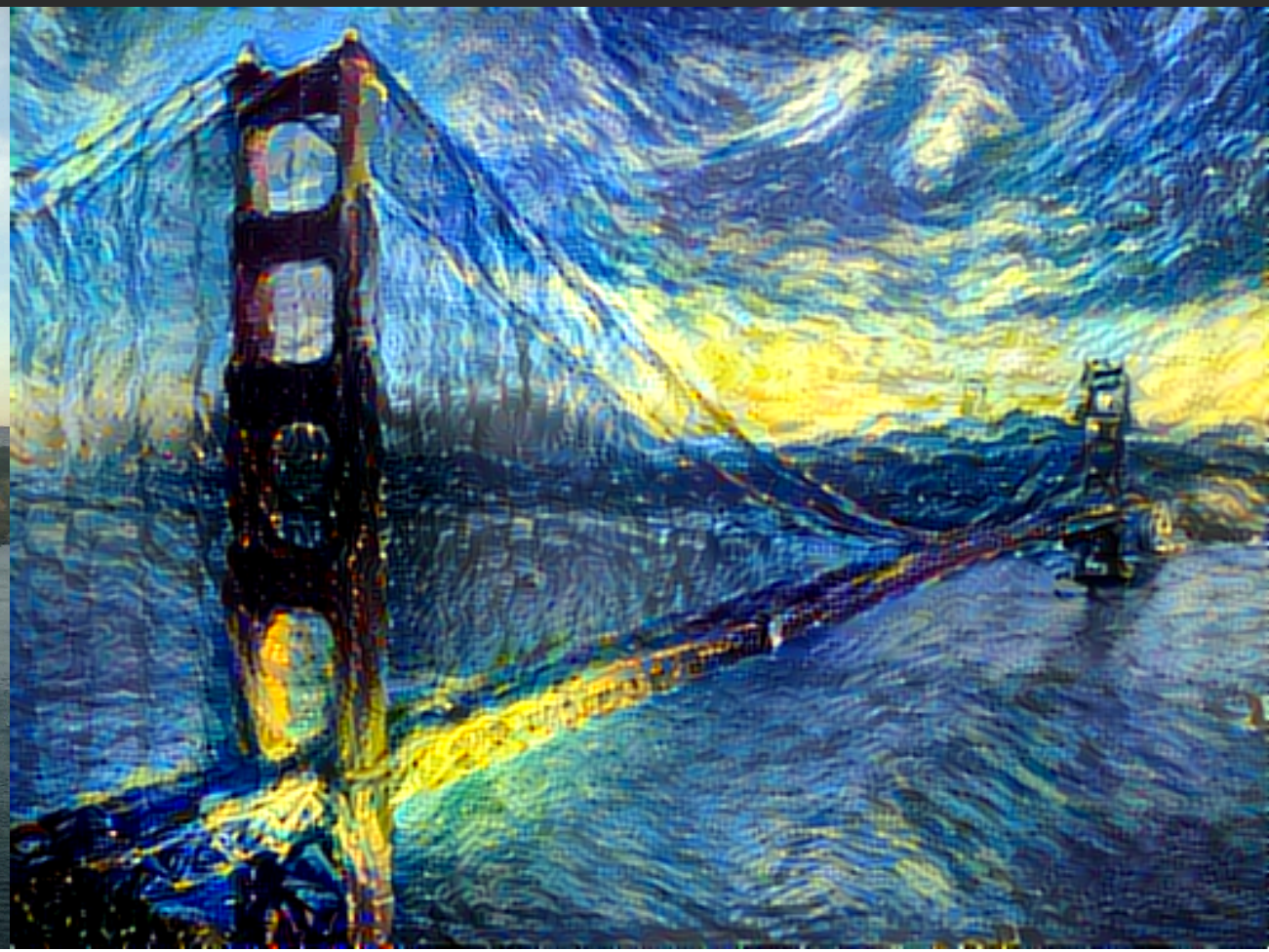
- No historical interaction data
- Typically use baselines or item content

New (or unknown) users

- Previously unseen or anonymous users have no user profile or historical interactions
- Have context data (but possibly very limited)
- Cannot directly use collaborative filtering models
  - Item-similarity for current item
  - Represent user as aggregation of items
  - Contextual models can incorporate short-term history



Deep Learning



Overview

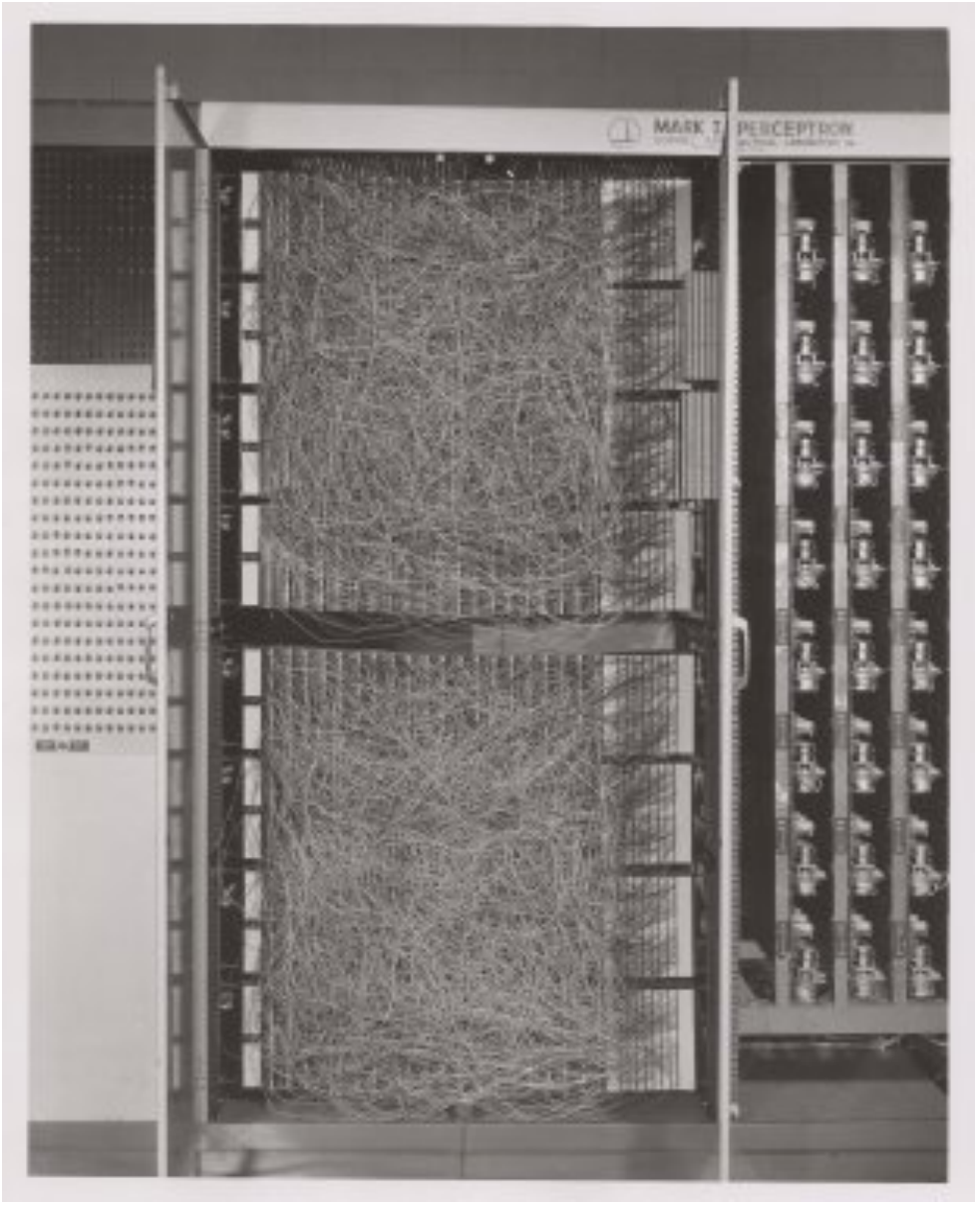
Original theory from 1940s; computer models originated around 1960s; fell out of favor in 1980s/90s

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

Source: Wikipedia



Source: Stanford CS231n

Modern Neural Networks

Deep (multi-layer) networks

Computer vision

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

Sequences and time-series

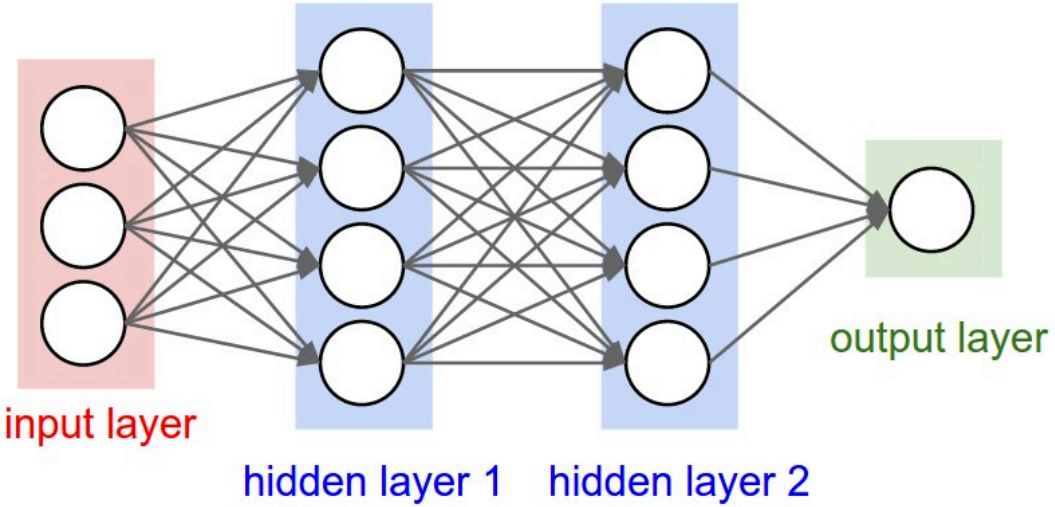
- Recurrent neural networks (RNNs)
- Machine translation, text generation
- LSTMs, GRUs

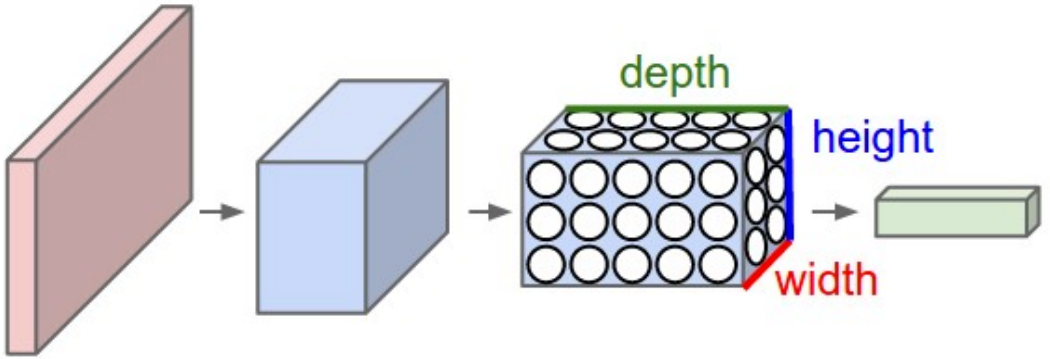
Embeddings

- Text, categorical features

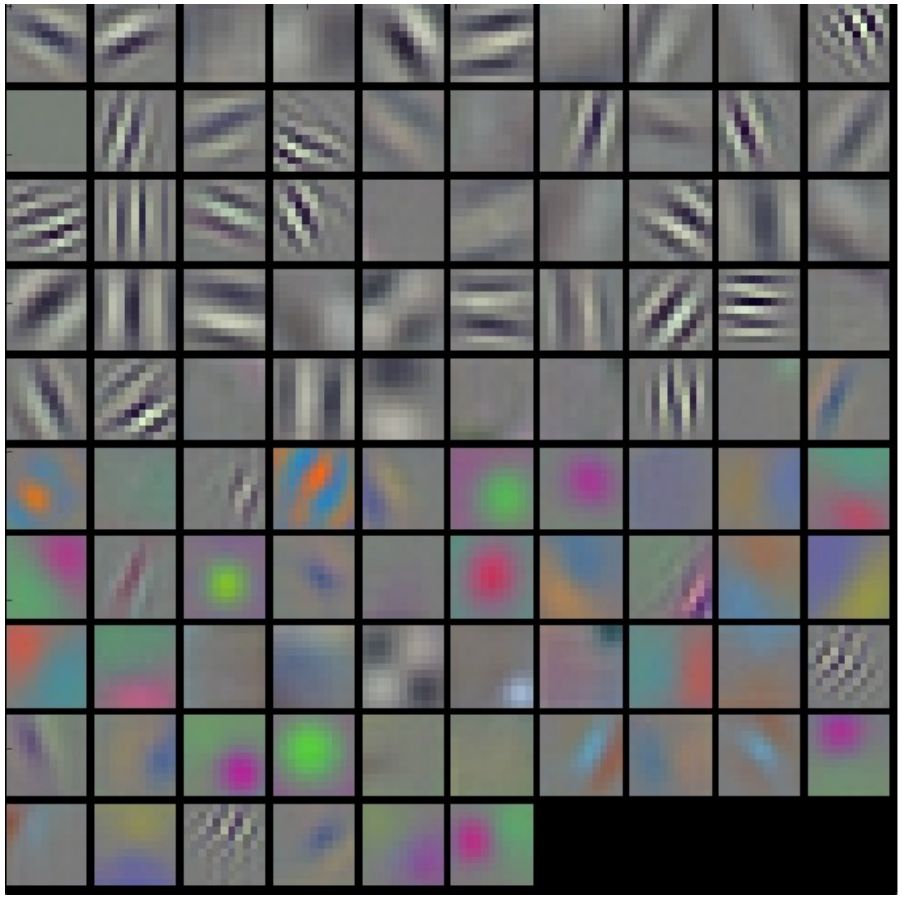
Deep learning frameworks

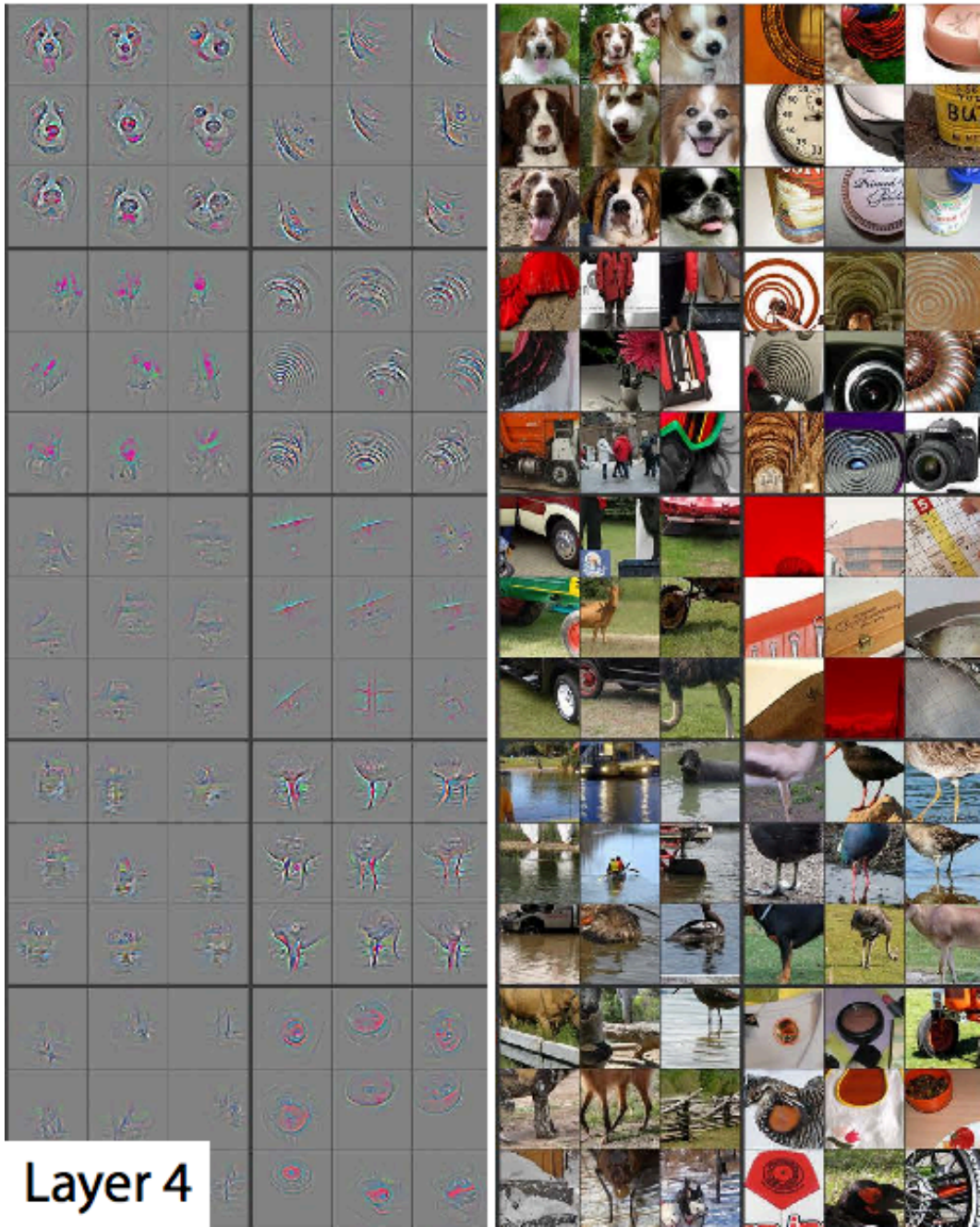
- Flexibility, computation graphs, auto-differentiation, GPUs





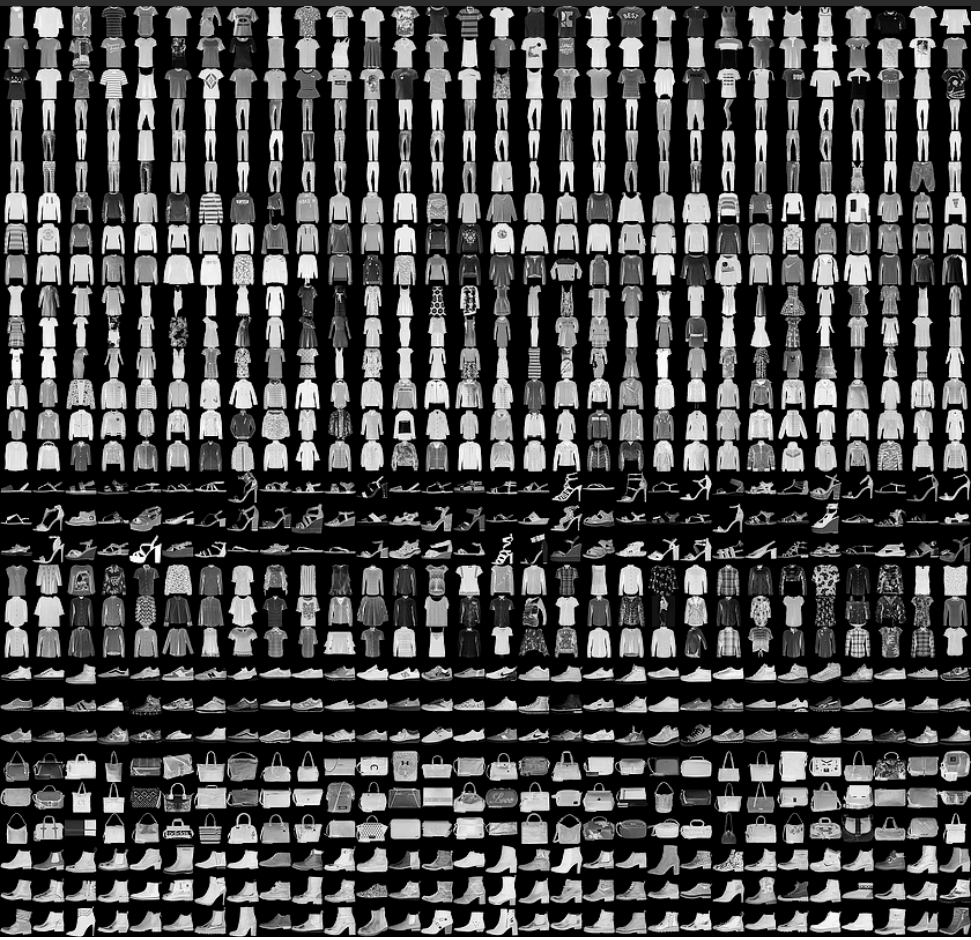
Deep Learning as Feature Extraction

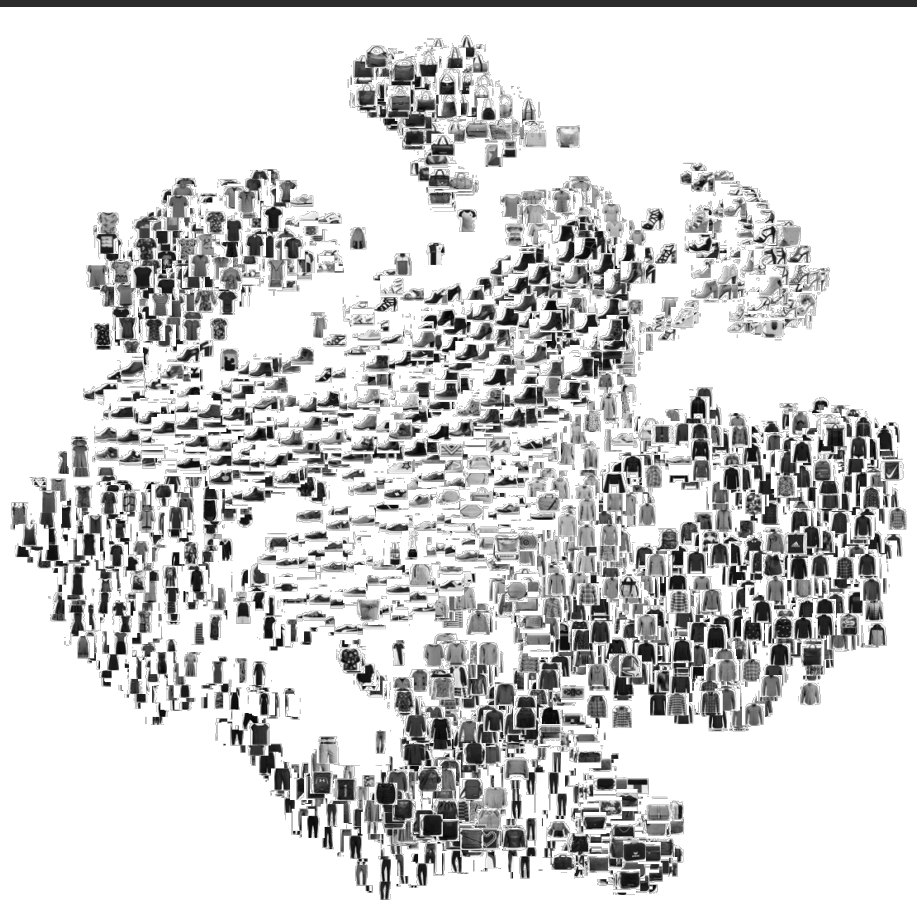


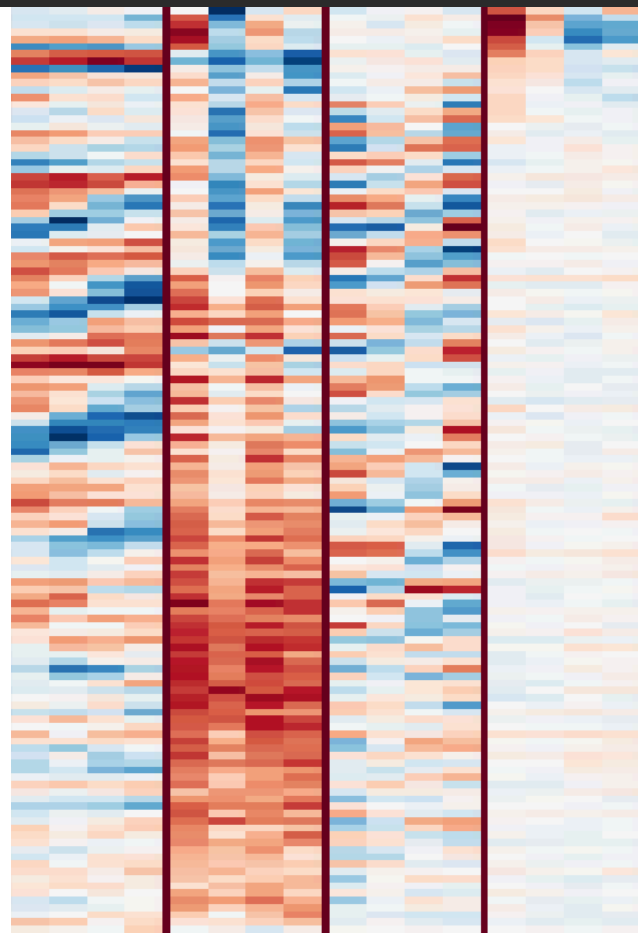


Source: Stanford CS231n; D. Zeiler,R. Fergus

Deep Learning for Recommendations

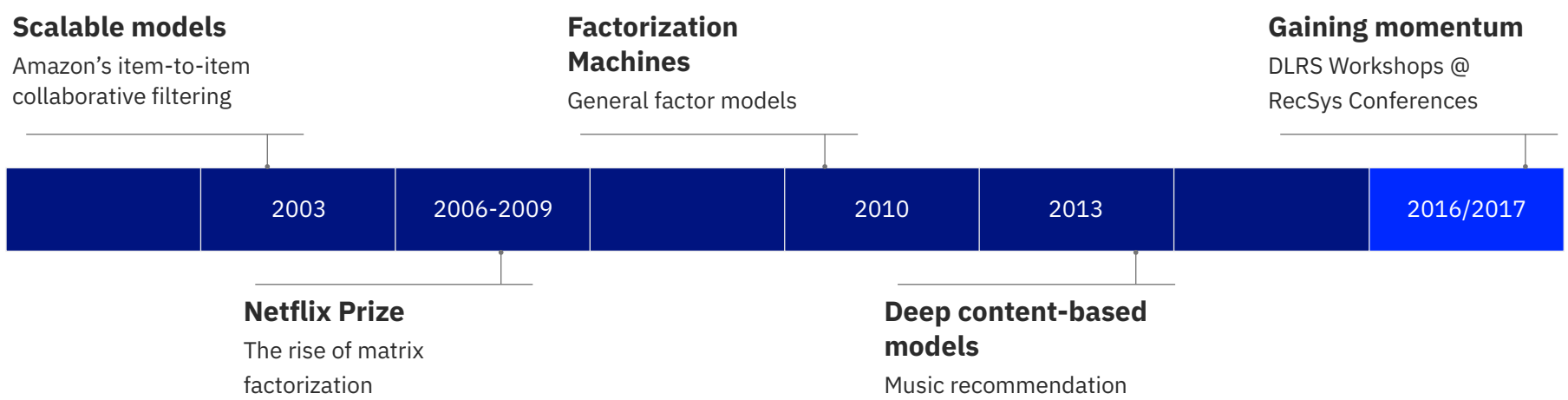






Evolution of Recommendation Models

- 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



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

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: dot product
- Can add user and item biases using single-dimensional embeddings
- What about implicit data?

```
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")
```

Source: Maciej Kula

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)

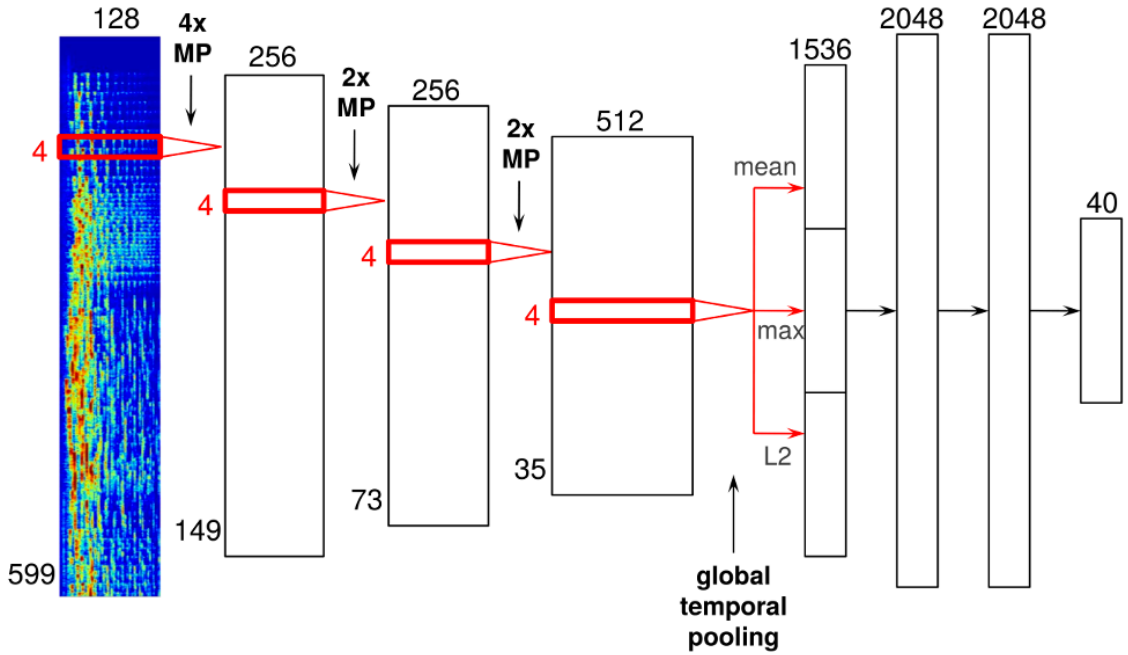
```
from keras import backend as K

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, ))
```

Source: Spotify / Sander Dieleman

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

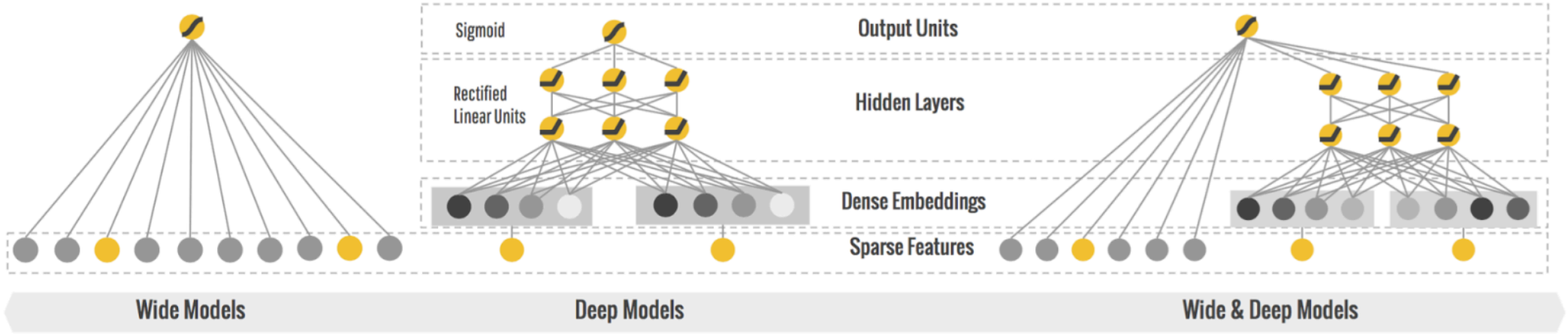


Source: Google Research

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: Guo, Tang, et al

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

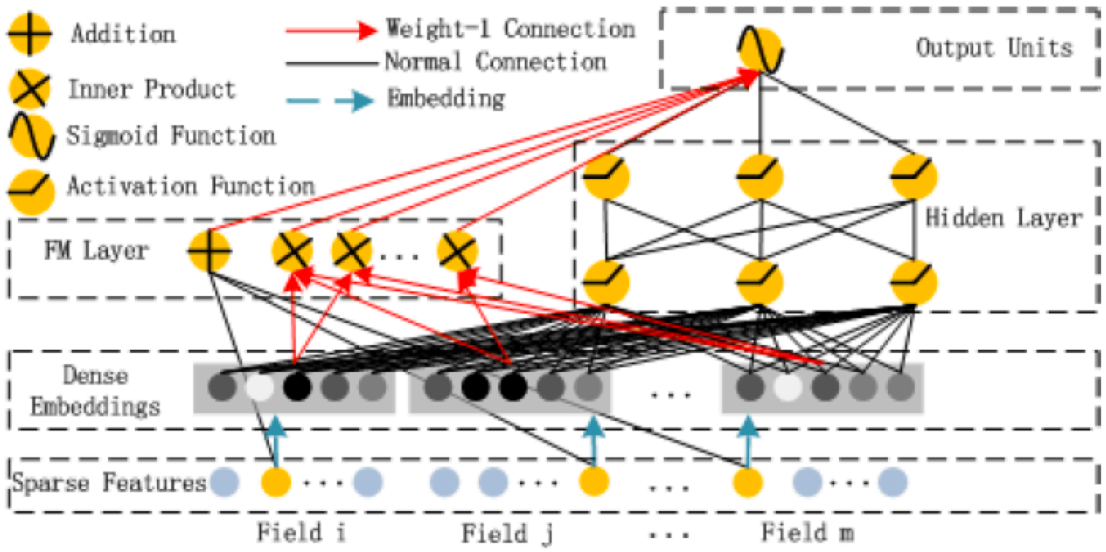


Table 2: Performance on CTR prediction.

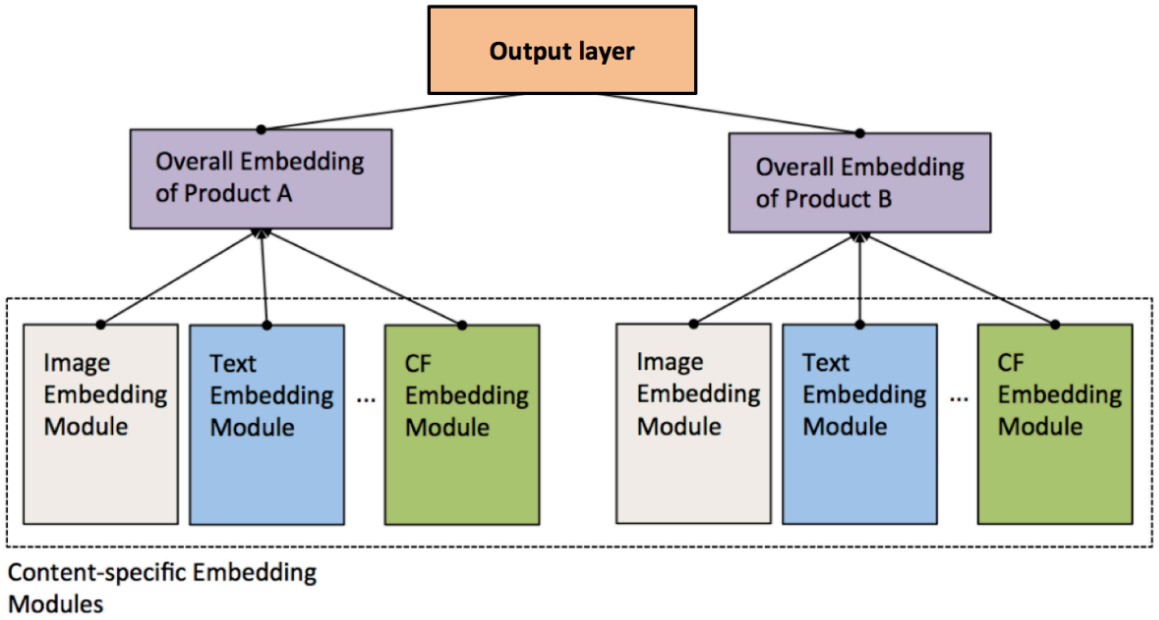
	Company*		Criteo	
	AUC	LogLoss	AUC	LogLoss
LR	0.8640	0.02648	0.7686	0.47762
FM	0.8678	0.02633	0.7892	0.46077
FNN	0.8683	0.02629	0.7963	0.45738
IPNN	0.8664	0.02637	0.7972	0.45323
OPNN	0.8658	0.02641	0.7982	0.45256
PNN*	0.8672	0.02636	0.7987	0.45214
LR & DNN	0.8673	0.02634	0.7981	0.46772
FM & DNN	0.8661	0.02640	0.7850	0.45382
DeepFM	**0.8715**	**0.02618**	**0.8007**	**0.45083**

Source: Criteo Research

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)

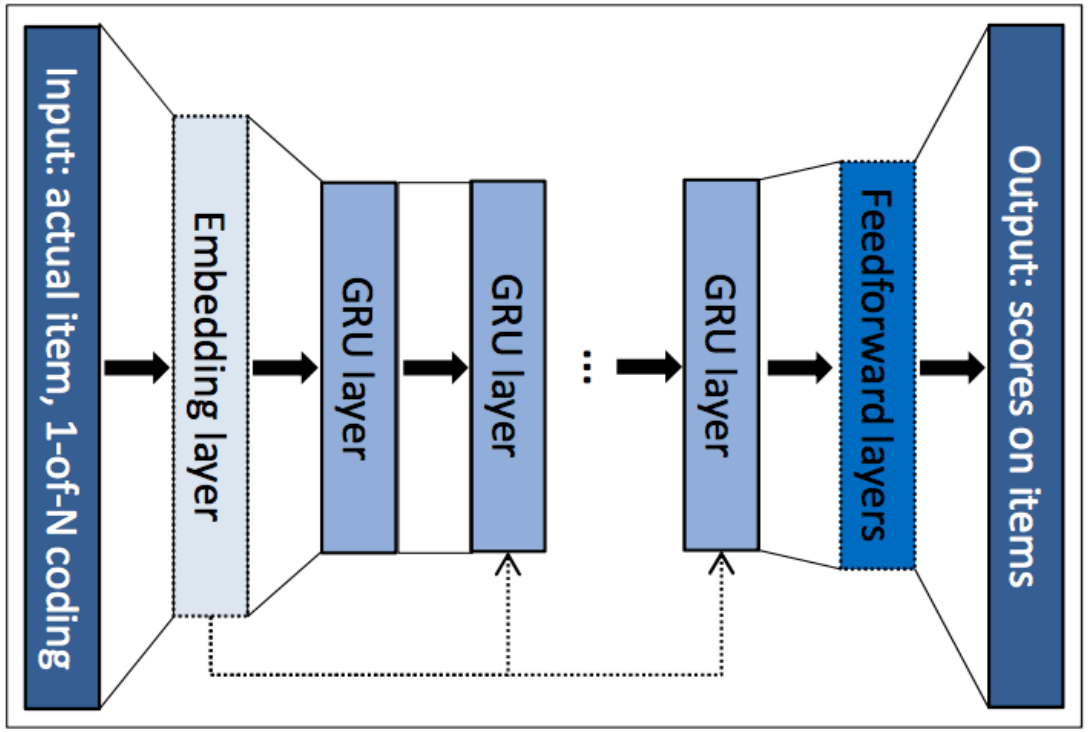


Source: Hidasi, Karatzoglou, Baltrunas, Tikk

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) to predict next item in session
- Adjustments for domain
  - Item encoding (1-of-N, weighted average)
  - Parallel mini-batch processing
  - Ranking losses – BPR , TOP1
  - Negative item sampling per mini-batch
- Report 20-30% accuracy gain over baselines

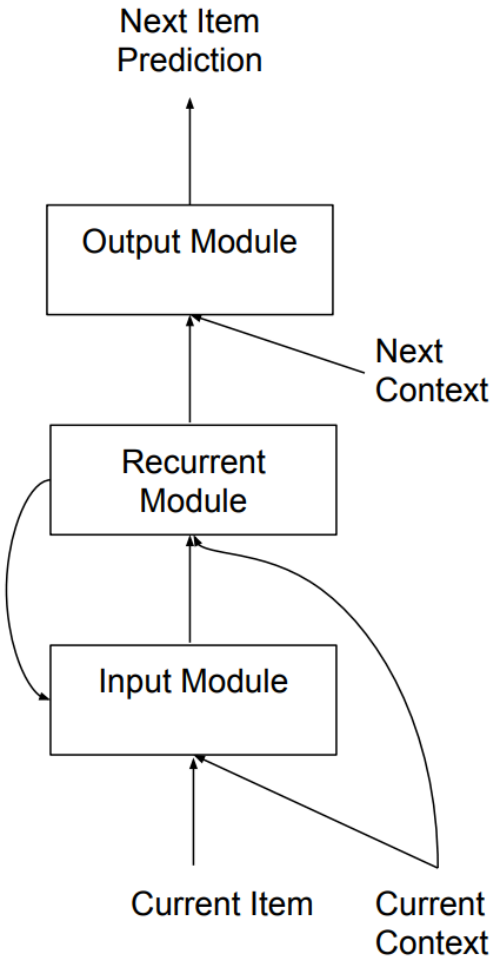


Source: Smirnova, Vasile

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)



Challenges

Challenges particular to recommendation models

- Data size and dimensionality (input & output)
- Extreme sparsity
  - Embeddings & compressed representations
- Wide variety of specialized settings
- Combining session, content, context and preference data
- Model serving is difficult – ranking, large number of items, computationally expensive
- Metrics – model accuracy and its relation to real-world outcomes and behaviors
- Need for standard, open, large-scale, datasets that have time / session data and are content- and context-rich
- Evaluation – watch you baselines!
  - When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation

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. feature hashing, Bloom embeddings for large input/output spaces)
  - Compression / quantization techniques
  - Distributed training
  - Efficient model serving for complex architectures

Summary

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
- Open-source libraries appearing (e.g. Spotlight)
- Check out DLRS workshops & tutorials @ RecSys 2016 / 2017, and upcoming in Oct, 2018

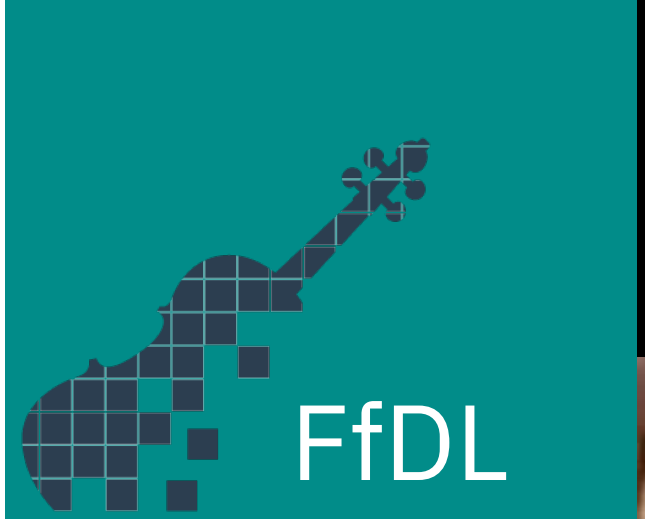
Thank you

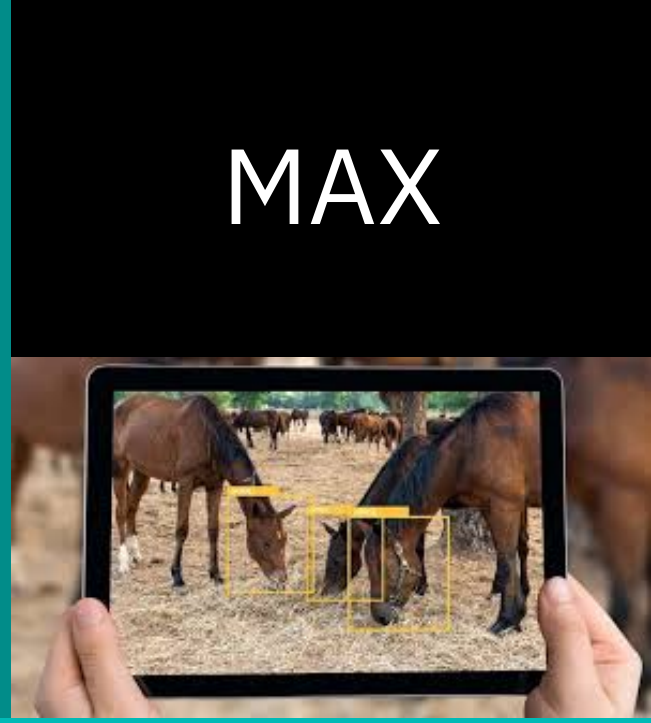
codait.org

twitter.com/MLnick

github.com/MLnick

developer.ibm.com/code





Sign up for IBM Cloud and try Watson Studio!

https://datascience.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

Visualizing and Understanding Convolutional Networks

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

3D Convolutional Networks for Session-based Recommendation with Content Features