Advanced Topics in Recommender Systems

Isabelle Augenstein
augenstein@di.ku.dk
@IAugenstein
http://isabelleaugenstein.github.io/

Web Science Course
5 March 2019

UNIVERSITY OF COPENHAGEN

This Lecture

- Machine Learning Based Recommender Systems
- Latent User and Item Representations
- Brief Introduction to Representation Learning
- Context-Dependent Recommendation

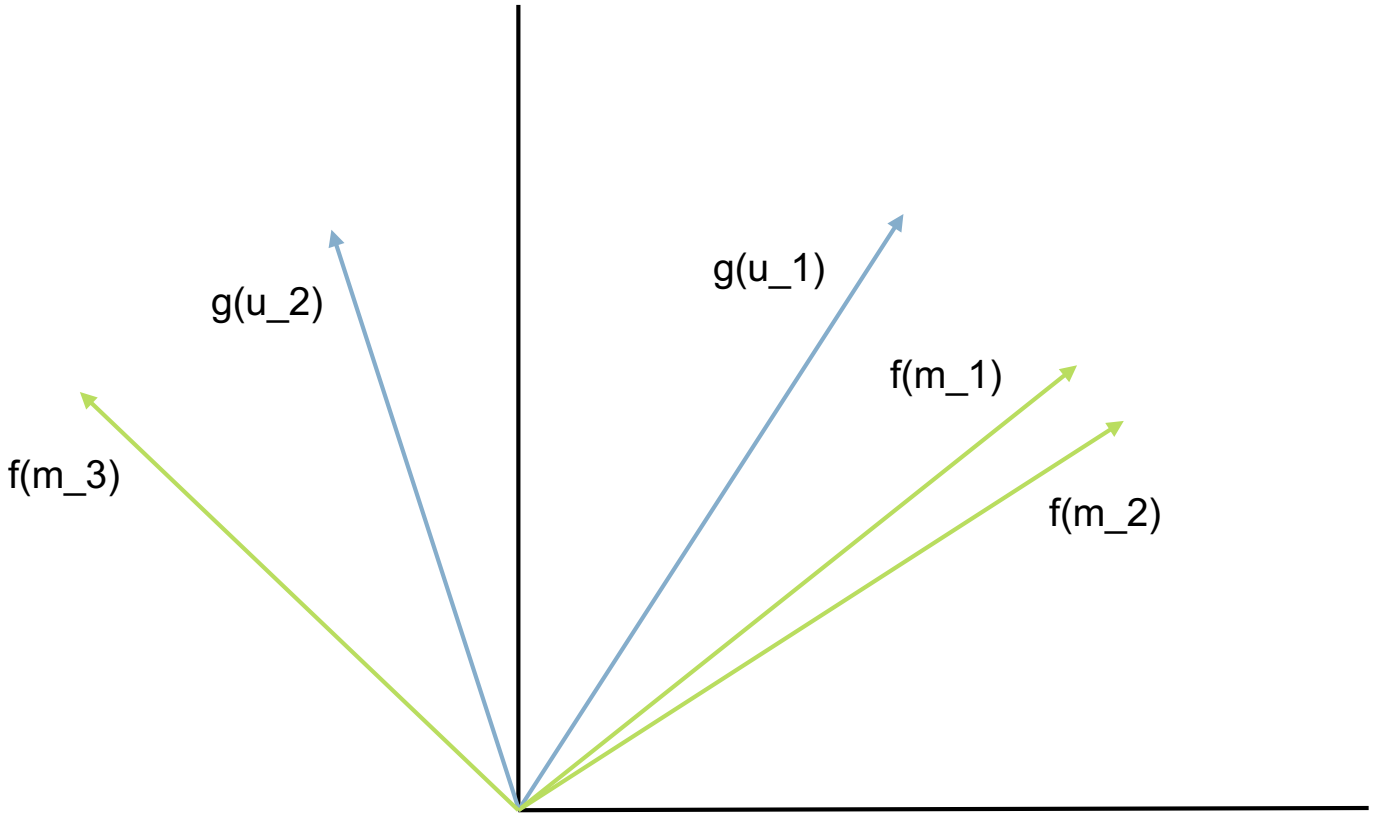
Reminder: Problem Setting

	Fargo	Splash	Titanic	Frozen	Juno	Gladiator
John	4				5	1
Mary	5		4			
Lars				2	4	5
Mette		3		3		

Utility / Ratings matrix: user-item pairs & their ratings

$$Y : U \times M \rightarrow R$$

Reminder: Content-Based Filtering



Distance Function: d(f(m_i), g(m_j) = d(x_i, x_j)

Reminder: User vs. Item-Based Collaborative Filtering

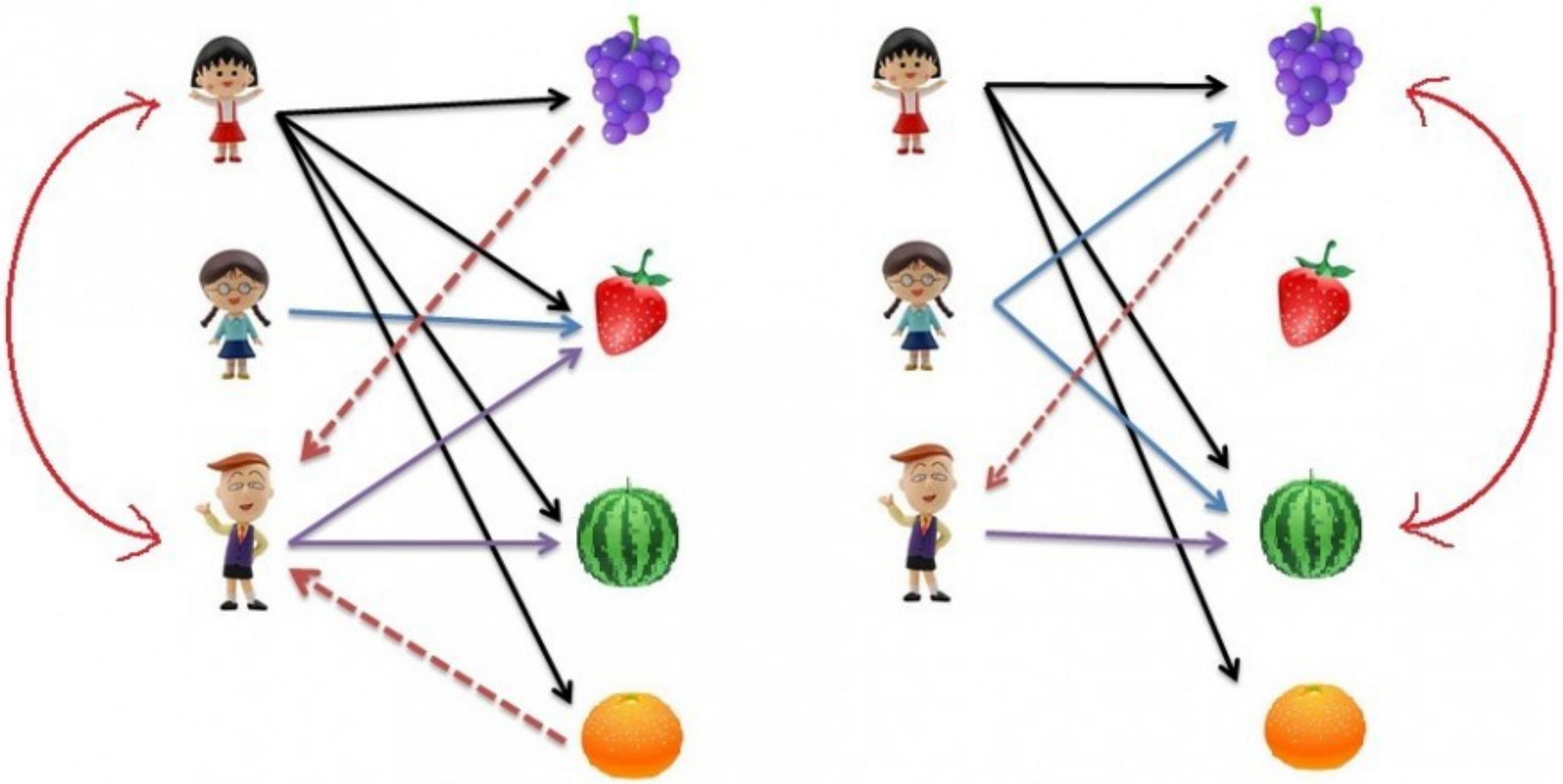


Image credit: https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f

Deep Learning for Recommender Systems

- Multilayer Perceptron (MLP)
- Autoencoder (AE)
- Convolutional Neural Network (CNN, ConvNet)
- Recurrent Neural Network (RNN)
- …

Deep Learning: Strengths & Weaknesses

Strengths & Weaknesses of Deep Learning?

https://tinyurl.com/y62v9whg

Deep Learning: Strengths & Weaknesses

Strengths:

- Nonlinear Transformations
- Representation Learning
- Sequence Modelling
- Flexibility

Weaknesses:

- Interpretability
- Data Requirements
- Architecture Search and Hyperparameter Tuning

Matrix Factorisation

- More effective than content-based or standard collaborative filtering
- Allows us to discover ***latent features*** underlying the interactions between users and items:
  - Two users would give high ratings to a certain movie if they both like the actors/actresses of the movie...
  - or if the movie is an action movie, which is a genre preferred by both users.
- Characterises both items and users by vectors of factors ***inferred from item rating patterns***.
- High correspondence between item and user factors leads to a recommendation.

Matrix Factorisation

Approach: find two (or more) matrices such that when you multiply them you will get back the original matrix.

$$R \approx P \times Q^T = \hat{R}$$

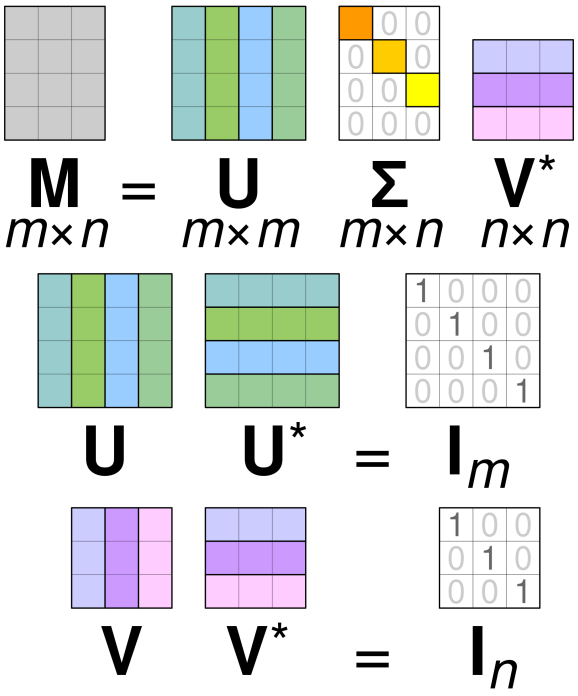
where:

- *R* is the user-item matrix
- *P* is a matrix where each row represents the strength of the associations between a *user* and the features.
- *Q* is a matrix where each row represents the strength of the associations between an *item* and the features.
- User *u*'s approximate rating of item *i* is then: $\hat{r}_{u,i} = p_u \cdot q_i^T$

Matrix Factorisation

Challenge: How to compute the factor vectors p_u and q_i^T ?

- We can use Singular Value Decomposition (SVD)



U: $m \; x \; m$ real or complex unitary matrix

V: $n \; x \; n$ real or complex unitary matrix.

Σ: rectangular diagonal matrix with non-negative real numbers on diagonal

σ_i: diagonal entries of Σ, *singular values* of **M**

Image from: https://en.wikipedia.org/wiki/Singular_value_decomposition

Matrix Factorisation

Challenge: How to compute the factor vectors p_u and q_i^T ?

- We can use Singular Value Decomposition (SVD)
- Conventional SVD is *undefined for matrices with missing values*.
- Using only the known entries is highly prone to over-fitting.
- Avoid over-fitting by using a regularised model:

$$\min_{q^{\cdot},p^{\cdot}} \sum_{(u,i)\in k} (r_{u,i} - p_u q_i^T)^2 + \lambda(\|p_u\|^2 + \|q_i\|^2)$$

- Minimised using e.g. stochastic gradient descent

Neural Matrix Factorisation

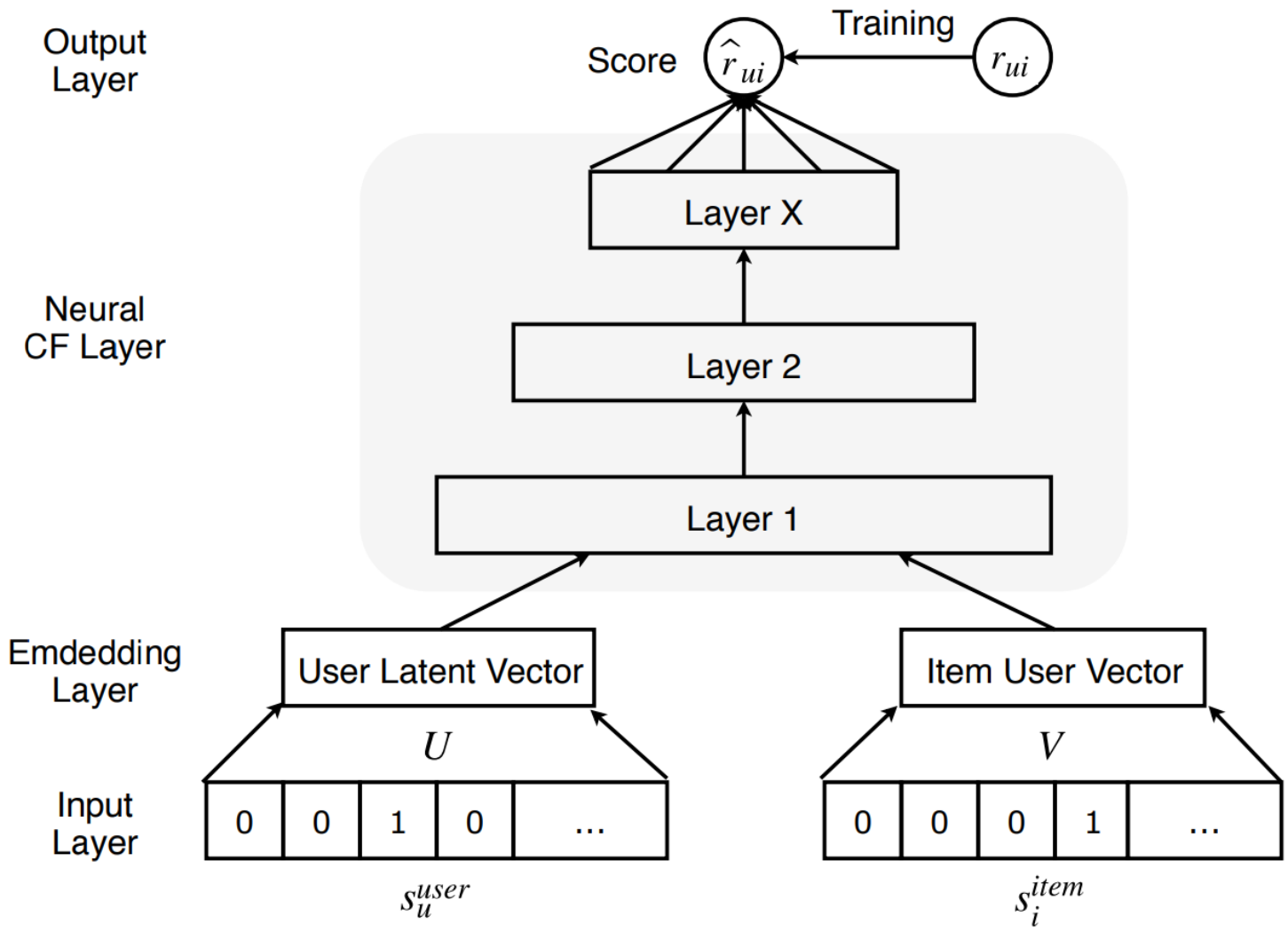
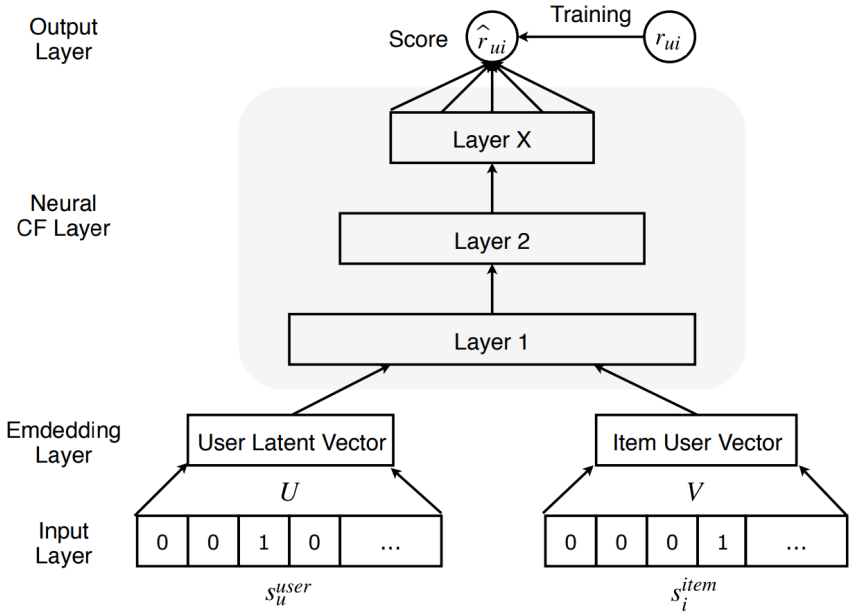


Image from: https://arxiv.org/abs/1707.07435

Neural Matrix Factorisation

s_u^{user} : user information
s_i^{item} : item information

$f(\cdot)$: MLP
θ : model parameters



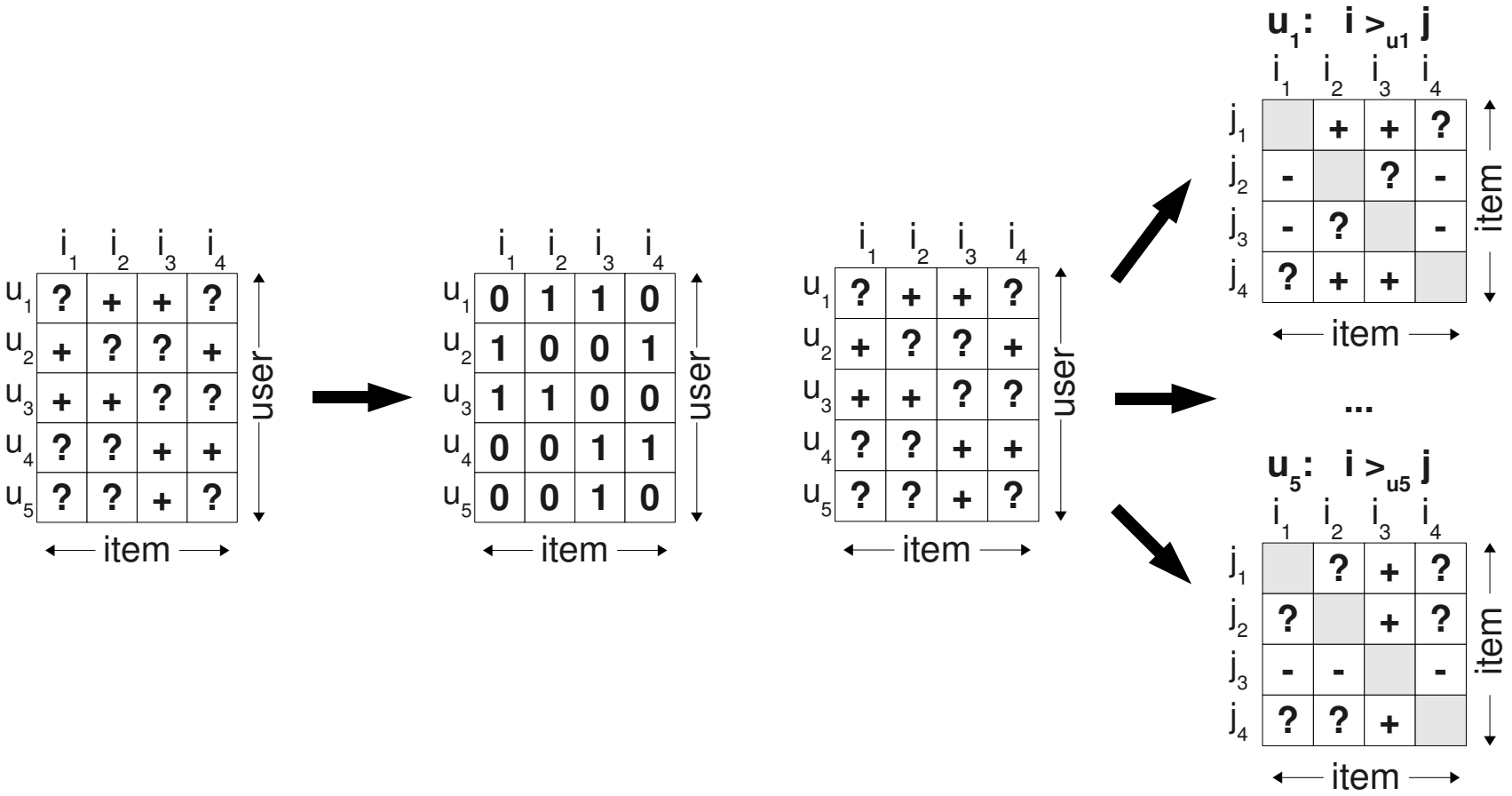
Scoring function: $\hat{r}_{u,i} = f\left(U^T \cdot s_u^{user}, V^T \cdot s_i^{item} \mid U, V, \theta\right)$
Logistic Loss:

$$\mathcal{L} = -\sum_{(u,i)\epsilon\mathcal{O}\cup\mathcal{O}^-} r_{u,i} \log \hat{r}_{u,i} + (1 - r_{u,i}) \log(1 - \hat{r}_{u,i})$$

Neural Matrix Factorisation

- Different ratings
  - Binary
  - Ordinal
  - Pairwise

Pairwise Ranking Loss Functions



Read more: https://arxiv.org/abs/1205.2618

Pairwise Ranking Loss Functions

Bayesian Pairwise Ranking (BPR):

$$\mathcal{L}_{bpr} = -\frac{1}{S}\sum_{j=1}^{S} \log \sigma\left(\hat{r}_{si} - \hat{r}_{sj}\right)$$

TOP1:

$$\mathcal{L}_{top1} = \frac{1}{S}\sum_{j=1}^{S} \sigma\left(\hat{r}_{sj} - \hat{r}_{si}\right) + \sigma\left(\hat{r}_{sj}^2\right)$$

S: sample size

$\hat{r}_{si}$, $\hat{r}_{sj}$: scores on negative item i / positive item j at session s

σ: logistic sigmoid function

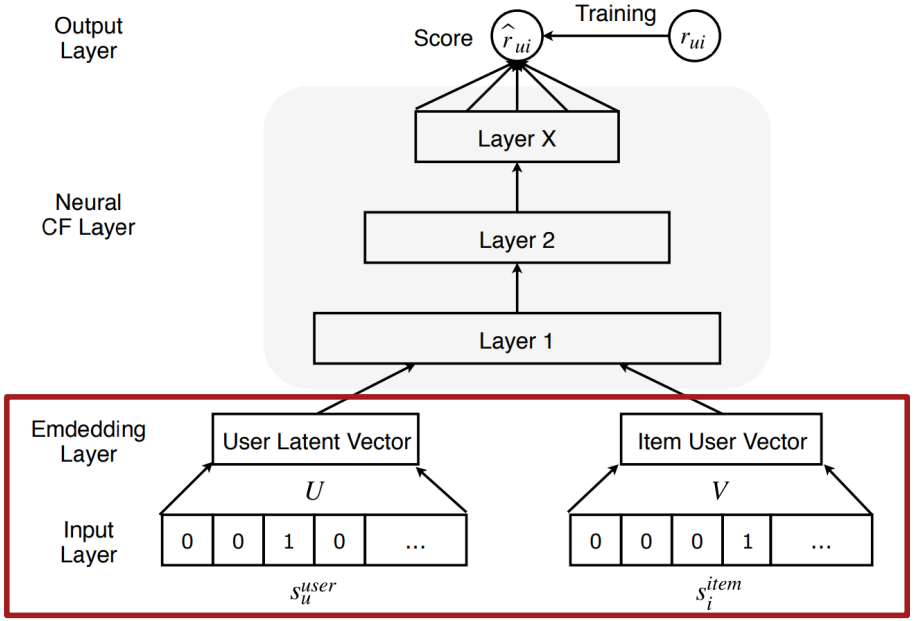
Read more: https://arxiv.org/abs/1706.03847

Neural Matrix Factorisation

- Exercise: exchange logistic loss for BPR + more

Representation Learning

- Raw input for user and item vectors?
  - User / item name
  - Content, e.g. text
  - Meta-data
  - …
- How to learn vectors
  - Directly from input
  - Unsupervised pre-training



Deep Learning for NLP

(without Magic)

Richard Socher and Christopher Manning

Stanford University

NAACL 2013, Atlanta

http://nlp.stanford.edu/courses/NAACL2013/

*with a big thank you to Yoshua Bengio, with whom we participated in the previous ACL 2012 version of this tutorial

#1 Learning representations

Handcrafting features is time-consuming

The features are often both over-specified and incomplete

The work has to be done again for each task/domain/…

We must move beyond handcrafted features and simple ML

Humans develop representations for learning and reasoning

Our computers should do the same

Deep learning provides a way of doing this

Part 1.3: The Basics

Word Representations

The standard word representation

The vast majority of rule-based **and** statistical NLP work regards words as atomic symbols: hotel, conference, walk

In vector space terms, this is a vector with one 1 and a lot of zeroes

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

Dimensionality: 20K (speech) – 50K (PTB) – 500K (big vocab) – 13M (Google 1T)

We call this a "one-hot" representation. Its problem:

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

Distributional similarity based representations

You can get a lot of value by representing a word by means of its neighbors

> "You shall know a word by the company it keeps"
>
> (J. R. Firth 1957: 11)

One of the most successful ideas of modern statistical NLP

government debt problems turning into	banking	crises as has happened in
saying that Europe needs unified	banking	regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

You can vary whether you use local or large context to get a more syntactic or semantic clustering

Class-based (hard) and soft clustering word representations

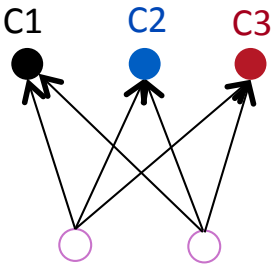
Class based models learn word classes of similar words based on distributional information (~ class HMM)

- Brown clustering (Brown et al. 1992)
- Exchange clustering (Martin et al. 1998, Clark 2003)
- Desparsification and great example of unsupervised pre-training

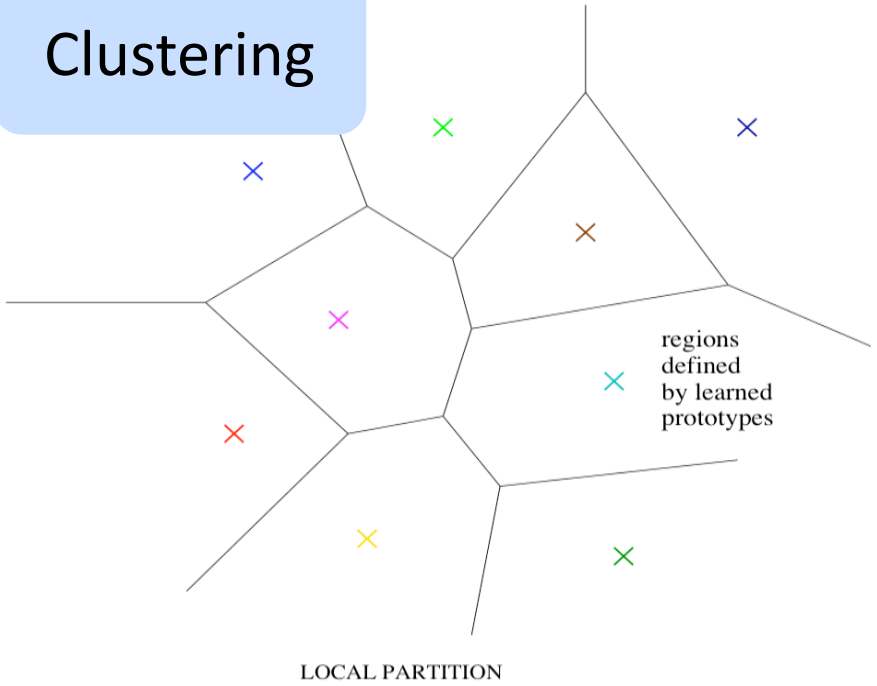
Soft clustering models learn for each cluster/topic a distribution over words of how likely that word is in each cluster

- Latent Semantic Analysis (LSA/LSI), Random projections
- Latent Dirichlet Analysis (LDA), HMM clustering

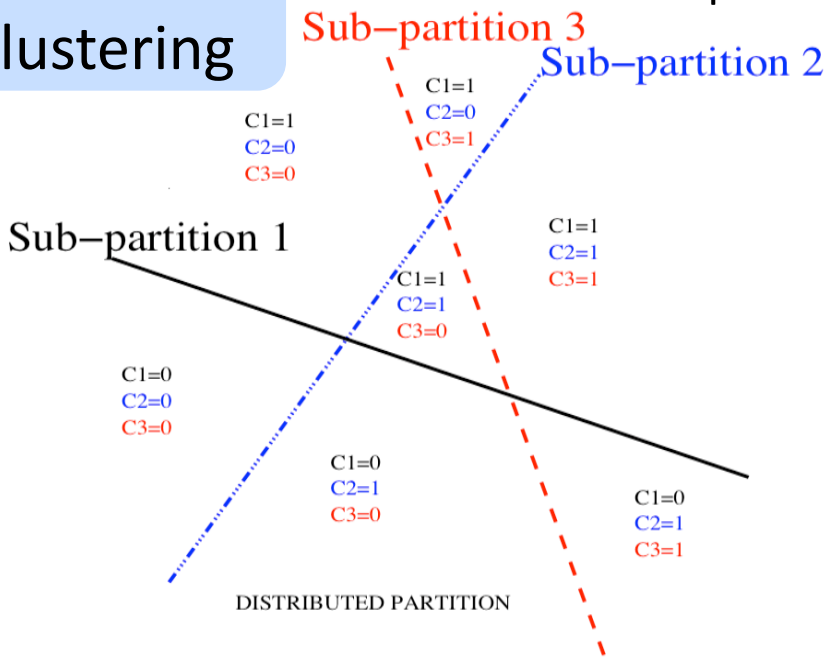
#2 The need for distributed representations



Clustering



Multi-Clustering



Learning features that are not mutually exclusive can be exponentially more efficient than nearest-neighbor-like or clustering-like models

Neural word embeddings as a distributed representation

Similar idea

Combine vector space semantics with the prediction of probabilistic models (Bengio et al. 2003, Collobert & Weston 2008, Turian et al. 2010)

In all of these approaches, including deep learning models, a word is represented as a dense vector

linguistics =

$$\begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Neural word embeddings - visualization

Stunning new result at this conference! Mikolov, Yih & Zweig (NAACL 2013)

These representations are *way* better at encoding dimensions of similarity than we realized!

- Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space

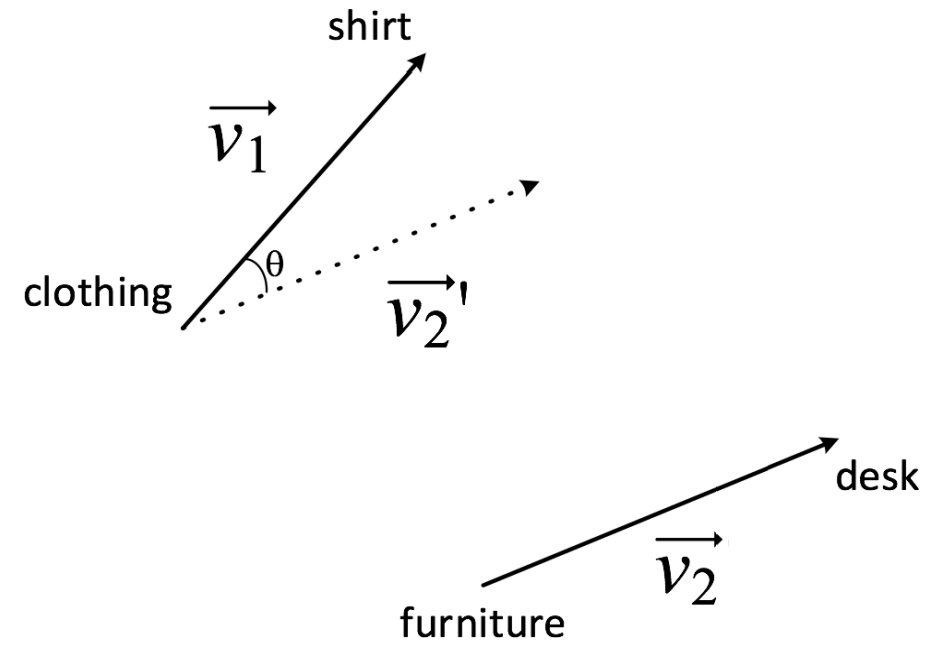
  Syntactically

  - $x_{apple} - x_{apples} \approx x_{car} - x_{cars} \approx x_{family} - x_{families}$
  - Similarly for verb and adjective morphological forms

  Semantically (Semeval 2012 task 2)

  - $x_{shirt} - x_{clothing} \approx x_{chair} - x_{furniture}$

Stunning new result at this conference! Mikolov, Yih & Zweig (NAACL 2013)



Method	Syntax % correct
LSA 320 dim	16.5 [best]
RNN 80 dim	16.2
RNN 320 dim	28.5
RNN 1600 dim	39.6
Method	**Semantics Spearm ρ**
UTD-NB (Rink & H. 2012)	0.230 [Semeval win]
LSA 640	0.149
RNN 80	0.211
RNN 1600	0.275 [new SOTA]

Advantages of the neural word embedding approach

Compared to a method like LSA, neural word embeddings can become more meaningful through adding supervision from one or multiple tasks

- "Discriminative fine-tuning"
- For instance, sentiment is usually not captured in unsupervised word embeddings but can be in neural word vectors

We can build representations for large linguistic units

- See part 2

Part 1.4: The Basics

Unsupervised word vector learning

A neural network for learning word vectors (Collobert et al. JMLR 2011)

Idea: A word and its context is a positive training sample; a random word in that same context gives a negative training sample:

cat chills on a mat

cat chills Jeju a mat

Similar: Implicit negative evidence in Contrastive Estimation, (Smith and Eisner 2005)

A neural network for learning word vectors

How do we formalize this idea? Ask that

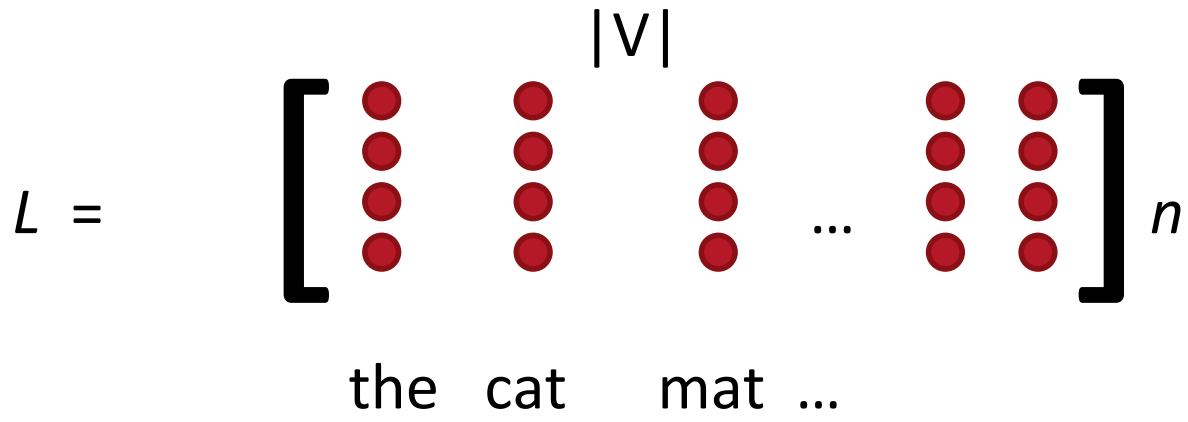
score(cat chills on a mat) > score(cat chills Jeju a mat)

How do we compute the score?

- With a neural network
- Each word is associated with an *n*-dimensional vector

Word embedding matrix

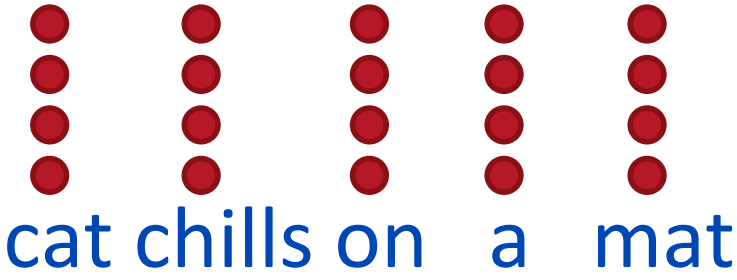
- Initialize all word vectors randomly to form a word embedding matrix $L \in \mathbb{R}^{n \times |V|}$



- These are the word features we want to learn
- Also called a look-up table
  - Conceptually you get a word's vector by left multiplying a one-hot vector *e* by *L*: $x = Le$

Word vectors as input to a neural network

- score(cat chills on a mat)
- To describe a phrase, retrieve (via index) the corresponding vectors from *L*



- Then concatenate them to 5*n* vector:
- *x* =[●●●● ●●●● ●●●● ●●●● ●●●●]
- How do we then compute score(x)?

A Single Layer Neural Network

- A single layer was a combination of a linear layer and a nonlinearity:

$$z = Wx + b$$
$$a = f(z)$$

- The neural activations *a* can then be used to compute some function
- For instance, the score we care about:

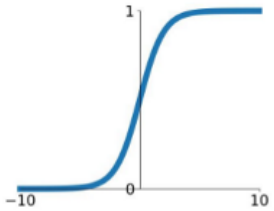
$$score(x) = U^T a \in \mathbb{R}$$

Non-Linear Activation Functions

Activation Functions

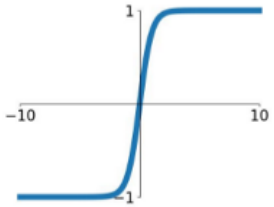
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



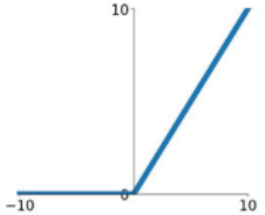
tanh

$$\tanh(x)$$



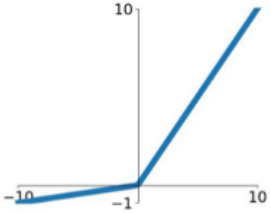
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$



Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

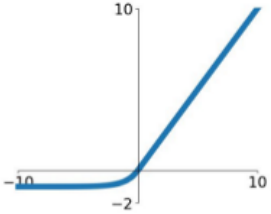


Image from: https://medium.com/@shrutijadon10104776/survey-on-activation-functions-for-deep-learning-9689331ba092

Summary: Feed-forward Computation

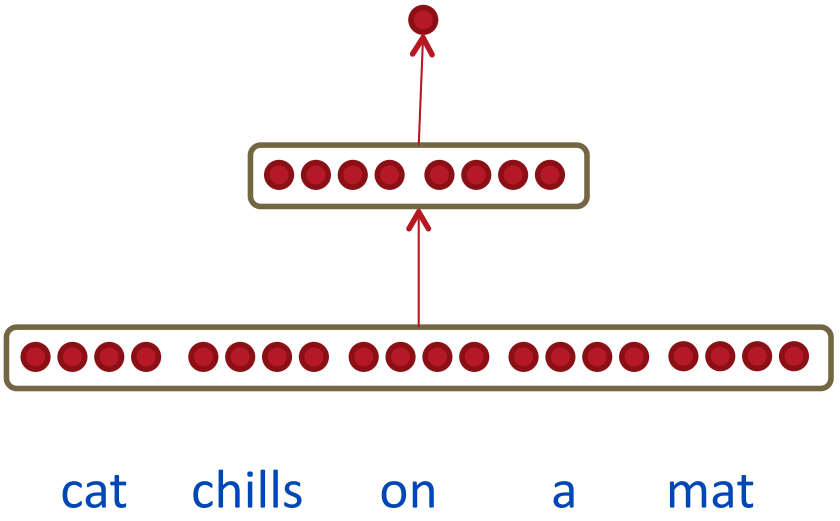
Computing a window's score with a 3-layer Neural Net: *s = score*(cat chills on a mat)

$$s = U^T f(Wx + b) \qquad x \in \mathbb{R}^{20\times 1}, W \in \mathbb{R}^{8\times 20}, U \in \mathbb{R}^{8\times 1}$$

$$\begin{aligned} s &= U^T a \\ a &= f(z) \\ z &= Wx + b \end{aligned}$$

$$x = [x_{cat} \quad x_{chills} \quad x_{on} \quad x_a \quad x_{mat}]$$

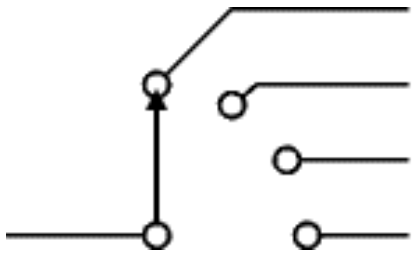
$$L \in \mathbb{R}^{n\times|V|}$$



Summary: Feed-forward Computation

- s = score(cat chills on a mat)
- s_c = score(cat chills Jeju a mat)
- Idea for training objective: make score of true window larger and **c**orrupt window's score lower (until they're good enough): minimize

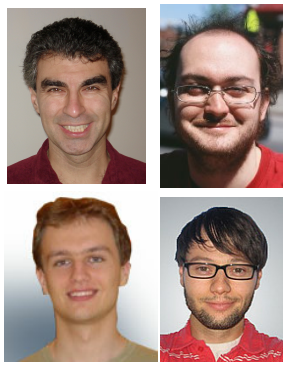
$$J = \max(0, 1 - s + s_c)$$



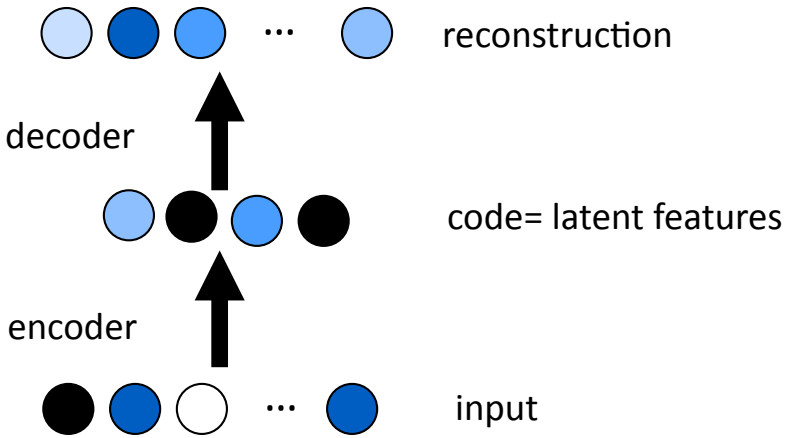
- This is continuous, can perform SGD

Auto-Encoders

- Multilayer neural net with target output = input
- Reconstruction=decoder(encoder(input))



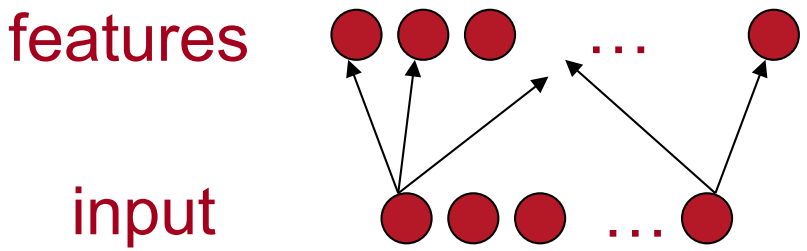
$$
\begin{aligned}
a &= \tanh(Wx + b) \\
x' &= \tanh(W^T a + c) \\
cost &= ||x' - x||^2
\end{aligned}
$$



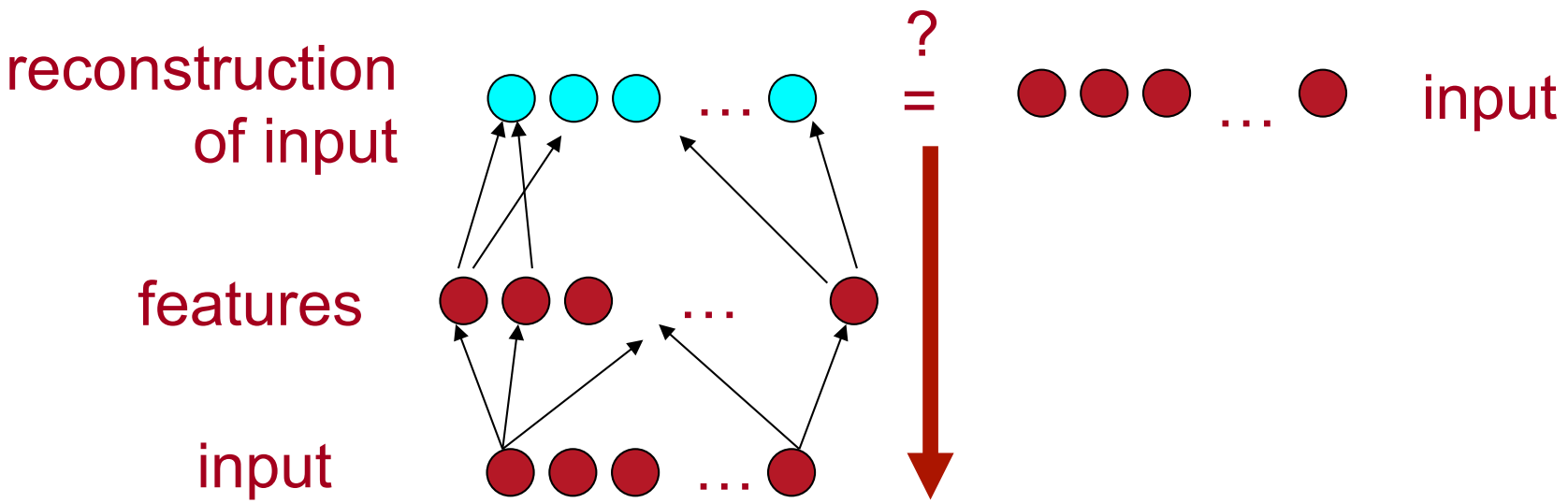
- Probable inputs have small reconstruction error

Layer-wise Unsupervised Learning

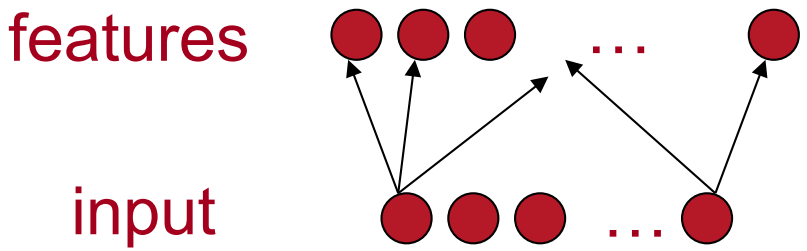
Layer-wise Unsupervised Pre-training



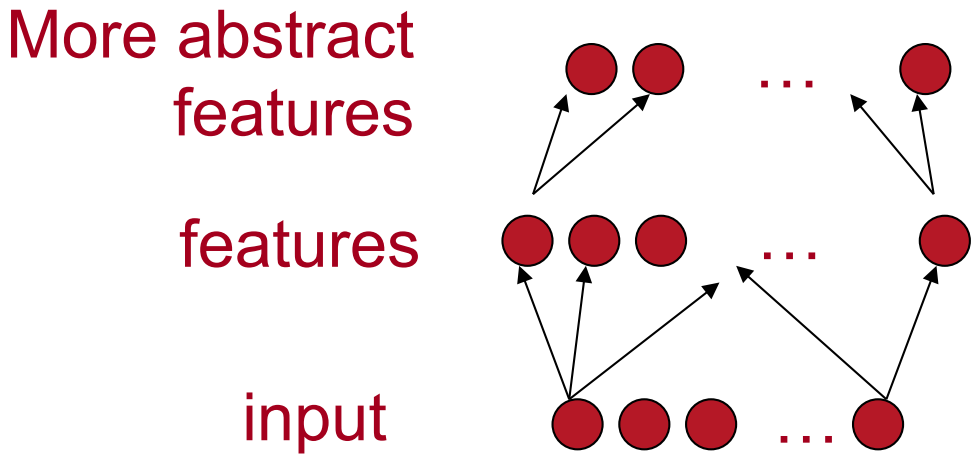
Layer-wise Unsupervised Pre-training



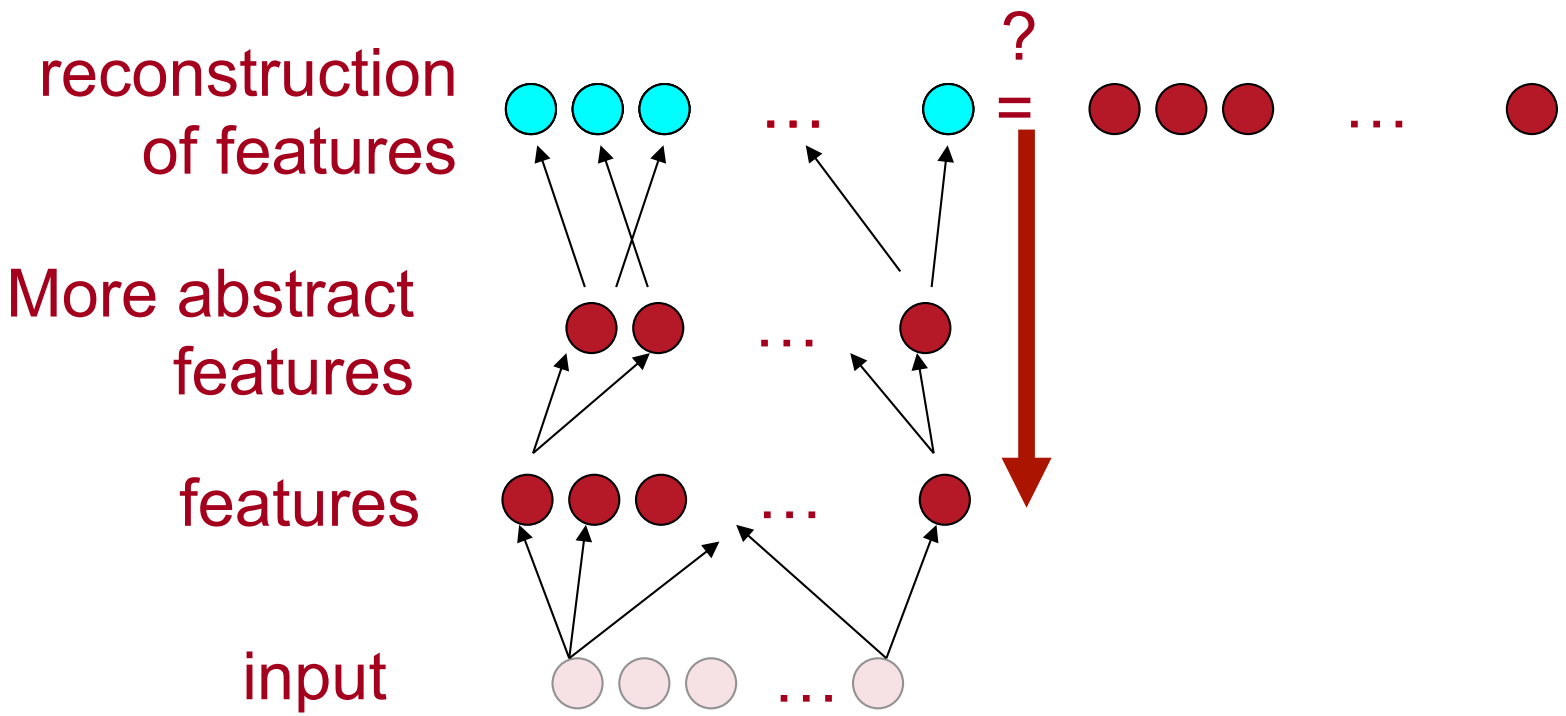
Layer-wise Unsupervised Pre-training



Layer-wise Unsupervised Pre-training



Layer-wise Unsupervised Learning

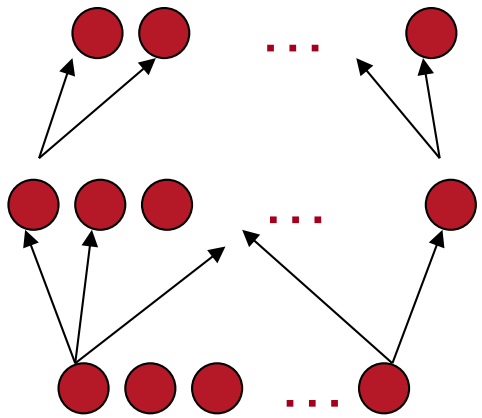


Layer-wise Unsupervised Pre-training

More abstract features

features

input



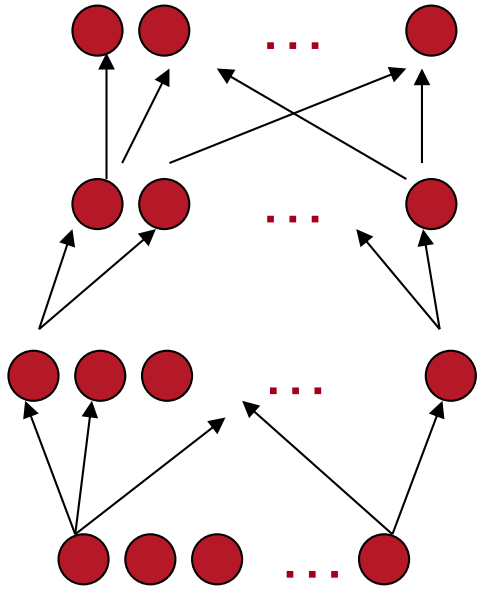
Layer-wise Unsupervised Learning

Even more abstract features

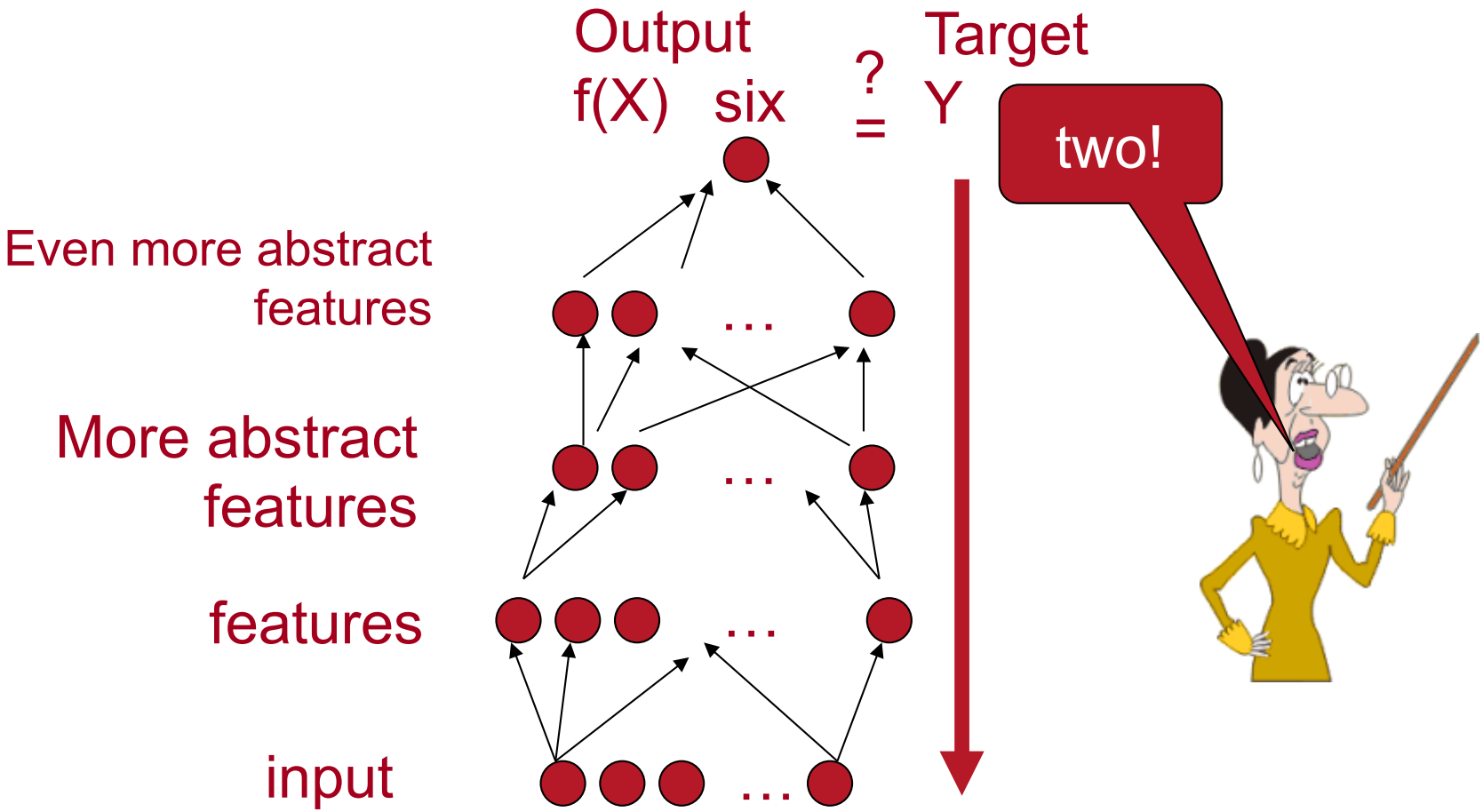
More abstract features

features

input

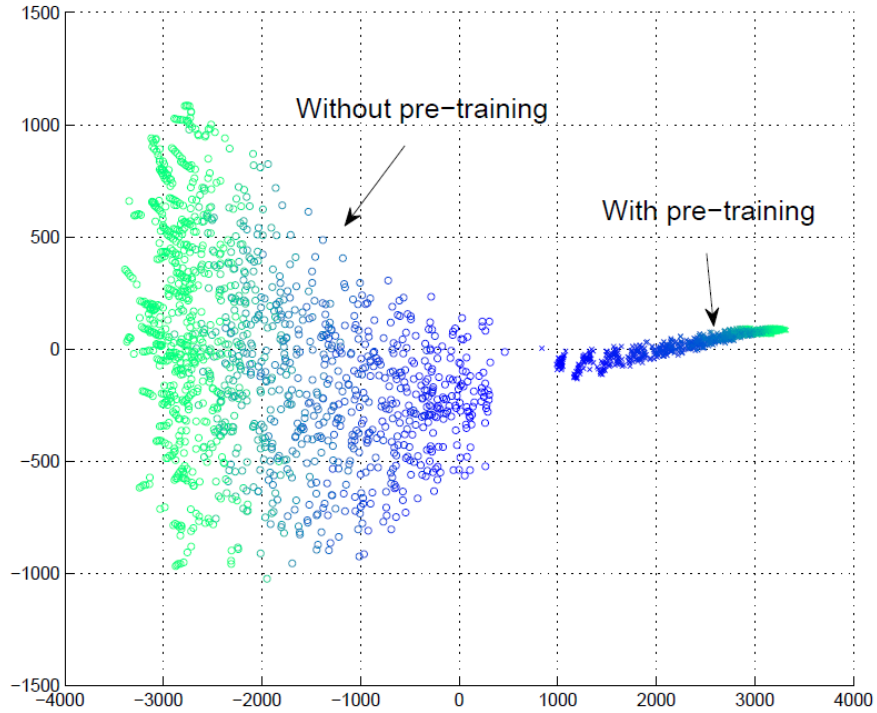


Supervised Fine-Tuning



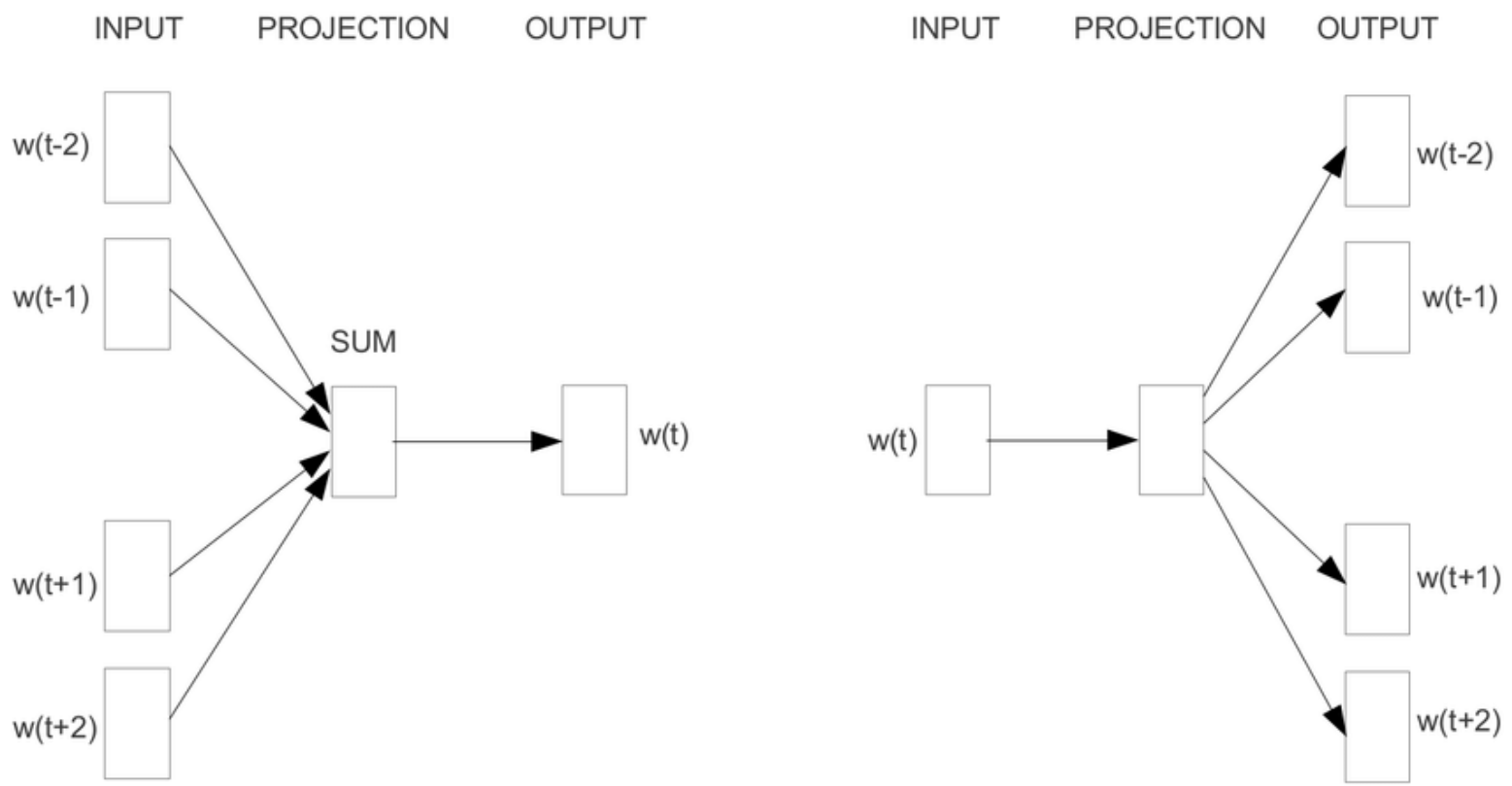
Why is unsupervised pre-training working so well?

- Regularization hypothesis:
  - Representations good for P(x) are good for P(y|x)
- Optimization hypothesis:
  - Unsupervised initializations start near better local minimum of supervised training error
  - Minima otherwise not achievable by random initialization



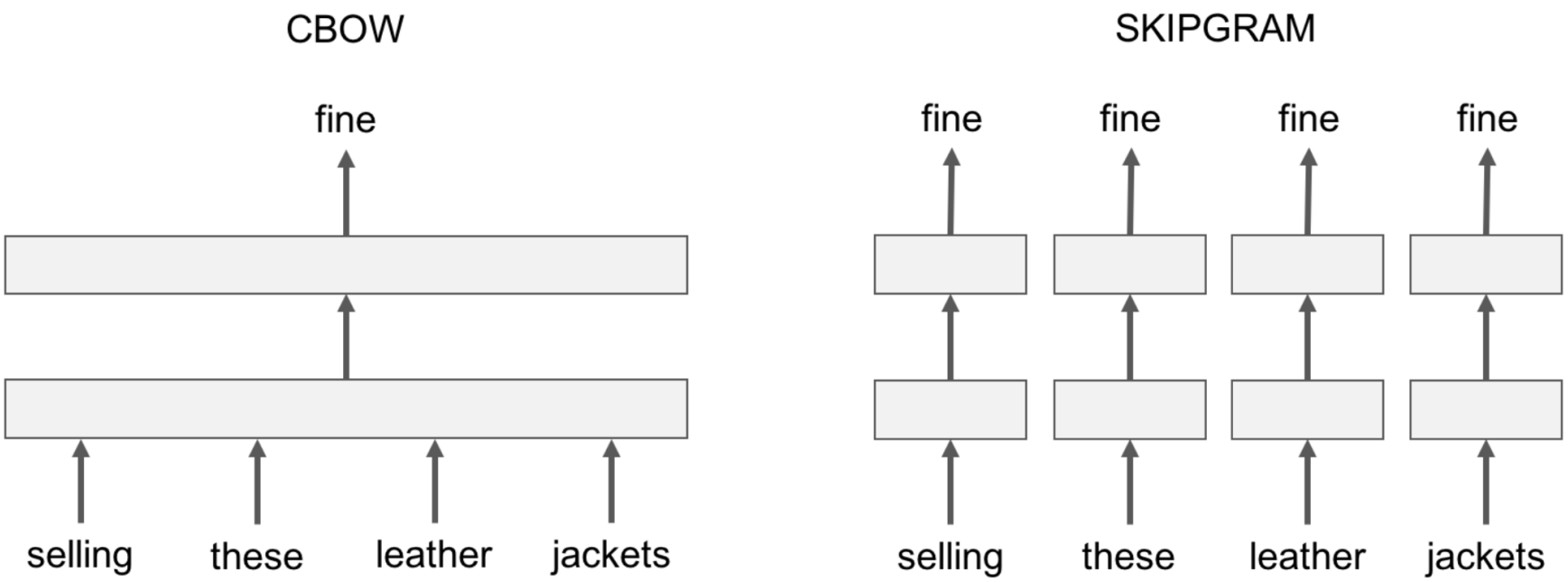
Erhan, Courville, Manzagol, Vincent, Bengio (JMLR, 2010)

word2vec: CBOW vs. skip-gram



Read more: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality

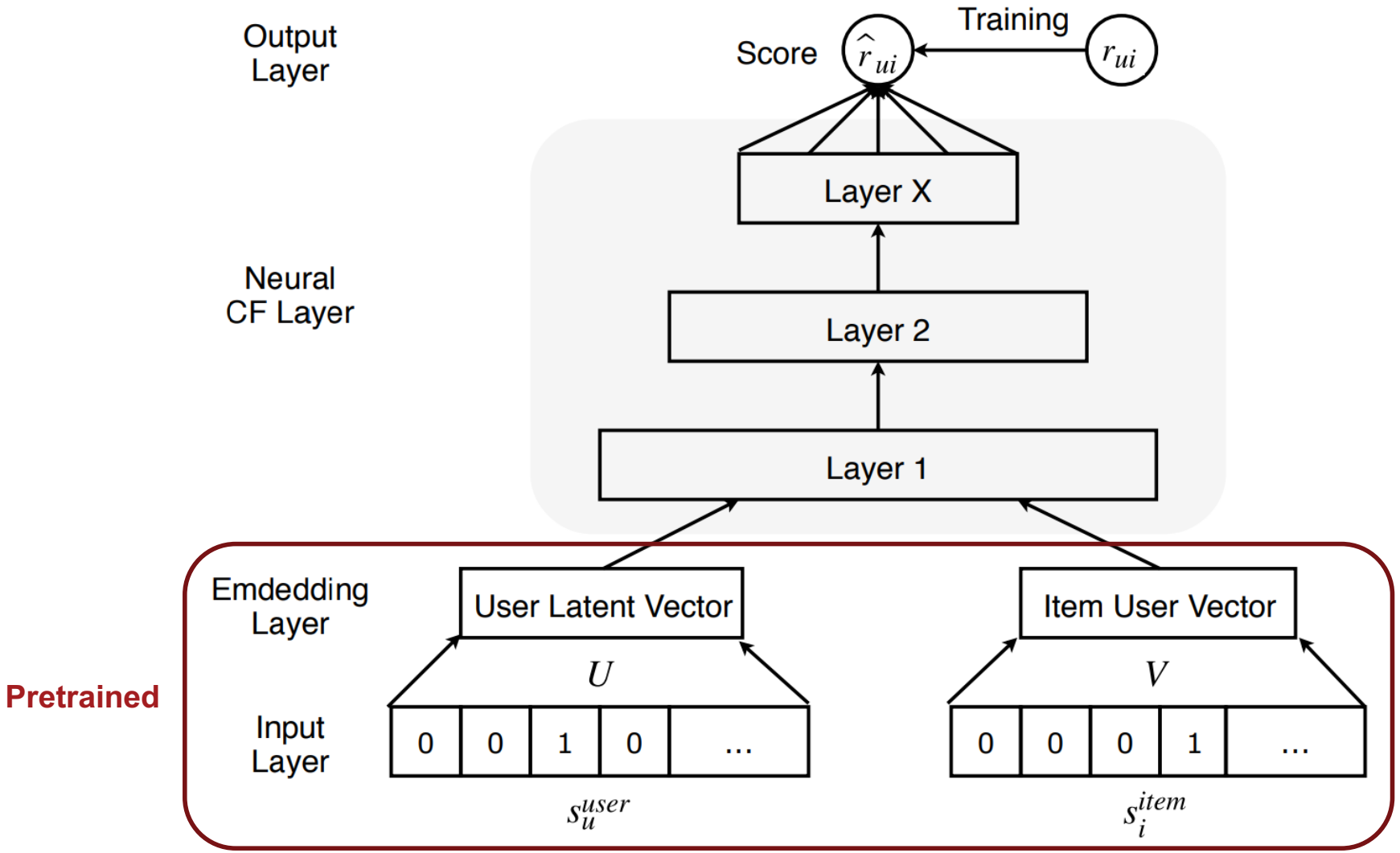
word2vec: CBOW vs. skip-gram



I am selling these fine leather jackets

Image: https://fasttext.cc/docs/en/unsupervised-tutorial.html

Revisiting Neural Matrix Factorisation



CNNs for Text

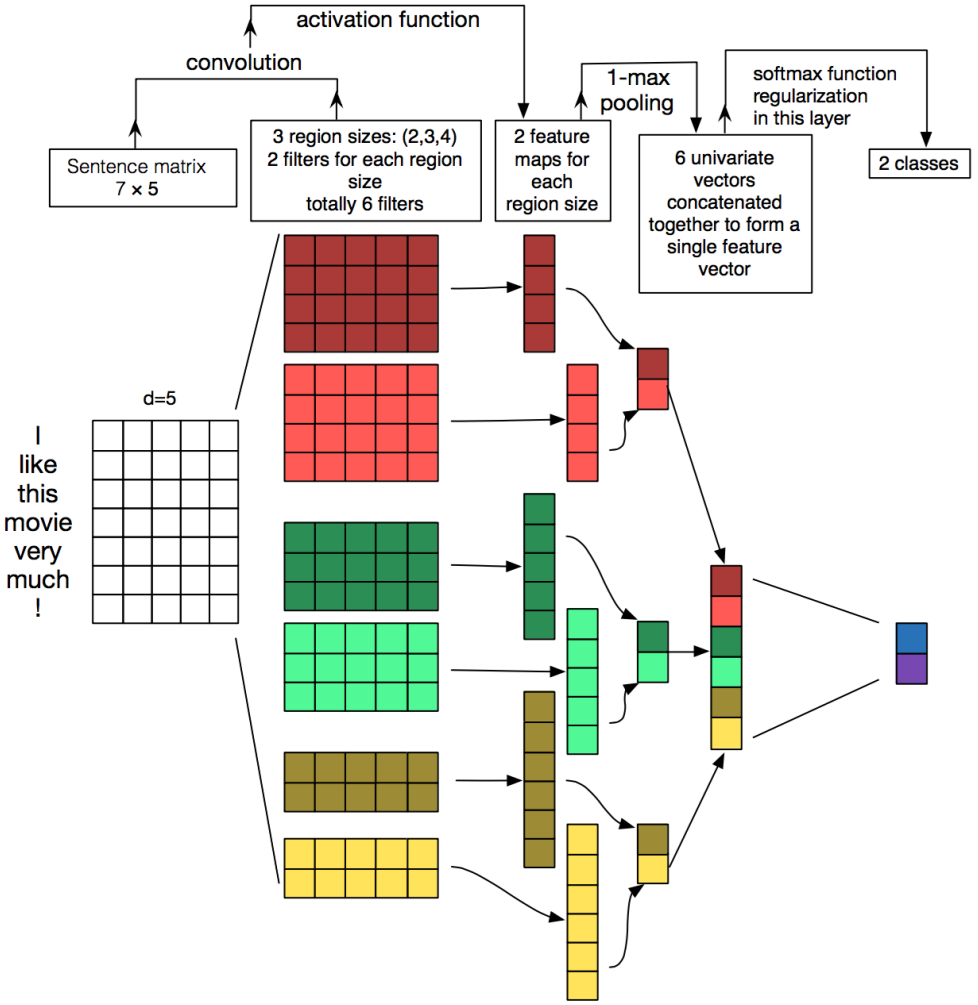


Image: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Context-Dependent Recommendation

Tensor Factorisation

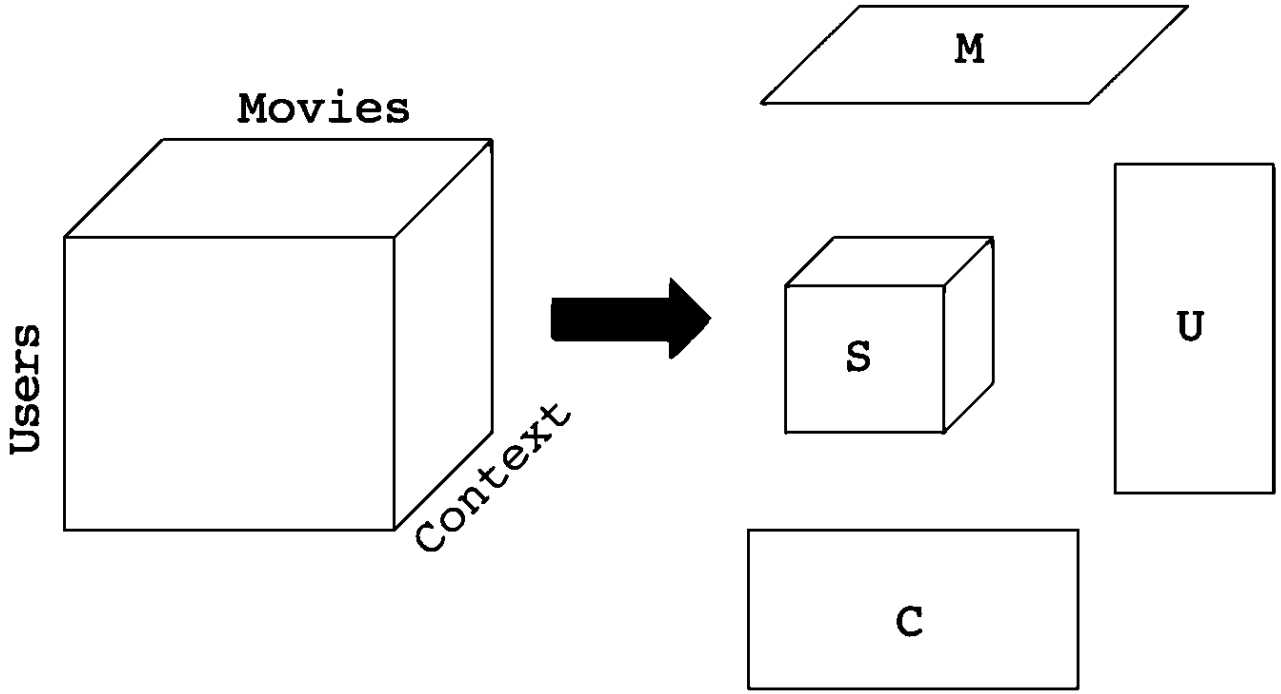
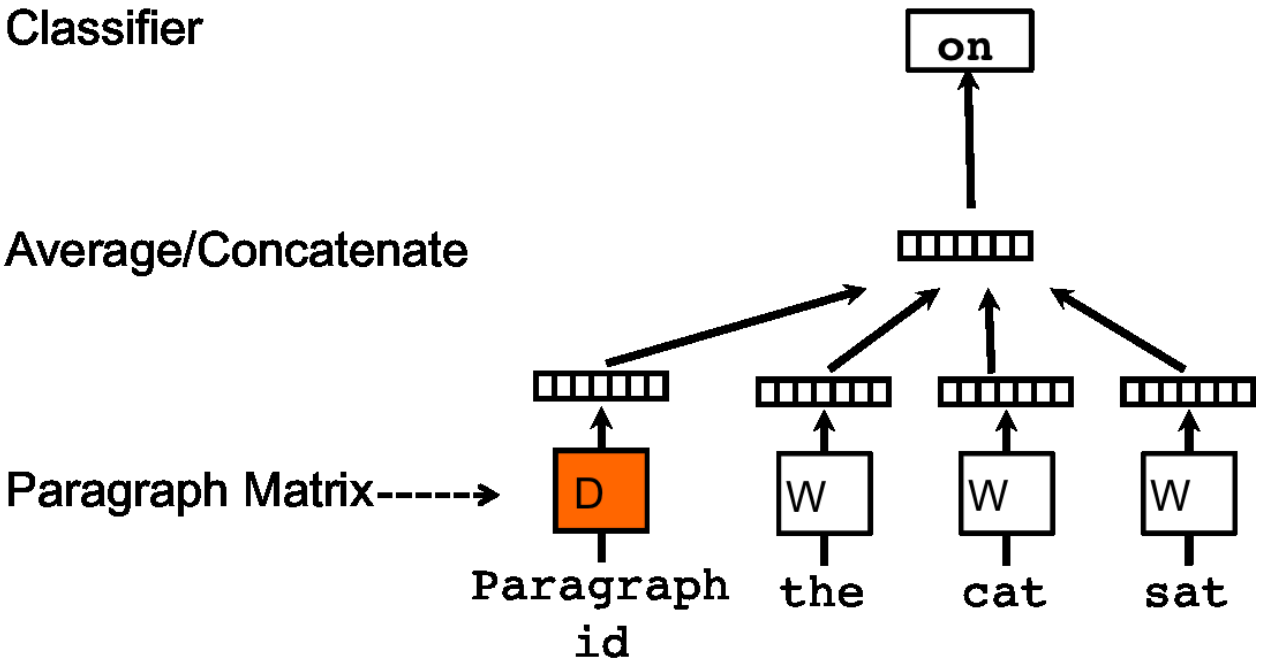


Image: https://xamat.github.io/pubs/karatzoglu-recsys-2010.pdf

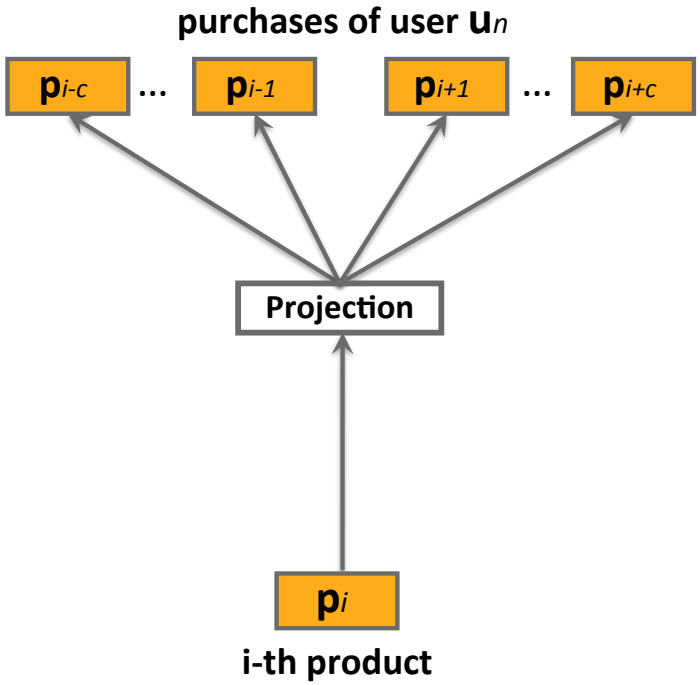
More Representation Learning for Incorporating Context

- doc2vec, Le & Mikolov, 2014 (https://arxiv.org/abs/1405.4053)



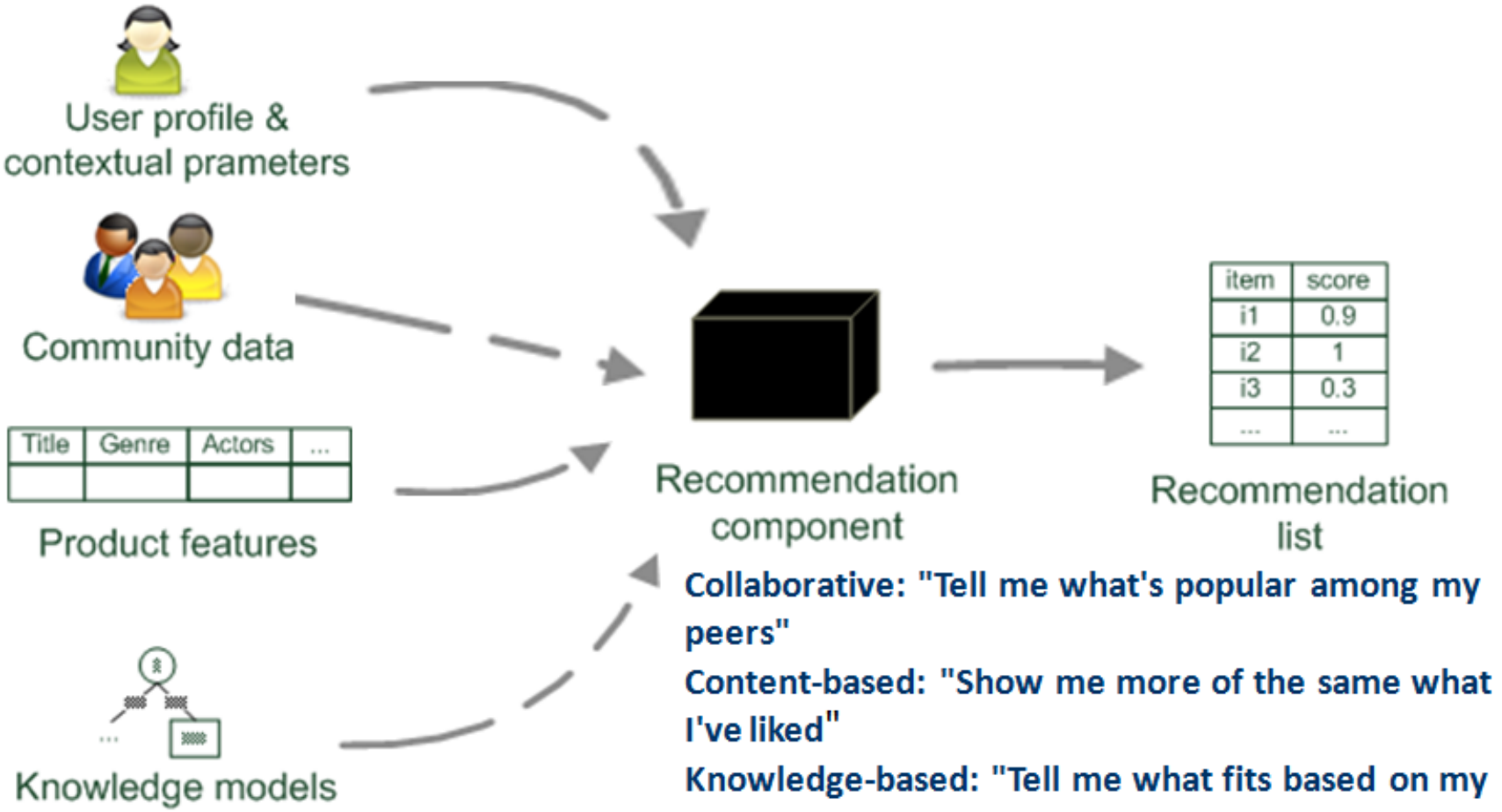
More Representation Learning for Incorporating Context

- prod2vec, Grbovic et al., 2015 (https://arxiv.org/abs/1606.07154)



- meta-prod2vec, Vasile et al., 2016 (https://arxiv.org/abs/1607.07326)
- Recurrent Neural Networks (RNNs)
  - Next lecture!

Hybrid Approaches



Hybridisation Methods

Weighted: The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.

Switching: The system switches between recommendation techniques depending on the current situation.

Mixed: Recommendations from several different recommenders are presented at the same time.

Feature combination: Features from different recommendation data sources are thrown together into a single recommendation algorithm

Cascade: One recommender refines the recommendations given by another.

Feature augmentation: Output from one technique is used as an input feature to another

Meta-level: The model learned by one recommender is used as input to another

Summary – This Lecture

- Machine Learning Based Recommender Systems
- Latent User and Item Representations
  - Matrix Factorisation
  - Logistic vs. Ranking Loss
- Brief Introduction to Representation Learning
  - Auto-encoders
  - Word representations
  - CNNs
- Context-Dependent Recommendation
  - Tensor Factorisation
  - Representation learning: doc2vec, prod2vec, …
  - Hybrid methods

Next Two Lectures

- Opinion Mining and Sentiment Analysis
- Various Methods
  - Rule-Based
  - Simple ML
  - Neural / Deep Learning
- Representation Learning Part 2

Thank you!

augenstein@di.ku.dk
@IAugenstein
http://isabelleaugenstein.github.io/