Part 2. Deep Learning Methods for Recommendation

Flavian Vasile & Olivier Koch

Criteo Research

July 2, 2018

criteo

Contents

Structure of the talk

- Introduction to Deep Recommendations
- Matrix Factorization Deep Learning Extensions
  - Word2Vec
- Complex input Recommendation with Convolutions
  - 2D CNNs for Image-based Reco
  - 1D CNNs for Text, Music and Sequence-based Reco
  - 3D CNNs for merging content
- User modeling with RNNs for Recommendation

Introduction to Deep Learning for Recommendation

What is Deep Learning?

Deep learning is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms.

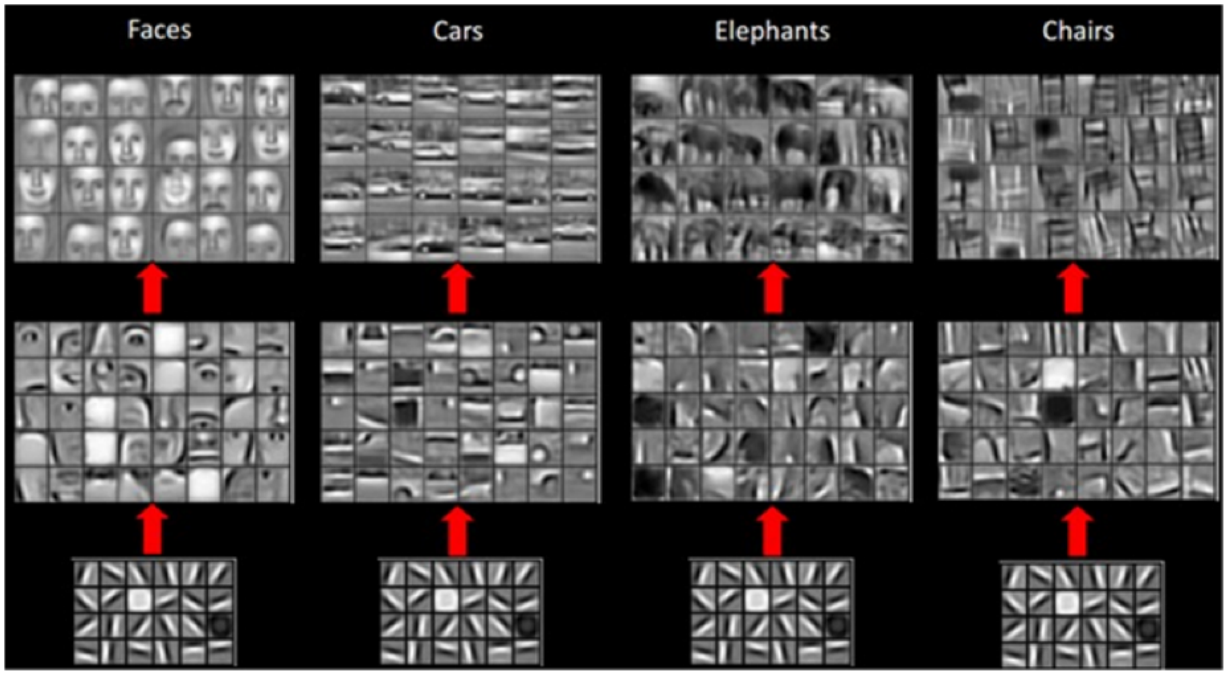
Whats so special about Deep learning?

DLs performance and speed of adoption relies on 3 Key Concepts:

- End-to-end ML by: Representation Learning
- Advances in practical Non-linear Optimization: SGD, Backpropagation and AutoDiff
- Flexibility: Differentiable Programming

What is Representation Learning?

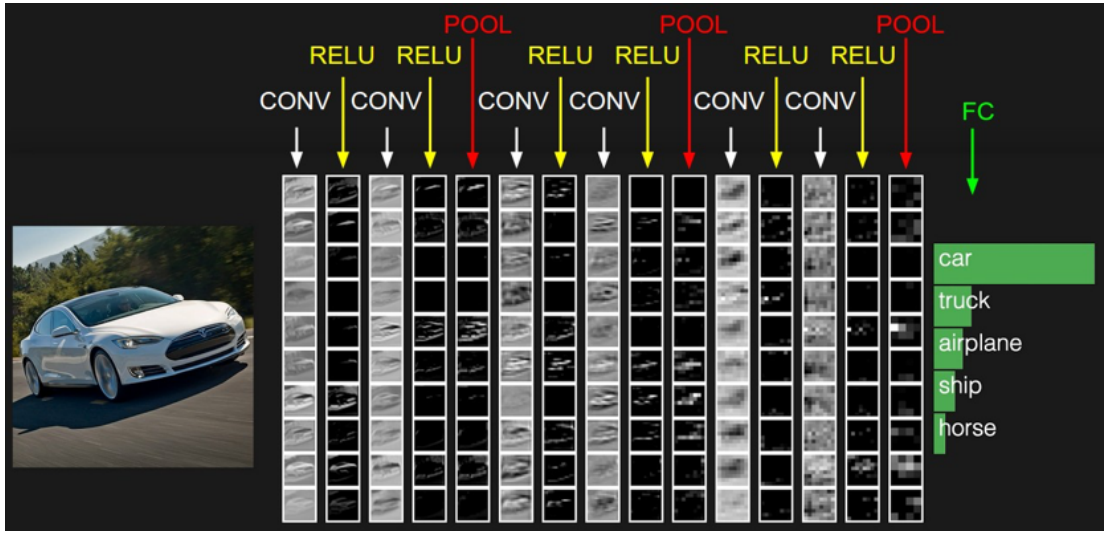
Representation Learning = automatic feature engineering



Extracted image features Image Source: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

What is Representation Learning?

ML Systems that do not require any human intervention on the collected input in order to learn and predict: end-to-end learning systems



CNN Architecture Image Source: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

What is Representation Learning?

Advantages:

- This makes manual feature engineering obsolete and saves the large amounts of time previously needed in order to find the right features (80% data preprocessing, 20% model tuning)

Furthermore, it allows a machine to learn in the same time:

- the specific task (using the features) and
- the best features for the task,
- The end-to-end automation can dramatically increase the final performance of the model

Optimization: SGD, BackProp and AutoDiff

SGD: Stochastic Gradient Descent

Most of the reported results of Deep Learning are due to the *Unreasonable Effectiveness of SGD*

Stochastic gradient descent

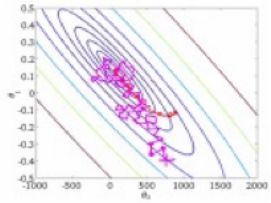
Minimize $F(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})$

Initialize $\mathbf{x_0}$

For $j = 1, 2, \ldots$

draw index $i = i_j$ at random

$\mathbf{x^{(j+1)}} = \mathbf{x^{(j)}} - \gamma \nabla f_i(\mathbf{x^{(j)}})$

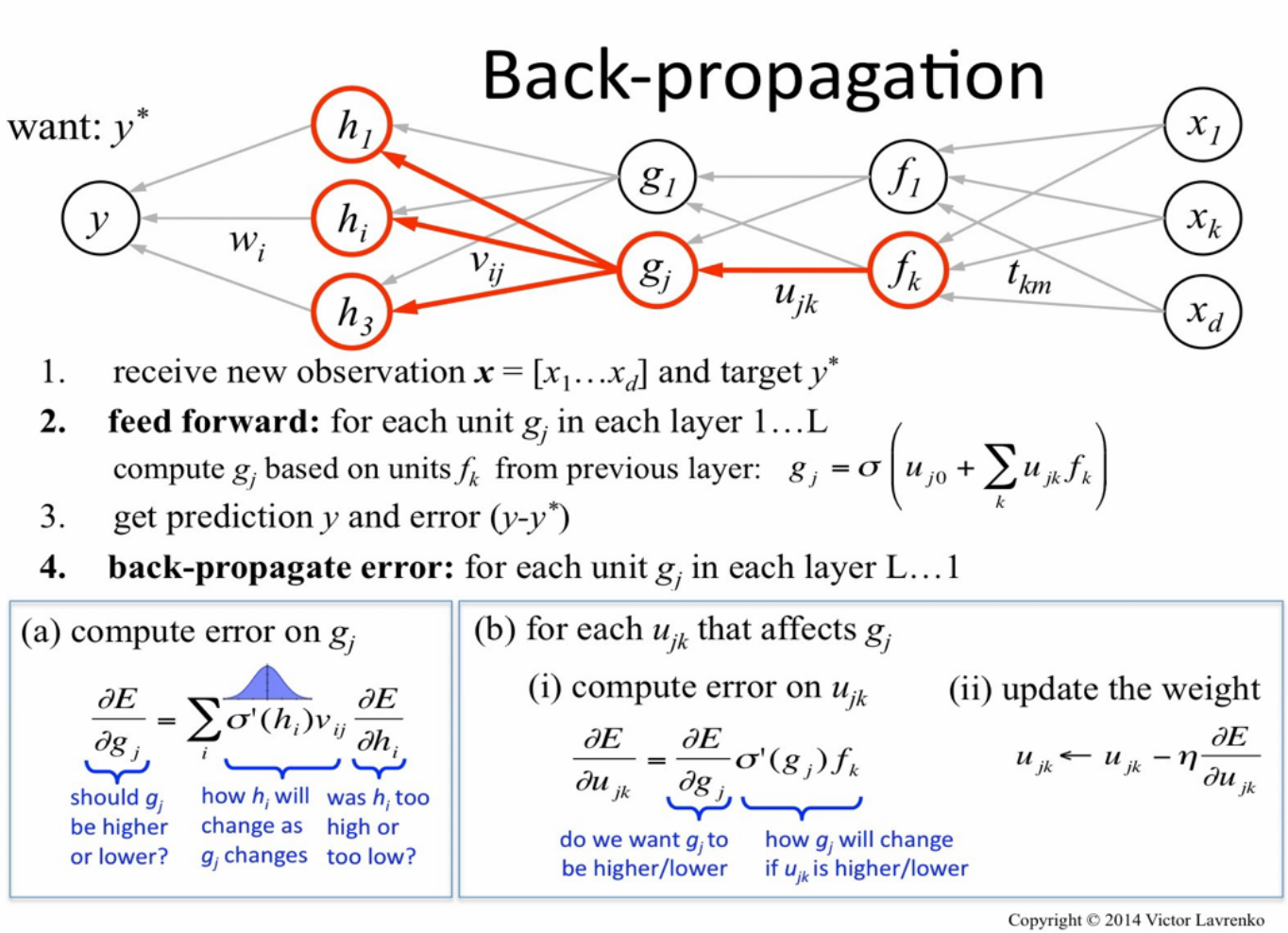


Goal: nonasymptotic bounds on $\mathbb{E}\|\mathbf{x}^{(j)} - \mathbf{x}^*\|^2$

The SGD algorithm

Back-propagation

It's really just the chain rulea simple calculus trickapplied in a very elegant way.



The Back-propagation algorithm

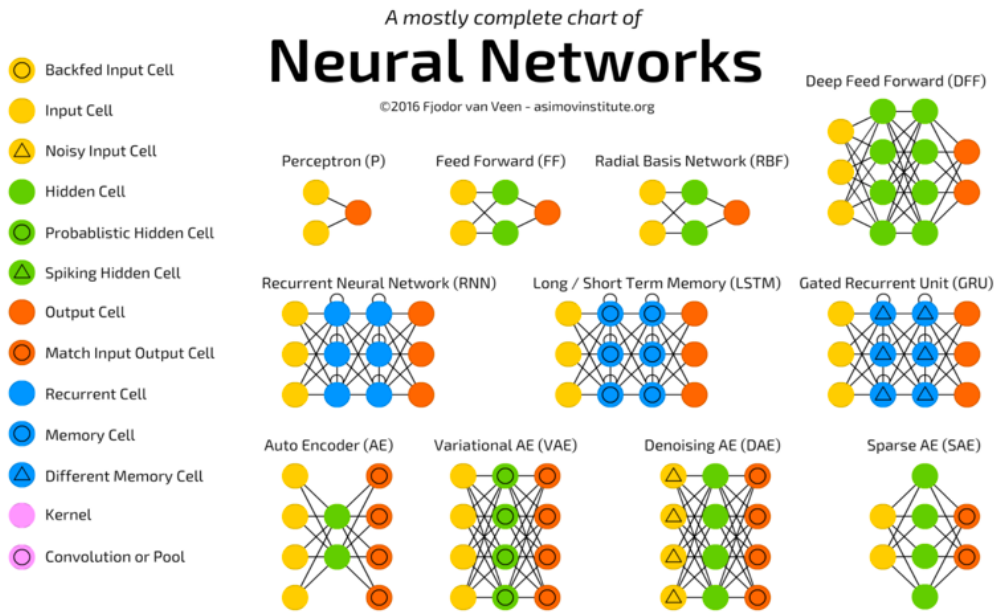
AutoDiff: Automatic Differentiation

AutoDiff: Set of techniques to numerically evaluate the derivative of a function specified by a computer program.

3 Steps:

- Decompose original Deep Neural Net into intrinsic functions (addition, substraction, exp, sin, cos, tanh)
- Differentiate the intrinsic functions
- Multiply together according to the chain rule

Flexibility: Differentiable Programming



DNN architecture types Image Source: `http://www.asimovinstitute.org/neural-network-zoo/`

Recap: 3 Key Concepts of DL

- Representation Learning capabilities
- The relative generality of the optimization solution: SGD, Backprop & AutoDiff
- Differentiable Programming framework

Because of the flexibility and the composability of neural modules, coupled with the possibility of training end-to-end ML systems, Deep Learning sees an accelerated adoption curve in many industries.

Q: What are we doing about it?

What can Deep Learning do for Recommendation?

In Recommendation, we can leverage Deep Learning in order to:

- Extend the idea of embedding users and products from MF (Word2Vec)
- Model the complex content information on the items (CNNs)
- Model the complex user interaction with the items (RNNs, TCNs)

Beyond Matrix Factorization: Deep Learning Extensions

The word2vec model

- Two-layer neural network trained to reconstruct a word context
- Motivation: explore simpler models that might not be able to represent the data as precisely as neural networks, but can possibly be trained on much more data efficiently.

Distributed Representations of Words and Phrases and their Compositionality, T. Mikolov et al, NIPS 2013

The word2vec model

Given a sequence of training words $w_1, w_2, \cdots, w_T$, maximize:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j} \mid w_t)$$

where c is the size of the training context

The word2vec model

Use hierarchical softmax for the representation of $p(w_O \mid w_I)$ for computational efficiency, instead of regular softmax:

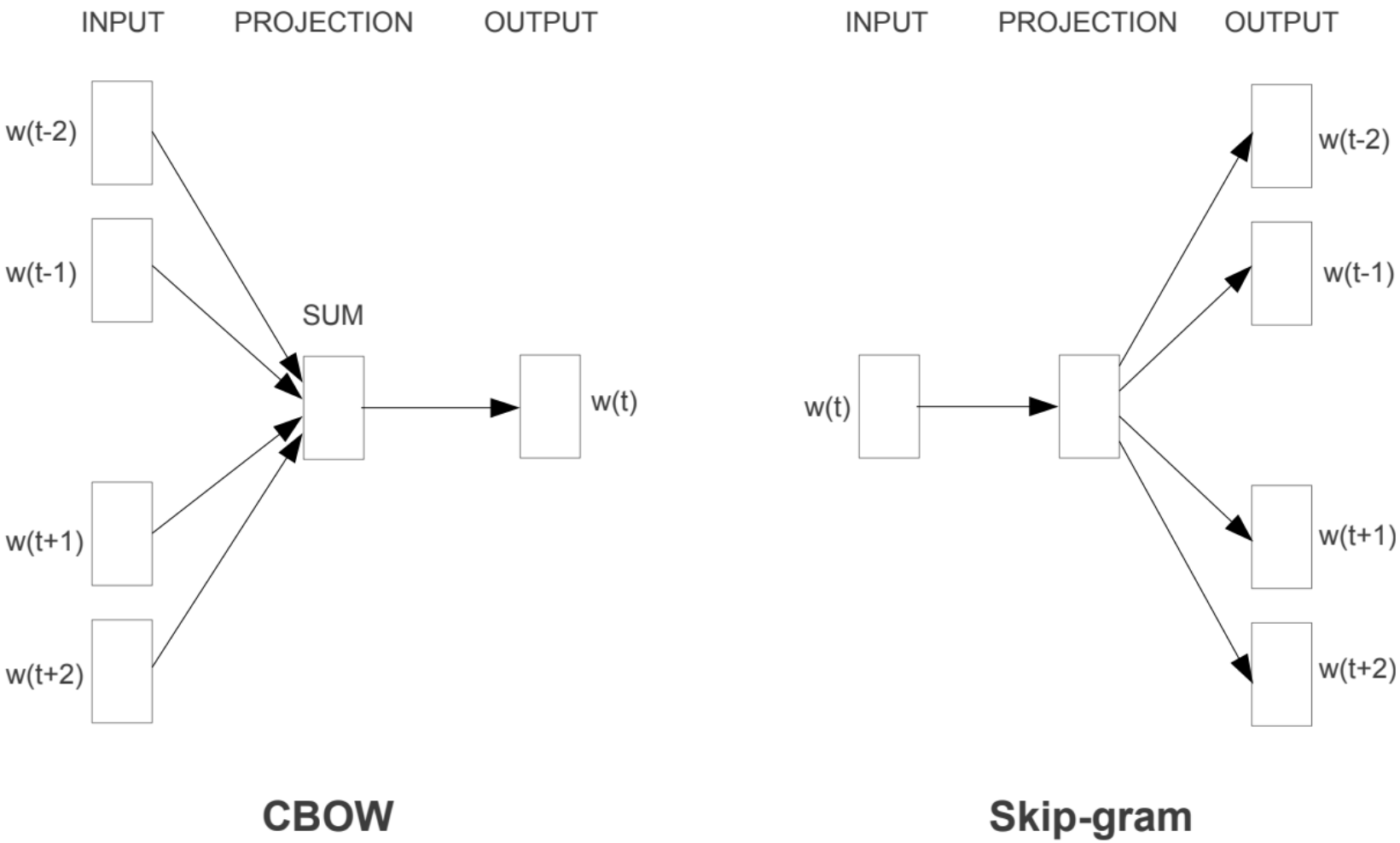
$$p(w_O \mid w_I) = \frac{exp(v'^T_{w_O} v_{w_I})}{\sum_{w=1}^{W} exp(v'^T_w v_{w_I})}$$

Skip-gram and Continuous Bag-of-Word (CBOW)

The word2vec architecture comes in two flavors:

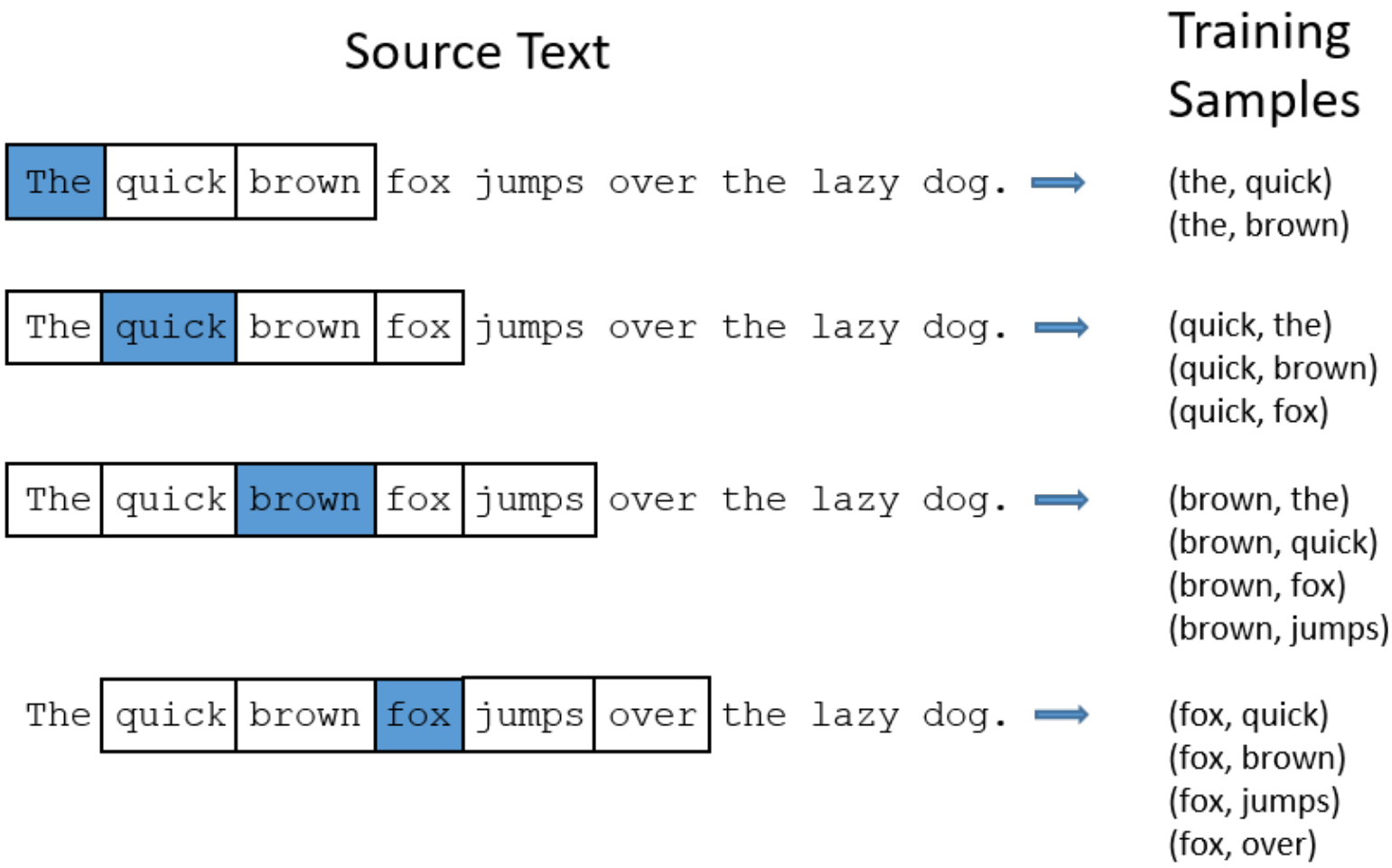
- Skip-gram: predict the current word given the context
- CBOW: predict the context given the current word

Word2vec skip-gram architecture



Efficient Estimation of Word Representations in Vector Space, ICLR 2013

Word2vec skip-gram training example

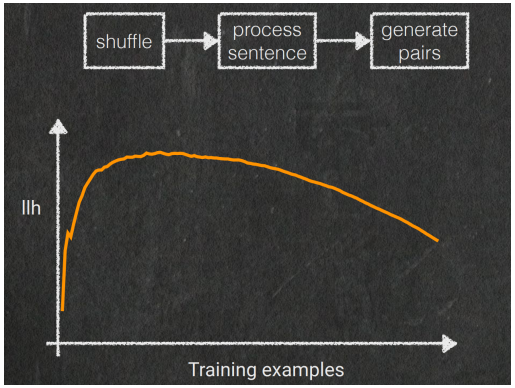


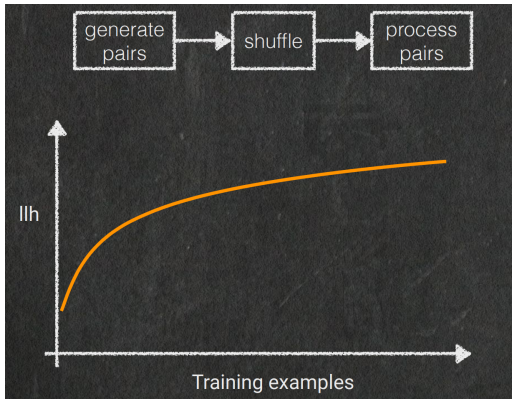
From word2vec to prod2vec

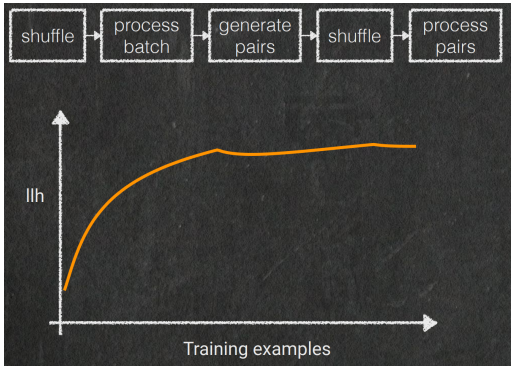
- prod2vec: consider products as words and sequences of interactions as sentences (skip-gram word2vec)
- user2vec: consider users as “global context“ (paragraph2vec)
- Brings significant uplift on CTR and YR (yield rate) compared to popular products

M. Grbovic et al, E-commerce in your inbox: Product recommendations at scale. SigKDD 2015

Word2vec at scale: preprocessing matters...







Feeding Word2vec with tens of billions of items, what could possibly go wrong?, S. Dolle, Berlin Buzzwords, 2015

Convolutional Neural Networks for Recommendation

Recommendation with Complex Content

In general, the items to be recommended have some meta-data associated to them.

These can be information such as: category, price, product image, textual description or in the case of media items, they are themselves pieces of content (songs, videos).

Some of this content has a complex structure, such as in the case of:

- Images
- Sounds
- Text
- Sequences (time)
- Combinations of the above

Recommendation with Complex Content

We are interested in ways to deal with complex content in a way that makes our recommendation more performant.

Of course, as in most content-based recommendation work, the main attraction of using content on top of collaborative filtering signal is to improve the cold start performance (the recommendation of new items).

Recommendation with Complex Content

Current state-of-the-art ML for content understanding is powered by CNNs

CNNs: Convolutional Neural Networks

In order to create state-of-the art content-based recommender systems we need to:

- Learn what are Convolutional Neural Networks and how do they work
- Study the existing deep architectures that are using CNNs for content-based recommendation

Quick intro to CNNs with an image processing example

What are Convolutional Neural Networks?

CNNs are using stacked convolutional modules paired with various output layers (e.g. softmax).

A convolutional module generally comprises of three main steps:

- Convolution
- Non-linearity
- Pooling

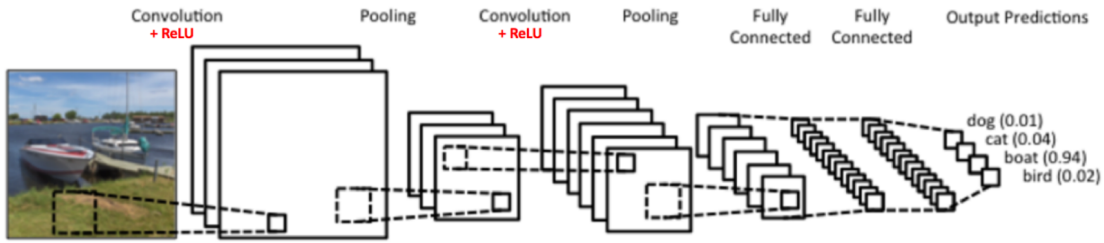


Image ConvNet Basic Architecture Image Source:
`https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/`

1. The Convolution Step

Intuition: Computing activations for a fully connected 1-layer neural network = Inner-product between the input vector and the weight matrix describing the layer.

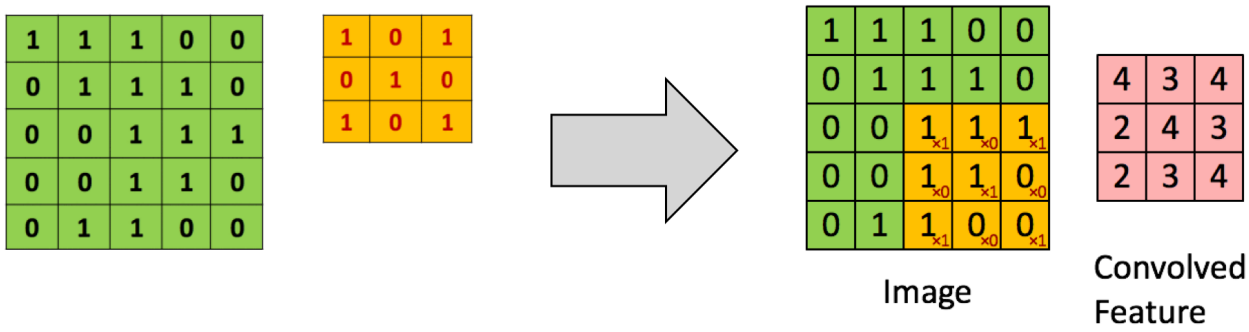
1. The Convolution Step

- What if we know that not all inputs are connected to the next layer?
- What if we know (like in the case of images) that there is a local structure in place (the relationship between neighboring pixels in 2D for images).
- Could we use this information when designing our network?
- Answer: Yes! And that it is exactly what a convolution is!
- An image convolution is a network that only takes the neighboring pixels into account when computing its activation.
- Because it takes a 2D neighborhood region, we will denote it as a 2D convolution.

1. The Convolution Step

Image Convolution: Local dot product of an input matrix with a smaller matrix that scans/moves along the input matrix

Example: See below the convolution of a 5x5 image (matrix of its binary pixels) with a smaller 3x3 matrix:



Convolution Example - Image Source: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

1. The Convolution Step

Vocabulary:

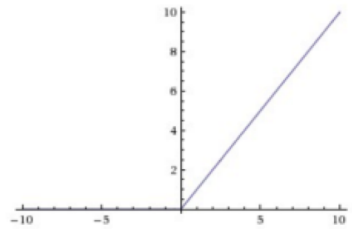
- The 33 matrix: filter, kernel, feature detector (since the matrices act as feature detectors from the original input image)
- The output of the operation: convolved feature, activation map, feature map

2. Non-linearity

Apply a non-linear function, usually **Rectified Linear Unit (ReLU)**:
$y = max(0, x)$

It is an pixel-wise transformation that replaces all negative pixel values in the feature map by zero.

Output = Max(zero, Input)



ReLU function - Image source: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

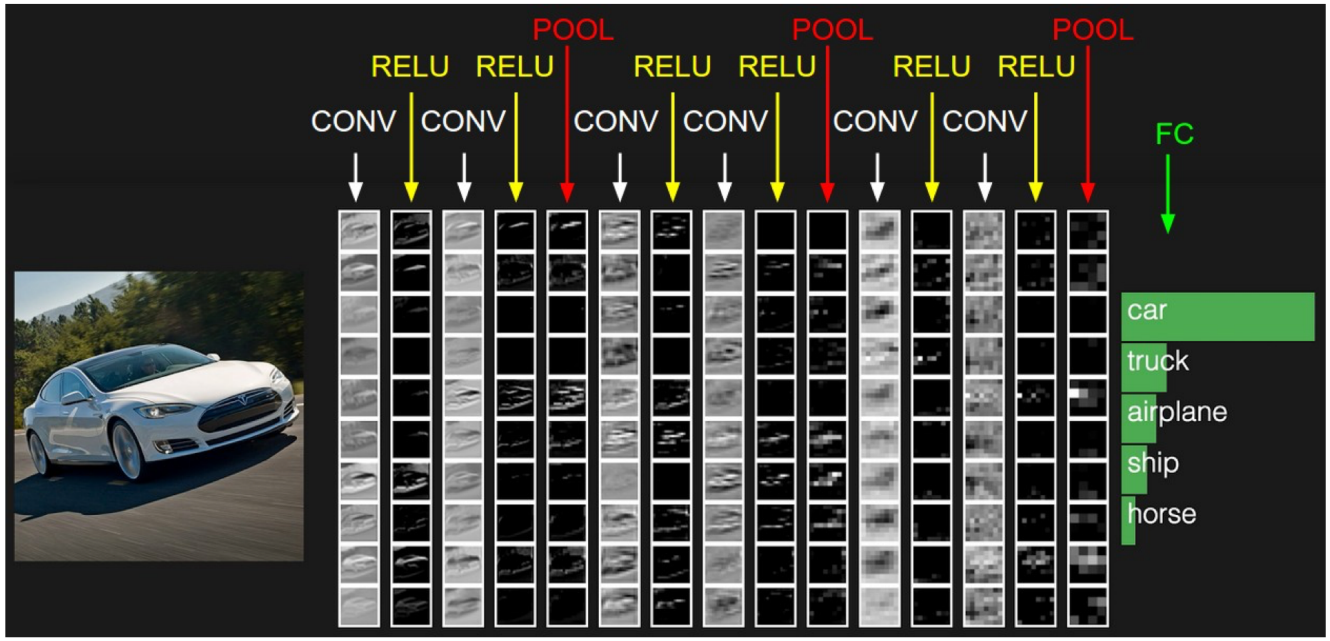
3. Pooling

- In the Pooling step, we define a region (e.g., a 33 window) and apply a function that summarizes the region (computes a statistic).
- In Max Pooling, we take the largest element from the rectified feature map within that window.
- Other pooling functions: Min, Average, Sum etc.

The function of pooling is to shrink the size of the input representation which reduces the number of parameters and associated computations and makes the network more invariant/robust to small transformations.

Stacking

ConvNet: Stacking the convolutional modules + output layer



Full ConvNet Architecture - Image source: `http://cs231n.github.io/convolutional-networks/`

2D Convolutions for Image Recommendation: Visual Bayesian Personalized Ranking (VBPR)

Image: Visual Bayesian Personalized Ranking (VBPR)

Main idea: Use visual features extracted from product images using pre-trained deep networks

Learn an additional layer on top of them that summarizes the visual impact of the item description in order to better explain the observed personalized purchase outcomes.

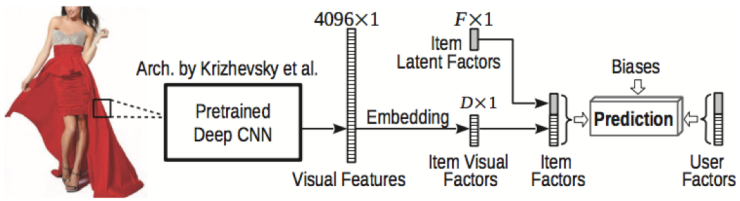


Figure 1: Diagram of our preference predictor. Rating dimensions consist of visual factors and latent (non-visual) factors. Inner products between users and item factors model the compatibility between users and items.

VBPR Architecture

He, Ruining, and Julian McAuley. "VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback." AAAI. 2016.

Image: Visual Bayesian Personalized Ranking (VBPR)

Task: Learn a ranking for each user over items from logged implicit feedback (e.g. purchase histories).

Model: The preference model is parametrized in terms of:

- Global bias term
- Bias terms for user u and item i
- Latent vectors for user u and item i
- Visual vectors for u and i
- The visual vector for i is a compressed representation of the ConvNet output (ConvNet used is AlexNet)
- Visual bias vector

$$\hat{x}_{u,i} = \alpha + \beta_u + \beta_i + \gamma_u^T \gamma_i + \theta_u^T (E f_i) + \beta'^T f_i$$

Image: Visual Bayesian Personalized Ranking (VBPR)

Loss function: *Bayesian Personalized Ranking (BPR)* - pairwise ranking loss

Each training example is a triple of the form (u, i, j), Where:

- (u, i) - positive pair
- (u, j) - unobserved pair

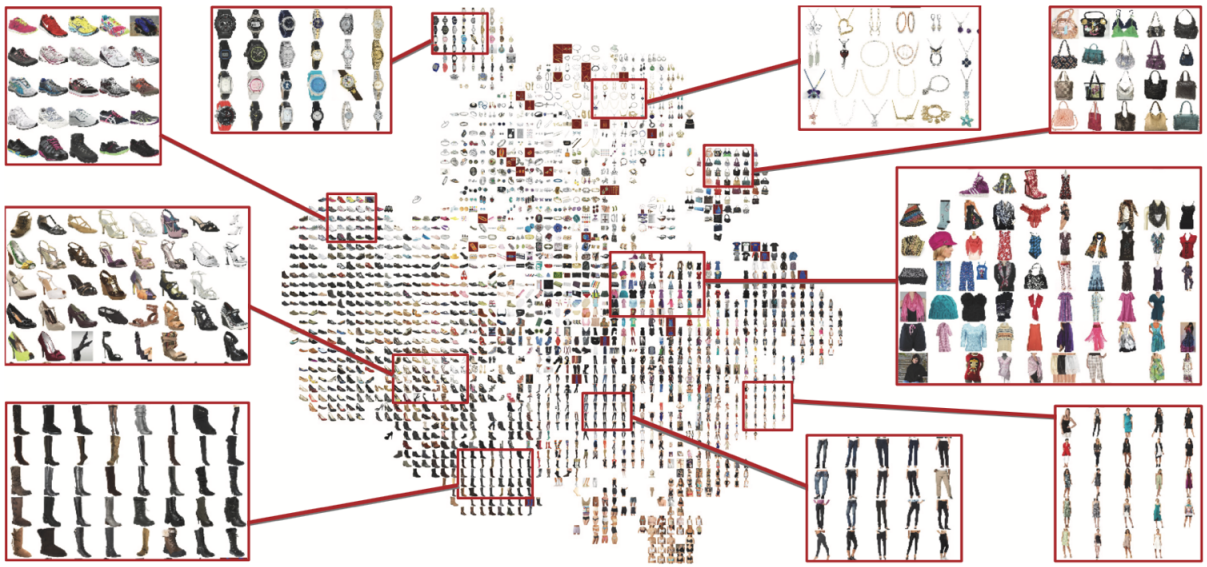
$\sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\theta \|\Theta\|^2$

where:

$\hat{x}_{uij} = \hat{x}_{u,i} - \hat{x}_{u,j}$

Image: Visual Bayesian Personalized Ranking (VBPR)

Results: VBPR improves on BPR-MF by over 12*Anecdote:* visual features show greater benefits on clothing than cellphone datasets.

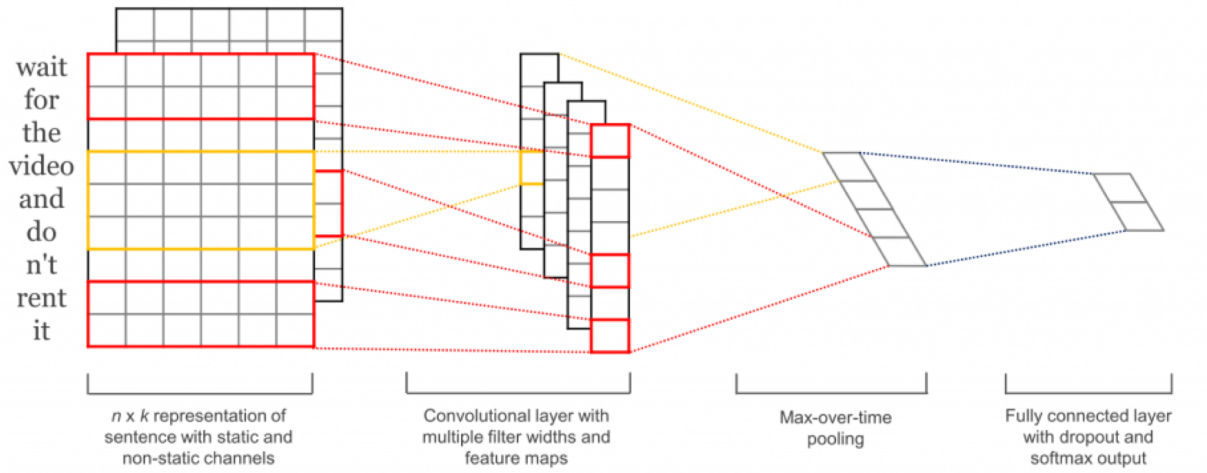


Visualizing the geometry of the product image embeddings

1D Convolutions for Text

1D Text Convolutions

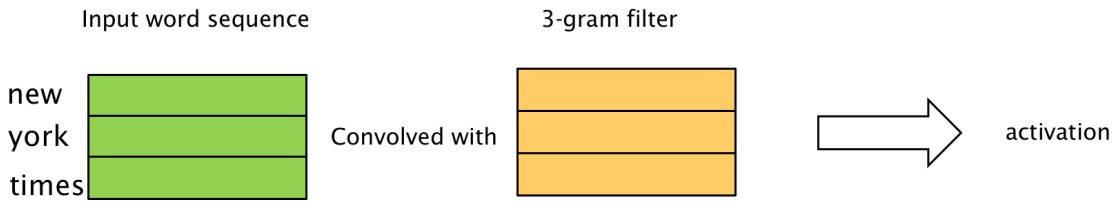
1D Convolutions The input of the convolution is still 2D dimensional: $nb_words x word_embedding_size$ In vision, the filters slide over local areas of an image, but in NLP the filters slide over words (represented as rows of the matrix) Even if the input is a 2D matrix, the geometry of the neighborhood is 1D (left right words). For this reason we denote these types of convolutions 1D.



The Anatomy of Text Convolutions - Image source:
http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/

1D Text Convolutions

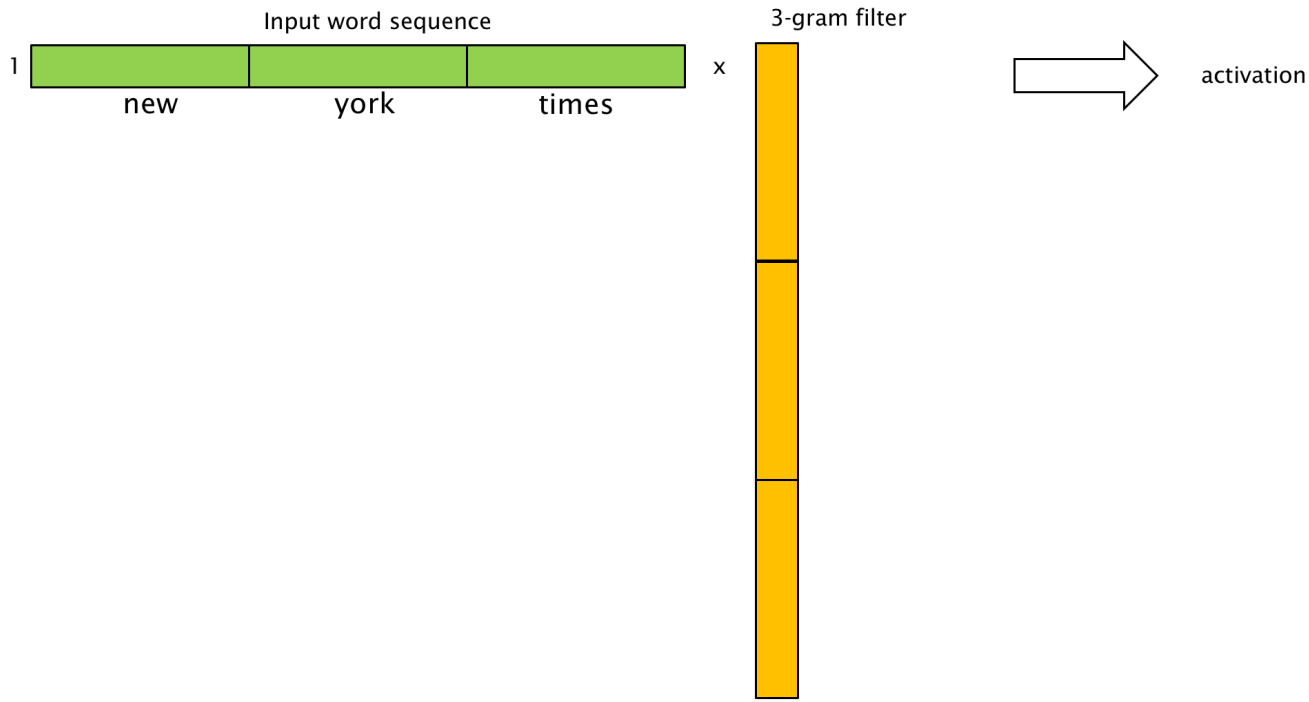
1D Convolutions Classical way of representing convolutions



Classical representation of 1D Convolution

1D Text Convolutions

1D Convolutions Alternative way that makes it more visible the 1Dimensional aspect of the operation



Alternative representation of 1D Convolution

Learning semantic representations using convolutional neural networks for web search

Text: Learning semantic representations using convolutional neural networks for web search.

Main ideas:

- Learn semantic vectors for search queries and Web documents by using CNNs on queries and document words
- Learn the representations in an end-to-end architecture that optimizes for the conditional likelihood of clicked documents given a query.

Text: Learning semantic representations using convolutional neural networks for web search.

Model: Convolutional Deep Structured Semantic Model (C-DSSM)

- The word hashing layer: each word is represented by a count vector of its letter trigrams. Option: word2vec.
- The 1D convolutional layer that takes word sequences and learns filters representing semantic clusters for the most useful k-grams.
- Max Pooling layer: extracts the most salient local k-grams.
- The final relevance score of a (D,Q) pair: the cosine similarity.

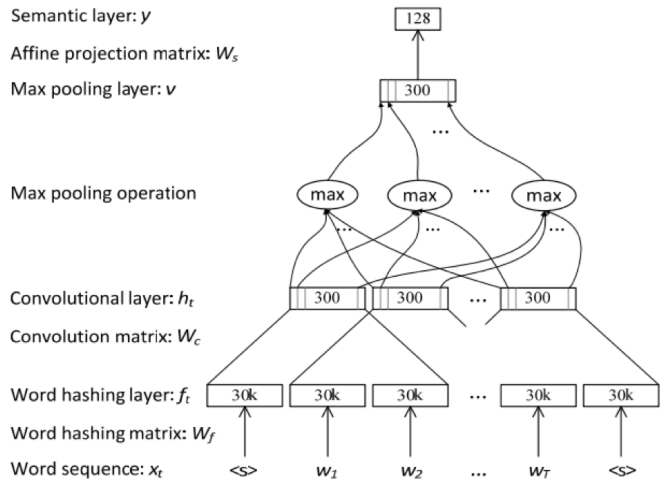


Figure 1: Illustration of the C-DSSM. A convolutional layer with the window size of three is illustrated.

C-DSSM Architecture

Text: Learning semantic representations using convolutional neural networks for web search.

Results: C-DSSM outperforms the non convolution-based previous state-of-the-art DSSM and other classical retrieval baselines such as BM25 and unigram and phrase-based language models in terms of NDCG.

Hashtag Recommendation Using Attention-Based Convolutional Neural Network

Text: Hashtag Recommendation Using Attention-Based Convolutional Neural Network[1]

Main idea: Use a modified text CNN architecture with attention in order to be able to use only some of the input words when computing the most likely hashtags recommendations.

Task: Predict which hashtag to use based on the input text of the document as an extreme multiclass problem.

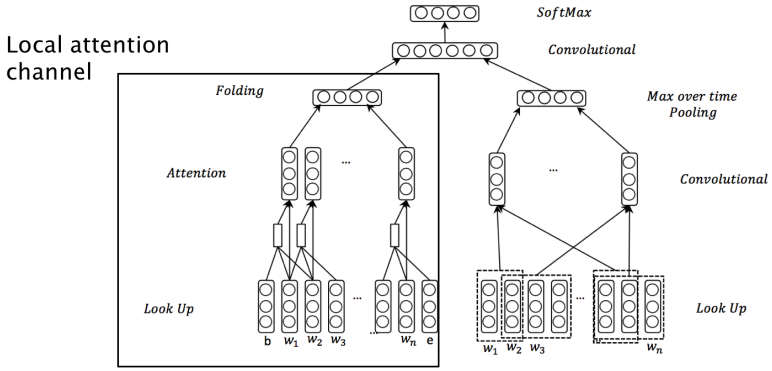


Figure 1: The architecture of the attention-based Convolutional Neural Network

Attention-based CNN Architecture

[1]Gong, Yuyun, and Qi Zhang. "Hashtag Recommendation Using Attention-Based Convolutional Neural Network." IJCAI. 2016.

Text: Hashtag Recommendation Using Attention-Based Convolutional Neural Network

Local Attention Channel:

Decision Task: Extract a subset of keywords/trigger words that can help with the selection of appropriate hashtags.

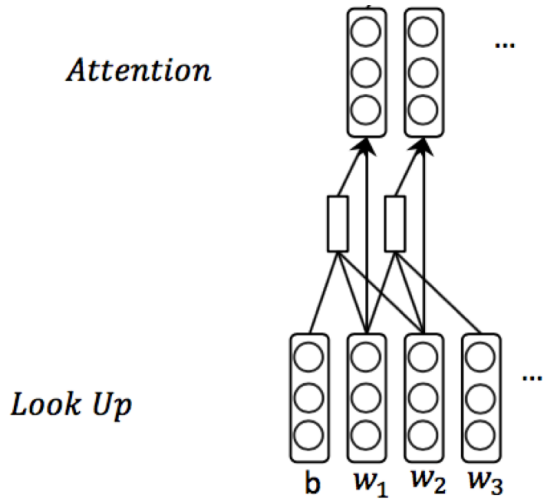
The local attention channel has the following 3 steps:

- Attention (decision/gating)
- Folding
- Compression/remapping

Text: Hashtag Recommendation Using Attention-Based Convolutional Neural Network

Local Attention Channel

The Attention Step:



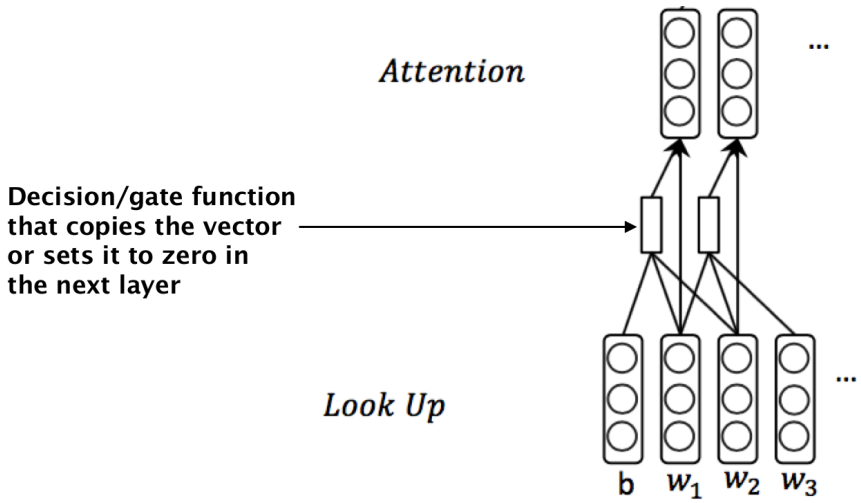
Local Attention Channel Module

Text: Hashtag Recommendation Using Attention-Based Convolutional Neural Network

Local Attention Channel

The Attention Step:

Decision/gate function that copies the vector or sets it to zero in the next layer



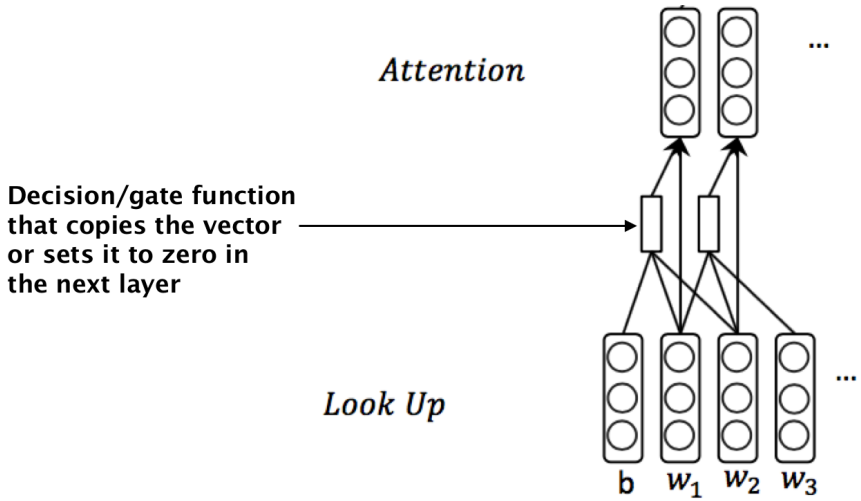
Local Attention Channel Module

Text: Hashtag Recommendation Using Attention-Based Convolutional Neural Network

Local Attention Channel

The Attention Step: Decision/gate function that copies the vector or sets it to zero in the next layer

The Decision function takes as input a word wi (e.g. w_1) and it surrounding words and evaluates its importance in the context by multiplying the context word matrix with a parametrized matrix M^{local} and passing it through a non-linear function $g(.)$



Text: Hashtag Recommendation Using Attention-Based Convolutional Neural Network

Local Attention Channel - Last 2 steps:

- Folding is the sum operation for each dimension of all the trigger words
- Compression/remapping takes the output of the folding operation and maps the resulting vector in a compressed space r

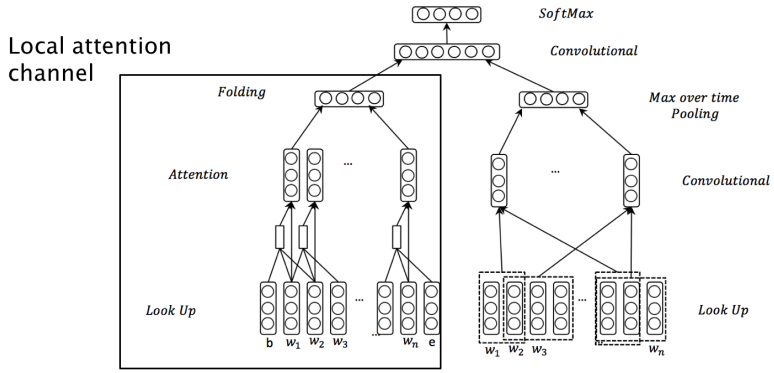


Figure 1: The architecture of the attention-based Convolutional Neural Network

Attention-based CNN Architecture

Text: Hashtag Recommendation Using Attention-Based Convolutional Neural Network

Results: The CNN+Attention significantly outperforms the previous state of the art CNN model on Precision/Recall metrics.

1D Convolutions for Music: Deep content-based music recommendation

1D Music Convolutions

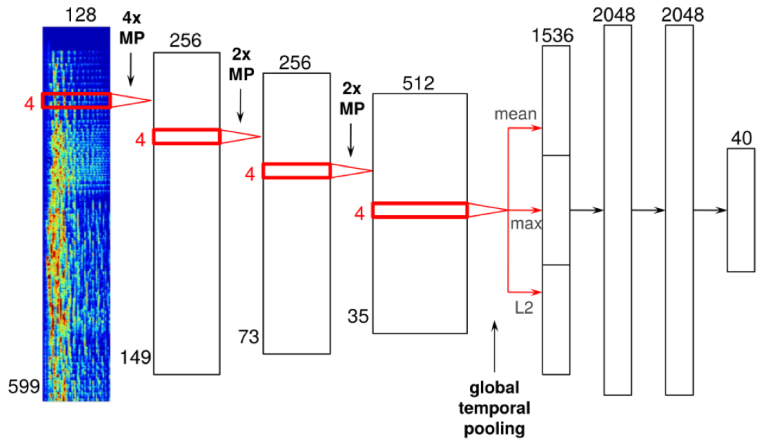
Main idea:

- Predict song preferences from audio signals
- The model is trained to map audio signal to pre-existing latent representations of songs that were obtained from a collaborative filtering model.

Task: The CNN is trained to minimize the mean squared error (MSE) between the audio-based predictions and the CF-based latent factor vectors.

1D Music Convolutions

Model: Paper: 2 convolutional + 2 fully connected layers. Shown below (blogpost on the paper): 4 convolutional layers and 3 fully connected layers

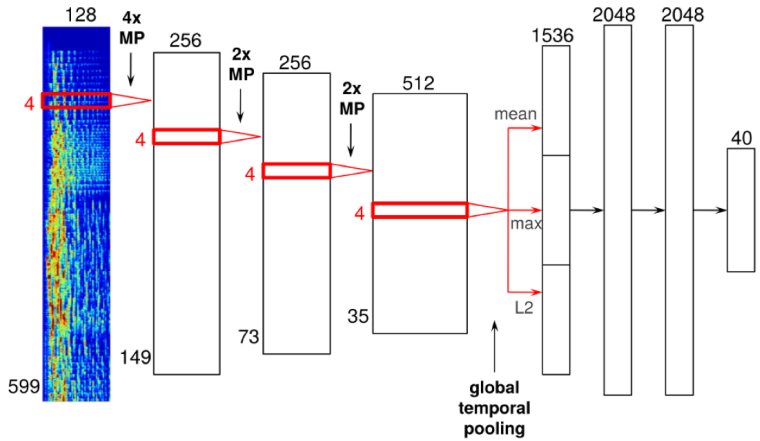


Music ConvNet Architecture - Image source: `http://benanne.github.io/2014/08/05/spotify-cnns.html`

1D Music Convolutions

Model:

- Input: mel spectrograms of 3 second audio fragments
- 1D convolutional layers: shown as red rectangles
- 1D: the convolution happens only in the time dimension, not in the frequency dimension.
- The activation function: ReLU max(0,x)



Music ConvNet Architecture

1D Music Convolutions

Model - pooling operators:

Max-pooling (MP): downsample the current input in time.

Global temporal pooling layer: pool across the entire time axis, effectively computing statistics of the learned features across time.

3 different global pooling functions:

- the mean
- the maximum
- the L2-norm

1D Music Convolutions

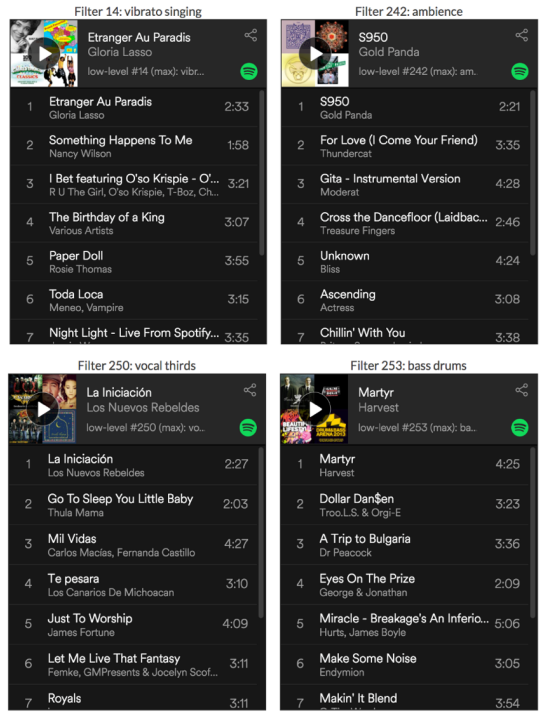
Results:

What are the filters learning?

- Playlist for low-level features (filters from the first convolutional layer)
- Each playlist is generated by finding songs that maximally activate a certain filter.

Example:

- Filter 14 - vibrato singing.
- Filter 242 - ringing ambience.
- Filter 250 - vocal thirds
- Filter 253 - bass drum sounds.

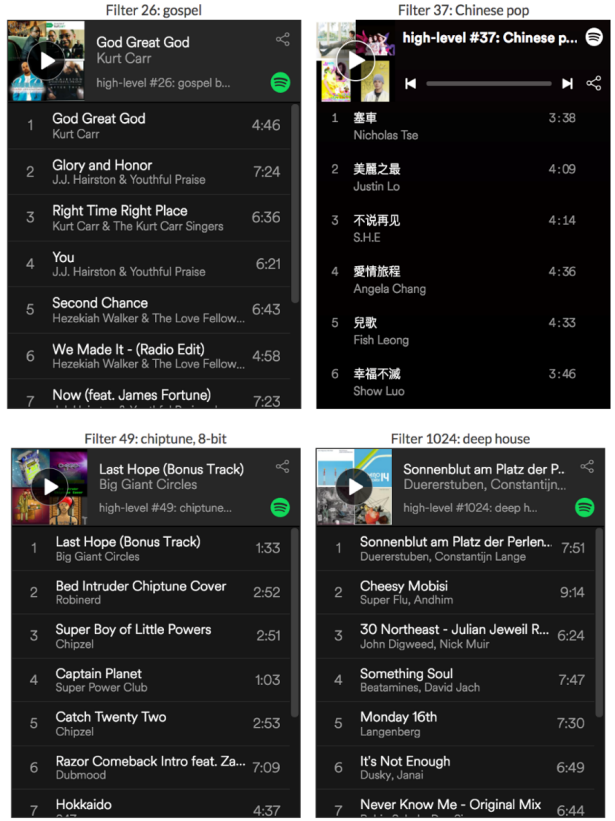


Low-level filters-based Reco - Image source: `http://benanne.github.io/2014/08/05/spotify-cnns.html`

1D Music Convolutions

Results: What are the filters learning?

- Playlist for high-level features
- Filters from the layer previous to the output
- The filters seem to pick certain subgenres.

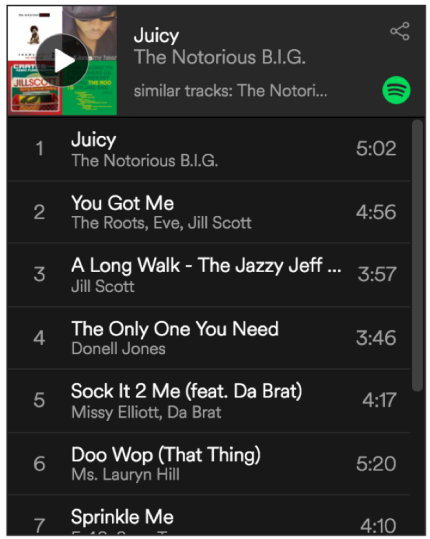


High-level filters-based Reco - Image source: `http://benanne.github.io/2014/08/05/spotify-cnns.html`

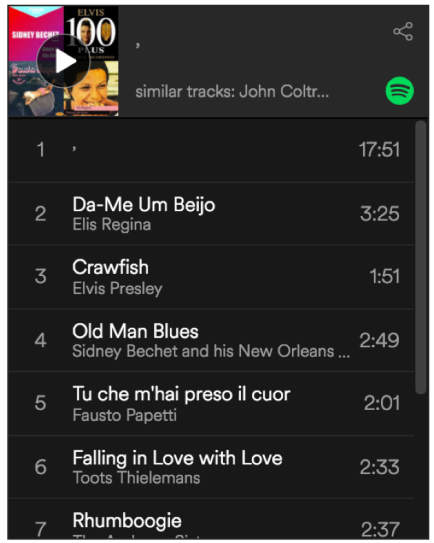
1D Music Convolutions

Results: Seed Song similarity playlists

The Notorious B.I.G. - Juicy (hip hop)



John Coltrane - My Favorite Things (jazz)



Song-based Reco - Image source: `http://benanne.github.io/2014/08/05/spotify-cnns.html`

1D Convolutions for Sequences: TemporalCNs (TCNs)

Sequences/Time: TemporalCNNs (TCNs)[1]

Main ideas: Sequence modeling does not have to be based on RNNs.

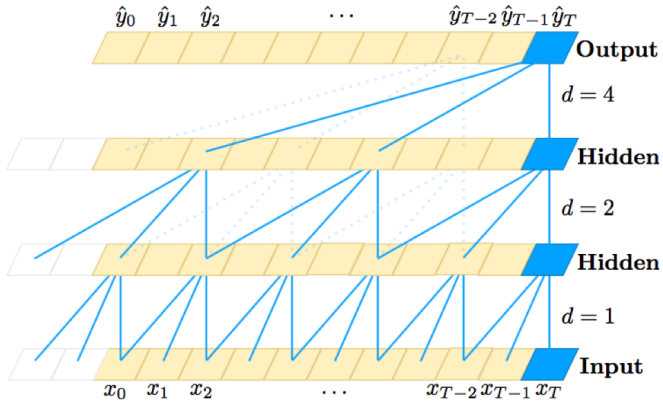
In fact, the authors show that for a variety of task temporal CNNs outperform RNNs and present other advantages.

[1] Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." arXiv preprint arXiv:1803.01271 (2018).

Sequences/Time: TemporalCNNs (TCNs)

Model:

- **Causal Convolutions:** Convolutions where an output at time t is convolved only with elements from time t or older.
- **Dilated Convolutions:** Causal convolutions that skip past states with dilation step d. By stacking dilated convolutions with exponentially increasing d, we can compress a very long history:



TCN Architecture

Sequences/Time: TemporalCNNs (TCNs)

TCNs vs RNNs:

Advantages of TCNs:

- Parallelism.
- Flexible (longer) receptive field size.
- Stable gradients. TCNs avoid the problem of exploding/vanishing gradients, which is a major issue for RNNs.
- Low memory requirement for training. TCNs use in practice an order of magnitude less memory than RNNs.

Disadvantages of TCNs:

- Data storage during evaluation. TCNs have to keep in memory the raw sequence, RNNs only the last hidden state
- Potential parameter change for a transfer of domain.

Sequences/Time: TemporalCNNs (TCNs)

Results: TCNs outperform RNNs in 10/11 of the benchmark tasks (seq2seq tasks such as translation are outside of the scope)

Merging multiple types of content - 3DCNNs: 3D Convolutional Networks for Session-based Recommendation with Content Features.

Merging multiple types of content: 3DCNN[1]

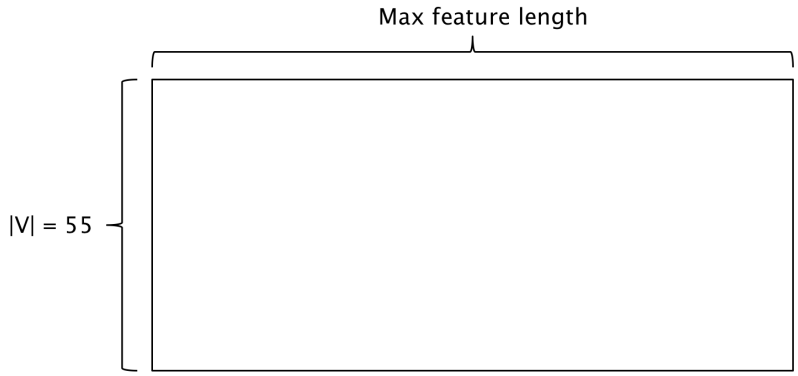
Main ideas: A method that combines user-based item sequence information and content features such as item description and categories to generate recommendations. Introduce the use of a 3DCNN with character-level encoding of all input data: the 3D architectures provide a natural way to capture spatio-temporal patterns the character-level network allows modeling different types of information.

The task: Predict what products will the users add to cart given its browsing history

[1]Refs: Tuan, Trinh Xuan, and Tu Minh Phuong. "3D Convolutional Networks for Session-based Recommendation with Content Features." Proceedings of the Eleventh ACM Conference on Recommender Systems. ACM, 2017

Merging multiple types of content: 3DCNN

Model: Character-level Representation of Input represent all item features, including item IDs using character-level encoding over a preset vocabulary V **Vocabulary V:** 55 characters, including all lower case characters from English alphabet, 10 digit characters, and several other characters

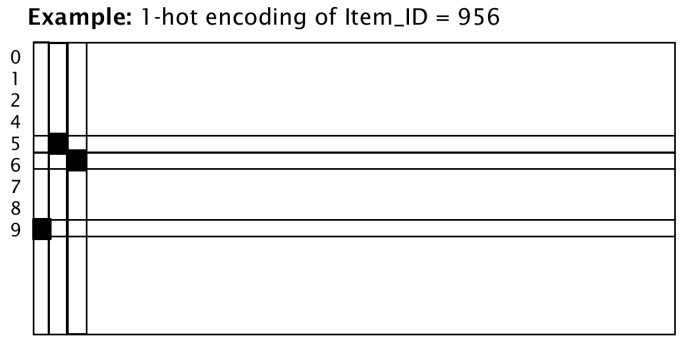


Character-level embeddings

Merging multiple types of content: 3DCNN

Model: Character-level Representation of Input represent all item features, including item IDs using character-level encoding over a preset vocabulary V

Vocabulary V: 55 characters, including all lower case characters from English alphabet, 10 digit characters, and several other characters



Character-level embeddings

Merging multiple types of content: 3DCNN

Model: 3DCNN Architecture

Input representation: For each event, we represent item id, text and category as three matrices. We stack the three matrices and obtain one frame matrix for each event. A sequence of events is represented by putting corresponding frames side by side. As a result, we get a cube-shaped input feature map.

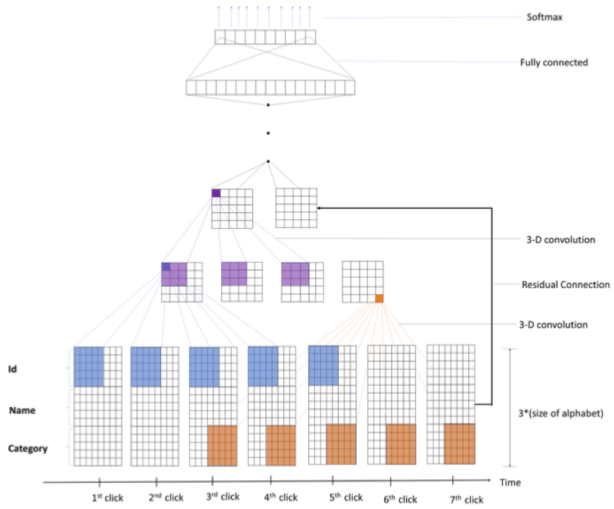


Figure 2: Illustration of the first and last layers.

3DCNN Architecture

Merging multiple types of content: 3DCNN

Results: As in the the TCN work, we see that CNN-based models can beat RNN models on sequential datasets by a significant margin.

CNN-Reco: Recap

Best practices

Other tricks:

- Attention
- Character-level encoding
- Skip connections / residual layers

Pros/Cons

Good properties:

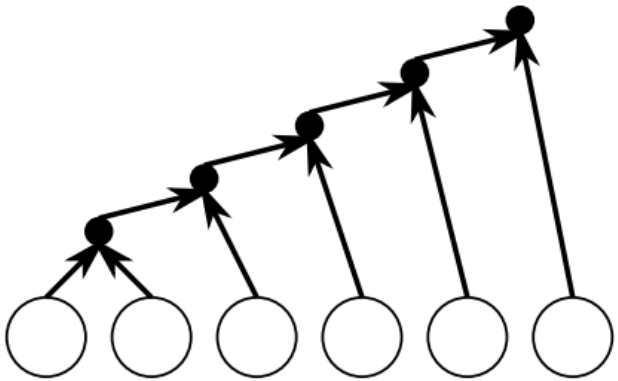
- Fast to train
- Share data, generalize

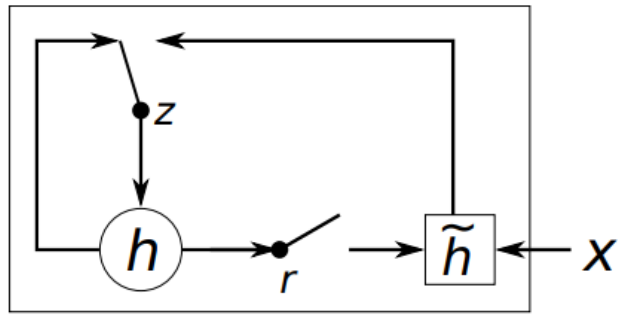
Limitations: Not rotation invariant Work on improving CNNs: CapsuleNets

User Modeling with RNNs for Recommendation

Recurrent Neural Networks

RNNs have a feedback loop in the hidden layer that lets them maintain information over time.





K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259,

Recurrent Neural Networks

Given a sequence $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$, the hidden state update is given by:

$$\mathbf{h}_t = g(W\mathbf{x}_t + U\mathbf{h}_{t-1}) \tag{1}$$

Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, J. Chung et al, NIPS 2014 Workshop on Deep Learning

Recurrent Neural Networks

The sequence probability is given by:

$$p(x_1, \cdots, x_T) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \cdots p(x_T \mid x_1, \cdots x_{T-1}) \quad (2)$$

where

$$p(x_t \mid x_1, x_2, \cdots x_t - 1) = g(h_t) \quad (3)$$

Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, J. Chung et al, NIPS 2014 Workshop on Deep Learning

The vanishing gradient problem

Even in very simple situations, gradient descent fails for long-term input/output dependencies

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. Transactions on Neural Networks, 1994.

The vanishing gradient problem

Two ways to solve it:

- use a better optimizer than SGD (e.g. clipped gradient)
- use a better activation function (recurrent unit = LSTM, gate recurrent unit = GRU)

Long Short Term Memory

- Standard RNNs apply a non-linear function to the weighted sum of input signal.
- LSTMs maintain a memory cell over time and can capture long-distance dependencies
- A set of gates controls when memory enters the memory cell (and when it is forgotten)

LSTMs in the real world

LSTMs are widely used in industry

- Apple uses LSTM for the "Quicktype" function on the iPhone and for Siri
- Google uses LSTM for voice search
- "Microsoft's speech recognition system is now as good as a human"

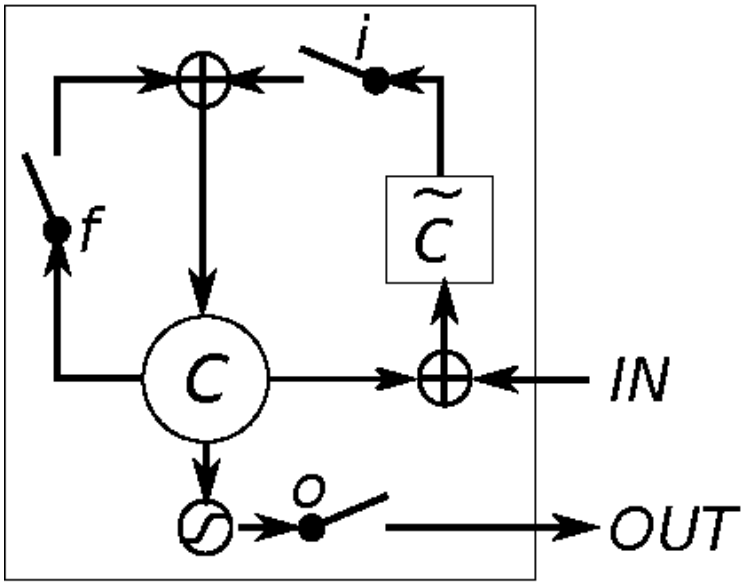
https://ai.googleblog.com/2015/08/the-neural-networks-behind-google-voice.html

Gated Recurrent Units

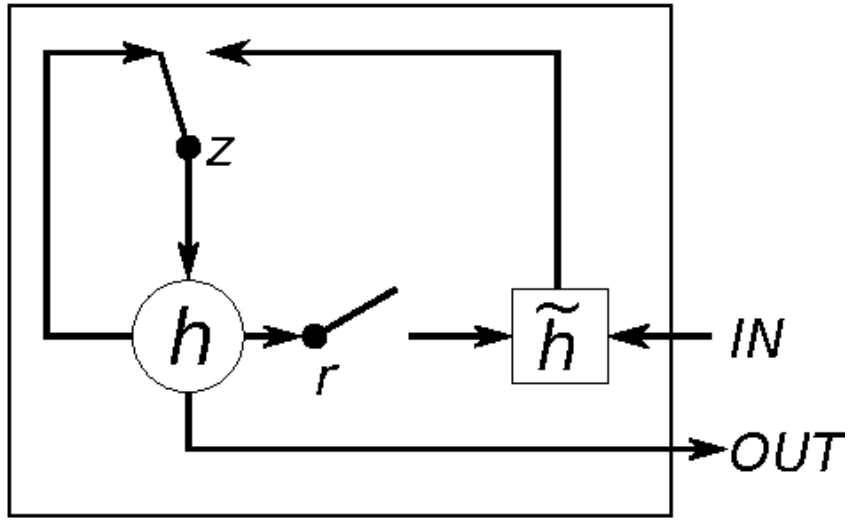
GRUs are similar to LSTMs but use a simpler architecture. They use fewer cells than LSTM and do not require a memory cell.

Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, J. Chung et al, NIPS 2014 Workshop on Deep Learning

GRU and LSTM

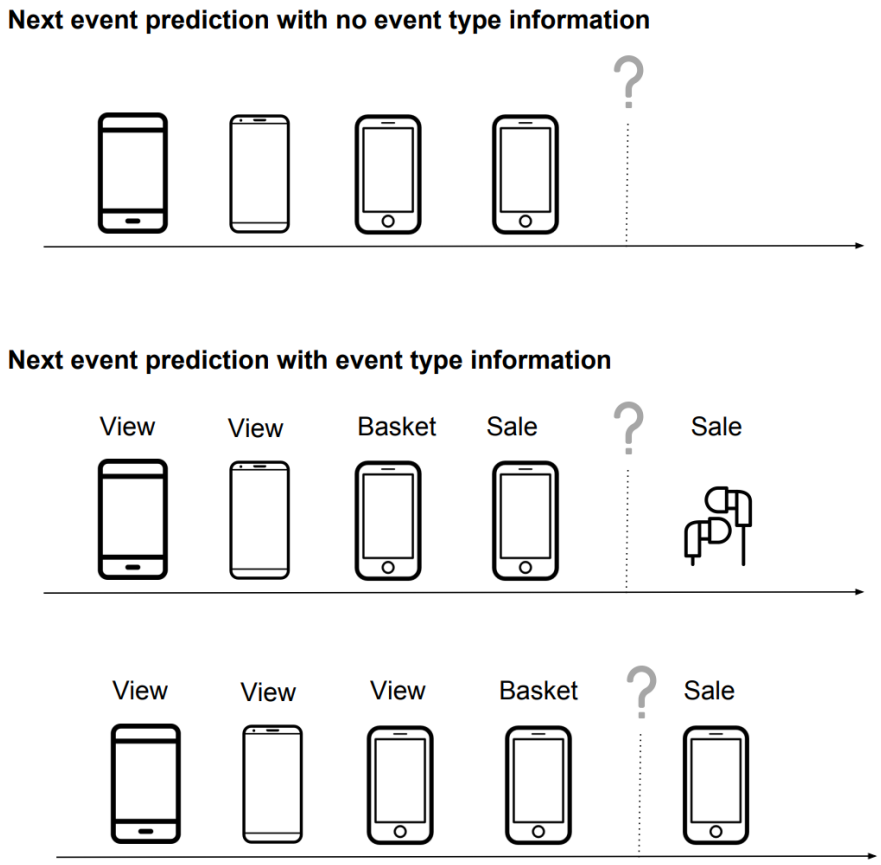


(a) Long Short-Term Memory



(b) Gated Recurrent Unit

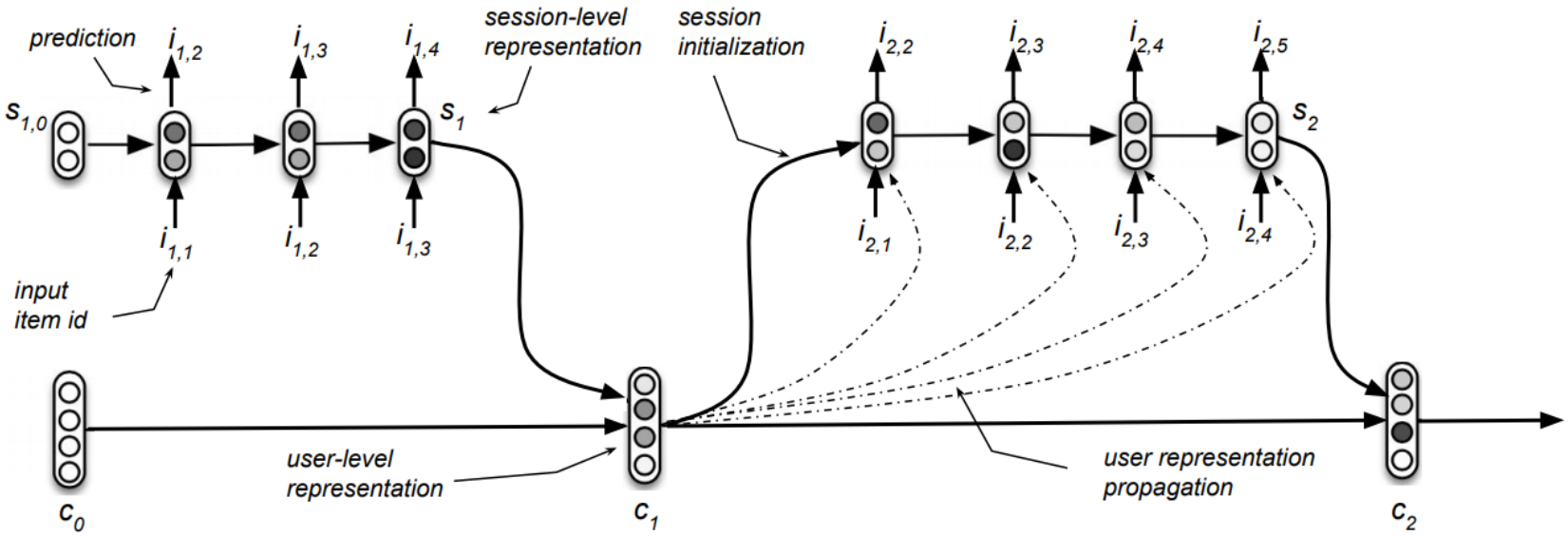
Contextual data is important



Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks, E. Smirnova F. Vasile, Deep Learning Workshop, RecSys 2017

Hierarchical RNNs

Combining a user-level GRU and a session-level GRU



Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks by Massimo Quadrana, Alexandros Karatzoglou, Balzs Hidasi, Paolo Cremonesi, RecSys 2017

Conclusions

- RNNs model temporal dependencies
- They are making their way into industrial applications
- They are harder to train than static neural nets

Thank You!