Modern Recommender Systems:
from Computing Matrices to Thinking with Neurons

Georgia Koutrika (ATHENA RC)

1

1

About me

Georgia Koutrika
- Research Director at ATHENA Research Center

Research interests
in the broader area of big data and in the intersection of databases, information retrieval, machine learning, and data mining, including:

- personalization and recommendation systems,
- user profiling and user analytics,
- large-scale information extraction, entity resolution and information integration,
- query and data exploration paradigms, including keyword search and natural language interfaces.

2

Tutorial Outline

- **Introduction**
- **Classical Recommendation Methods**
- **Matrix Factorization**
- **Multi-Armed Bandits**
- **Deep Learning Systems**

3

Why using Recommender Systems?

Life without recommender systems

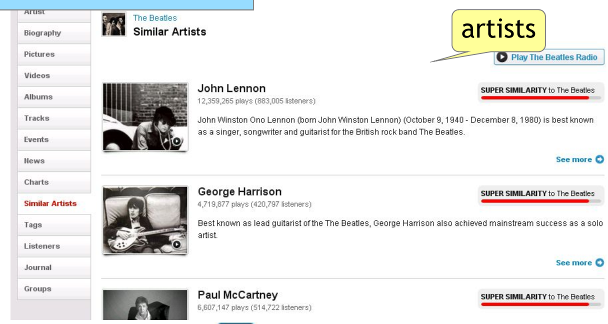
4

Why using Recommender Systems?

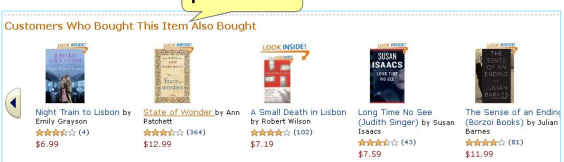
Recommender systems help users find and evaluate **items of interest**

movies

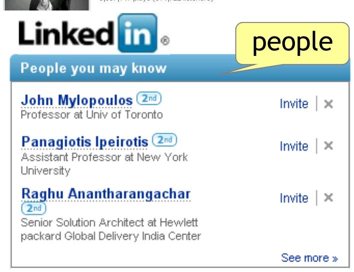
artists



products



people



5

Recommender Systems: how do they work?

In order to find items of interest, recommender systems leverage the **content of items** as well as **user actions and opinions**

1

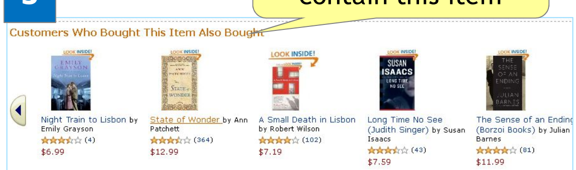
Viewing history of this viewer

2

Purchases based on frequency

3

Purchase histories that contain this item



4

Highlighting histories that contain this item

6

Why using Recommender Systems?

- Value for the customer
  - Find things that are interesting
  - Narrow down the set of choices
  - Explore the space of options
  - Discover new things

- Value for the provider
  - Additional and personalized service for the customer
  - Increase trust and customer loyalty
  - Increase sales, click trough rates, conversion etc.
  - Opportunities for promotion, persuasion
  - Obtain more knowledge about customers

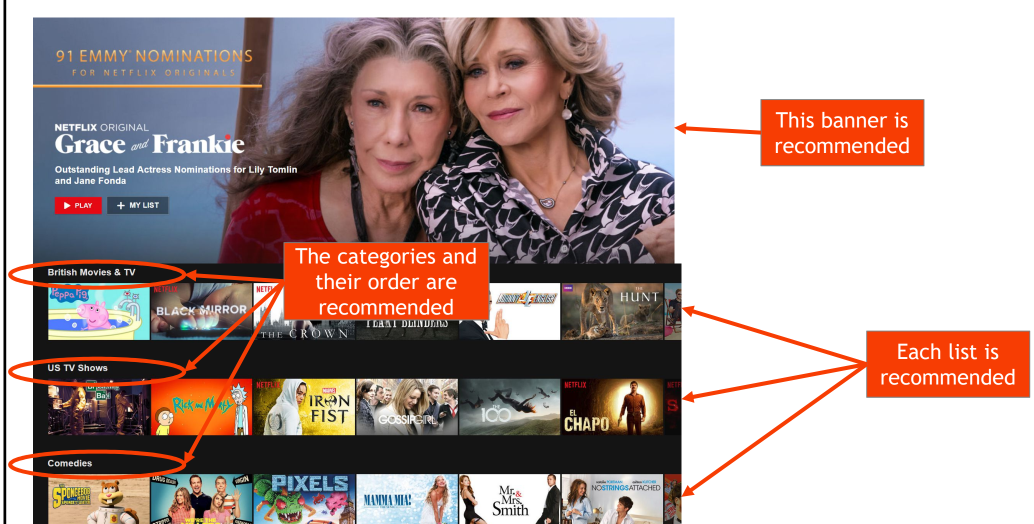
Netflix:
2/3 of the movies watched are recommended

Google News:
recommendations generate 38% more clickthrough

Amazon:
35% sales from recommendations

https://www.slideshare.net/xamat/recommender-systems-machine-learning-summer-school-2014-cmu

They are everywhere



The Recommendation Problem

U : the set of all users
I: be set of all possible recommendable items

Let **p** be a utility function measuring the usefulness of item **i** to user **u**, i.e.,
$p : U \times I \rightarrow R$, where *R* is a totally ordered set.

Objective
Learn *p* based on the past data

- Past behavior
- Relations to other users
- Item similarity
- Context
- …

Use *p* to predict the utility value of each item ***i in I*** to each user ***u in U***

User-Item Ratings Matrix

	$user_1$	$user_2$	$user_3$	…	$user_N$
$item_1$	**0.7**	**-**	**-**	**…**	**-**
$item_2$	**0.9**	**-**	**?**	**…**	**0.9**
…	**…**	**…**	**…**	**…**	
$item_M$	**-**	**-**	**0.9**	**…**	**0.8**

User Observable Behaviors

Minimum scope (columns) / Behavior Category (rows)

Behavior Category	Segment	Object	Class
Examine	View Listen	Select	
Retain	Print	Bookmark Save Purchase Delete	Subscribe
Reference	Copy / paste Quote	Forward Reply Link Cite	
Annotate	Mark up	Rate Publish	Organize

With no rating information, the system will not predict ratings but the likelihood that a user will like an item

Oard, D. W., Kim, J. (2001). Modeling information content using observable behavior. Proc. of the Annual Meeting of the American Society for Information Science and Technology (ASIST '01), 38-45.

Recommendations: Beyond Users and Products

The recommendation problem emerges in many database problems

Data exploration, Query optimization, Visualization, Data integration, Workflow design, etc

where the purpose may be to recommend :

tuples [1], queries [2], views [3],
exploration actions [4], query plans [5]
visualization graphs [6], work flows [7]

[1] M. Drosou, E. Pitoura. Ymaldb: Exploring relational databases via result-driven recommendations. The VLDB Journal, Dec2013
[2] M. Eirinaki et al. Querie: Collaborative database exploration. IEEE Trans. Knowl. Data Eng., 2014
[3] H. Ehsan,et al. Muve: Efficient multi-objective view recommendation for visual data exploration. ICDE 2016.
[4] T. Milo, A. Somech. React: Context-sensitive recommendations for data analysis. SIGMOD'16
[5] J. Zahir et al. A recommendation system for execution plans using machine learning. Math. and Comp. Applications, 21(2),2016
[6] M. Vartak et al. SEEDB: efficient data-driven visualization recommendations to support visual analytics. PVLDB,8(13), 2015
[7] D. Jannach et al. Supporting the design of machine learning workflows with a recommendation system. TiiS,6(1), 2016

Recommender Systems

A truly multi-disciplinary field

Information Retrieval

Machine Learning

Crowdsourcing

Data Mining

Information Filtering

Social Network Analysis

HCI

Probability theory and statistics

Statistical Analysis

Experimentation

User Studies

System Design

Classical Approaches

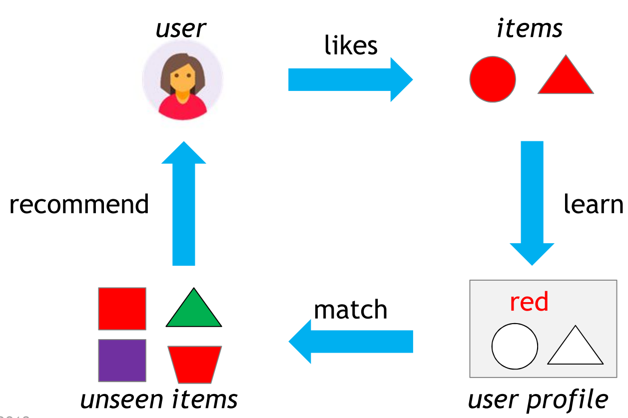
13

13

Content-based Filtering

Analyzes user past selections (e.g., web pages, movies) to learn user preferences

Recommends items with similar content (e.g., metadata, description, topics) to the user's past selections and likes



14

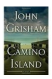
14

Step 1: Content Representation

all books

5	-	-	4	-	-

user selections (ratings)



For each item, create an **item profile.**

The profile is a vector (set) of features

1. **Metadata**: e.g., author, type, genre, year …
   e.g., John Grisham, hardcover, mystery

2. **Text**: "important" words in document
   e.g., gang, heist, books, Florida, …

   How to pick important words?

   Item profile = vector of words with their **TF*IDF** scores
   (Term frequency * Inverse Doc Frequency)

Standard IR-based approach

TF-IDF weighting scheme

Term frequency

$$TF_{ij} = \frac{f_{ij}}{\sum_{k \text{ in document } j} f_{kj}}$$

f_{ij} = frequency of term (feature) i in doc (item) j
we normalize TF to discount for "longer" documents

The higher the tf, the higher the importance of term i in the doc.

Inverse Doc Frequency

$$IDF_i = \log N/n_i$$

n_i = number of docs that mention term i
N = total number of docs

The more the term is distributed evenly, the less specific it is to a document

Step 1: Content Representation

all books

user selections

book representations

J. Grisham	mystery	Legal thriller	Paul Aertker	Crime	D. Baldacci	Florida	E.B White	friends
1	1	0	0	1	0	1	0	0
0	0	0	0	0	0	0	1	1
0	0	0	0	1	1	0	0	0
1	0	1	0	0	0	1	0	0
1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0

Step 2: User Profile Learning

all books

user selections

For each user, create a **user profile:**

1. **Detailed profile:** the profiles of the items the user has rated

J. Grisham	mystery	Legal thriller	Paul Aertker	Crime	D. Baldacci	Florida	E.B White	friends
1	1	0	0	1	0	1	0	0
1	0	1	0	0	0	1	0	0

2. **Aggregate Profile:** the weighted average of rated item profiles

J. Grisham	mystery	Legal thriller	Paul Aertker	Crime	D. Baldacci	Florida	E.B White	friends
4.5	2.5	2	0	2.5	0	4.5	0	0

Step 3: Recommendation

Prediction heuristic

1. Given user profile x and item profile i, estimate
   $score(\boldsymbol{x},\boldsymbol{i}) = \cos(\boldsymbol{x},\boldsymbol{i}) = (\boldsymbol{x}\cdot\boldsymbol{i})/(||\boldsymbol{x}||\cdot||\boldsymbol{i}||)$
   where $||x|| = \sqrt{\sum_{i=1}^{n} x_i^2}$
2. Select top-n items to show as recommendations

Standard IR-inspired approach:
User profile → Query
Item → Document

Step 3: Recommendation

all books

5	-	-	4	-	-

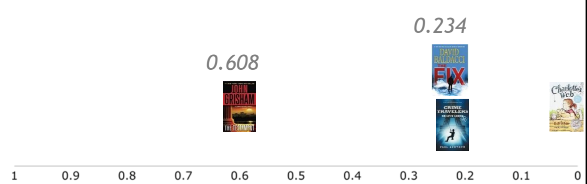
user selections

User

J. Grisham	mystery	Legal thriller	Paul Aertker	Crime	D. Baldacci	Florida	E.B White	friends
4.5	2.5	2	0	2.5	0	4.5	0	0

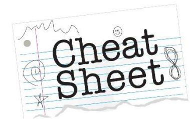
Books

1	1	0	0	1	0	1	0	0
0	0	0	0	0	0	0	1	1
0	0	0	0	1	1	0	0	0
1	0	1	0	0	0	1	0	0
1	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0



Cosine similarity between user profile and books

Content-based filtering



Content-based filtering recommends items with similar content (e.g., metadata, description, topics) to the items the user liked in the past

Input: depends only on the content/descriptions of the items and the users (not usage data)

Types:
- Information Retrieval (e.g., tf-idf, Okapi BM25)
- Machine Learning (e.g., Naïve Bayes, Support Vector Machines, decision trees, etc)

Pros	Cons
• No item cold-start • No popularity bias, can recommend items with rare features • Can use content features to provide explanations	• Item content needs to be machine readable and meaningful • Finding enough good features • Easy to pigeonhole the user • Serendipity is difficult

21

21

Collaborative Filtering

Analyzes usage data (e.g., ratings, purchases, downloads)

Recommends items with similar usage characteristics or items from users with similar usage characteristics to the user

A pure collaborative filtering system is one that performs no analysis of the items at all, instead the only thing it knows about an item is a unique identifier.

User-Item Ratings Matrix

	$user_1$	$user_2$	$user_3$	…	$user_N$
$item_1$	**0.7**	**?**	**?**	…	**?**
$item_2$	**0.9**	**?**	**-**	…	**0.9**
…	…	…	…	…	…
$item_M$	**?**	**?**	**0.9**	…	**0.8**

22

22

History of the term

1992: *Using collaborative filtering to weave an information tapestry*, D. Goldberg et al., Communications of the ACM

Experimental mail system at Xerox Parc that records reactions of users when reading a mail

- Basic idea: "Eager readers read all docs immediately, casual readers wait for the eager readers to annotate"
- Users are provided with personalized mailing list filters instead of being forced to subscribe
- E.g. Mails to [all] which were replied by [John Doe] and which received positive ratings from [X] and [Y].

1994: *GroupLens: an open architecture for collaborative filtering of netnews*, P. Resnick et al., ACM CSCW

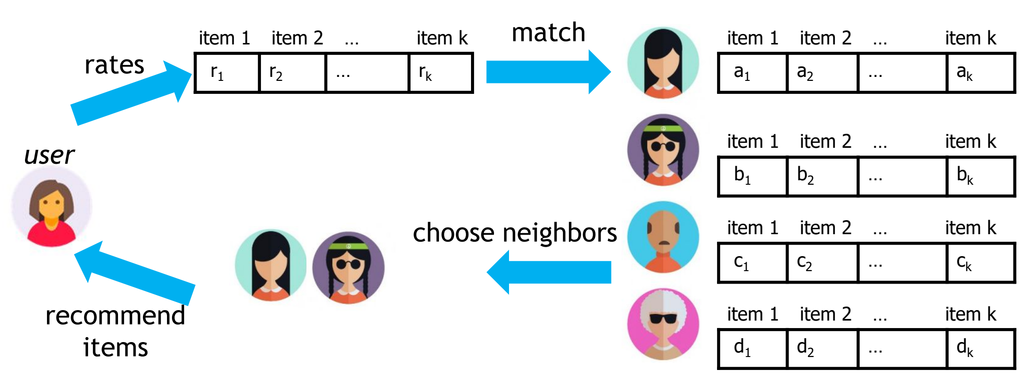
- Basic idea: "People who agreed in their subjective evaluations in the past are likely to agree again in the future"
- Builds on newsgroup browsers with rating functionality

Typical CF : Neighborhood methods

User-Based CF finds similar users, i.e., who have similar appreciation for items as you and recommends new items based on what they like.

Assumption

There are similar people to me!



Step 1: Matching the User to Other Users

- Pearson correlation (GroupLens)

$$w_{a,u} = \frac{\boldsymbol{covar}(r_a, r_u)}{\sigma_{r_a}\sigma_{r_u}} = \frac{\sum_i (r_{a,i} - \overline{r_a})(r_{u,i} - \overline{r_u})}{\sqrt{\sum_i (r_{a,i} - \overline{r_a})^2}\sqrt{\sum_i (r_{u,i} - \overline{r_u})^2}}$$

r_a and r_u are the rating vectors for the items rated by **both** a and u

$\overline{r_a}, \overline{r_u}$ are the users' average ratings

GroupLens:
Built on the intuition that every time a user read a Usenet News article she formed a valuable opinion, captured those opinions as “ratings” and used the ratings of like-minded readers to produce personal predictions that were displayed as part of the article header.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. Proc. of the ACM Conf. on Computer Supported Cooperative Work, 75-186

Step 2: Choosing User Neighborhood

- thresholding (Ringo)
- K-nearest neighbors (GroupLens)

Step 3: Recommendation

Prediction heuristic:

1. Given user profile a and item i, estimate the user's rating for i as the weighted average of the ratings of the similar users u for i

similarity of users a and u

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u=1}^{n} w_{a,u}(r_{u,i} - \bar{r}_u)}{\sum_{u=1}^{n} w_{a,u}}$$

rating of user u to item i

average rating of user a

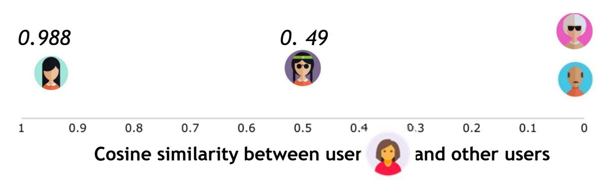
2. Select top-n items to show as recommendations

Recommendation

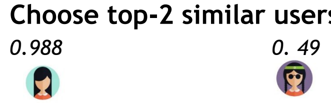
	item1	item 2	item 3	item 4	item 5
Ann	0.5	0.9	?	-	-
Kate	0.5	0.9	0.8	-	-
Joan	0.8	0.1	0.5	-	0.2
Paul	-	-	0.9	0.7	0.3
Betty	-	-	-	0.8	0.8

1. User similarity as vector similarity

$W_{a,u} = \cos(\mathbf{r}_a, \mathbf{r}_u) = (r_a.r_u)/(||r_a||\cdot||r_u||)$



2. Choose top-2 similar users

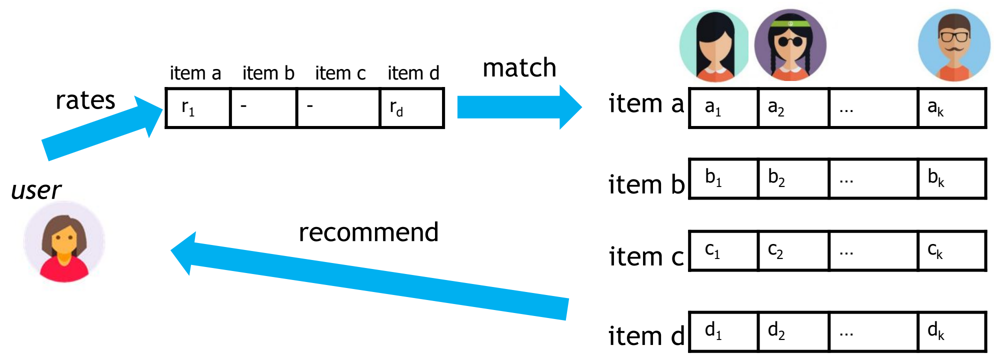


3. Predict rating of Ann for item 3

$p_{Ann,3} = 0.7 + (0.988*(0.8 - 0.73) + 0.49*(0.5 - 0.46))/(0.988 + 0.49) = 0.75$

Item-based CF

Item-based CF recommends items that are similar to the ones the user likes, where similarity is based on item ratings



Karypis, G. (2000). Evaluation of Item-Based Top-N Recommendation Algorithms. Technical Report CS-TR-00-46, Computer Science Dept., University of Minnesota.

Neighborhood methods : Recap

Pros:

- Simplicity
- Efficiency
- Ability to produce accurate and personalized recommendations.

Cons:

- Scalability limitations as they require a similarity computation (between users or items) that grows with both the number of users and the number of items. In the worst case this computation can be O(m*n), but in practice the situation is slightly better with O(m+n), partly due to exploiting the sparsity of the data.

- Sparsity presents a challenge because we only have user ratings for small percentage of the large number of items.

Model-based Collaborative Filtering

- Use the ratings to learn a predictive model based on which predictions on new items are made.
- Models are updated / re-trained periodically

+ Model-based CF approaches can help overcome some of the limitations of neighborhood-based methods.

– Model-building and updating can be computationally expensive

- Large variety of algorithms used:
  Bayesian networks, clustering, classification, regression, matrix factorization, restricted Boltzmann machines, etc.

31

Model-based Item-Based CF

In a typical e-Commerce scenario:

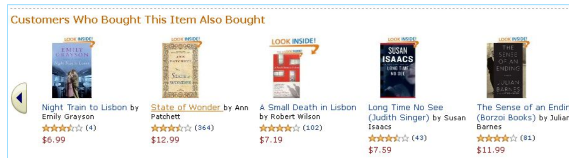
- Set of items that is static compared to the number of users that changes most often.
  → The static nature of items leads us to the idea of precomputing the item similarities.
- One possible way of precomputing the item similarities is to compute all-to-all similarity and then performing a quick table look-up to retrieve the required similarity values.
- This method, although saves time, requires an $O(n^2)$ space for n items.

32

Model-based Item-Based CF (Amazon)

To determine the most-similar match for a given item, the algorithm builds a similar-items table by finding items that customers tend to purchase together.

```
For each item in product catalog, I1
  For each customer C who purchased I1
      For each item I2 purchased by customer C
          Record that a customer purchased I1 and I2
  For each item I2
      Compute the similarity between I1 and I2
```



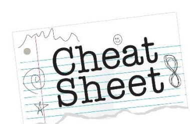
This offline computation of the similar-items table is extremely time intensive, with $O(N^2M)$ as worst case. In practice, however, it's closer to $O(NM)$, as most customers have very few purchases.

Sampling customers who purchase best-selling titles reduces runtime even further, with little reduction in quality.

Pre-processing approach by Amazon.com (in 2003)

Greg Linden, Brent Smith, and Jeremy York. *Amazon.com Recommendations. Item-to-Item Collaborative Filtering. Industry Report*

Collaborative filtering



Collaborative filtering looks for patterns in the overall user activity to produce user-specific recommendations	
Input: depends only on usage data (e.g., ratings, purchases, downloads)	
Types: • Neighborhood-based CF (user-based, item-based) • Model-based CF (clustering, matrix factorization, restricted Boltzmann machines, Bayesian networks, etc)	
Pros	**Cons**
• Minimal domain knowledge required • User and item features not required • Good enough results in many cases	• Cold start • Requires high user:item ratio (1:10) • Popularity bias (doesn't play well with long tail) • Explanations are difficult

Recommender Systems: A map

Traditional methods	Modern methods
• Popularity-based	• Multi-armed bandits (explore/exploit)
• Content-based filtering	• Matrix Factorization
• Collaborative filtering	• Graphical models
• Clustering	• Deep learning
• Association Rules	• Social recommendations
• Demographic	• Learning to rank

Tutorial Outline

- **Introduction**
- **Classical Recommendation Methods**
  – Offline, Scalability Issues, Cannot learn from diverse user signals
- **Matrix Factorization**
  + Learning more accurate and scalable recommendations
- **Multi-Armed Bandits**
  + Making online/interactive recommendations
- **Deep Learning Systems**
  + Learning from huge amounts of multiple user signals

Matrix Factorization

37

37

Netflix Prize

- Netflix Prize:
    - *>10% improvement, win $1,000,000*
- Top performing model(s) ended up be a variation of Matrix Factorization (*SVD++, Koren, et al*)
- MF is still the foundational method on which most collaborative filtering systems are based

38

38

Netflix Prize

- Large-scale, industrial strength data set
  - Released training data: 100M movie ratings by 500K users on 17K movies
  - Test data: 3M ratings
  - Held-out truth set
- 50,051 contestants
- Rapid development

Winner Team

Matrix Factorization (without the math)

Matrix factorization *assumes* that:

- Each user can be described by *k attributes* or *features*.
  For example, feature 1 might be a number that says how much each user likes sci-fi movies.

- Each item (movie) can be described by an analogous set of *k* attributes or features.
  To correspond to the above example, feature 1 for the movie might be a number that says how close the movie is to pure sci-fi.

- If we multiply each feature of the user by the corresponding feature of the movie and add everything together, this will be a good approximation for the rating the user would give that movie.

41

41

Evolution of MF in RS

Dimension Reduction Techniques: PCA & SVD

SVD-Based Matrix Factorization (SVD)

Early stage

Modern stage

Basic Matrix Factorization

Extended Matrix Factorization

https://www.slideshare.net/irecsys/matrix-factorization-in-recommender-systems

42

42

Why Dimension Reduction?

The curse of dimensionality
Problems that arise when analyzing and organizing data in high-dimensional spaces, such as:

- high sparsity, unnecessarily large storage space and processing time

Applications

- **Information Retrieval:** Web documents, where the dimensionality is the vocabulary of words
- **Recommender Systems:** Large scale of rating matrix
- **Social Networks:** Facebook graph, where the dimensionality is the number of users
- **Biology:** Gene Expression
- **Image Processing:** Facial recognition

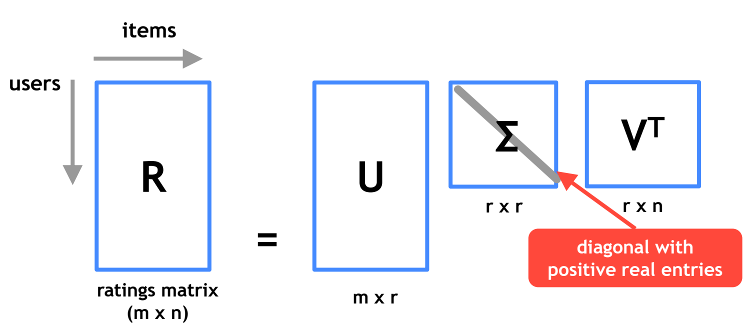
There are many techniques for dimension reduction

- Linear Discriminant Analysis (LDA)
- Principal Component Analysis (PCA)
- Singular Value Decomposition (SVD)

Singular Value Decomposition (Origins)

Early systems use Singular Value Decomposition (SVD) for collaborative filtering

Singular Value Decomposition (SVD) is a factorization of a matrix into the product of three matrices

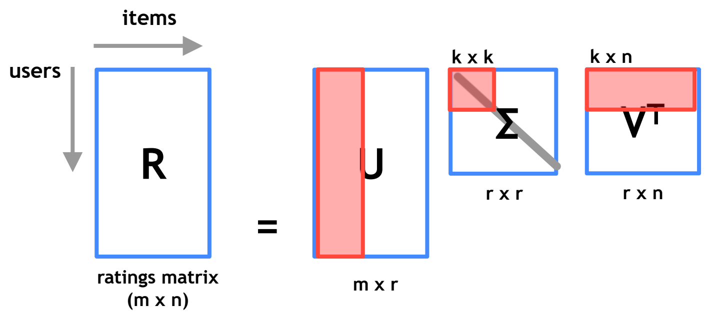


$$R = U \Sigma V^T$$

SVD: Low-rank approximation

So, how to realize the dimension reduction in SVD?

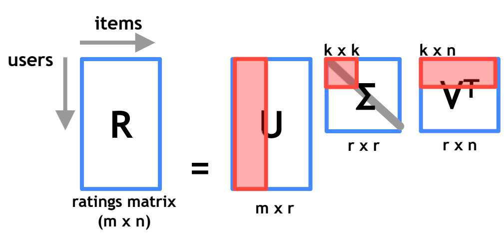
Simply, SVD tries to find a matrix to approximate the original matrix R :
take the k largest singular values along with their associated left and right singular vectors;
other dimensions will be discarded.



A **low-rank approximation of R** of rank k using SVD

45

SVD: Low-rank approximation



- Low-rank approximation is a **minimization problem**
- The **cost function** measures the fit between a given matrix and an approximating matrix (the optimization variable), subject to a constraint that the approximating matrix has reduced rank.

Matrix approximation lemma or Eckart-Young-Mirsky theorem

The result is an approximating matrix that minimizes the Frobenius norm over all rank-k matrices

$$R' = [U', \Sigma', V^{T}{}'] = \operatorname{argmin} \|R - U_k \Sigma_k V_k^T\|_F^2$$

Frobenius Norm

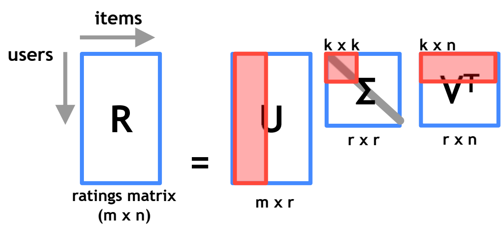
46

Frobenius norm

The Frobenius norm is matrix norm of an m×n matrix A defined as the square root of the sum of the absolute squares of its elements

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

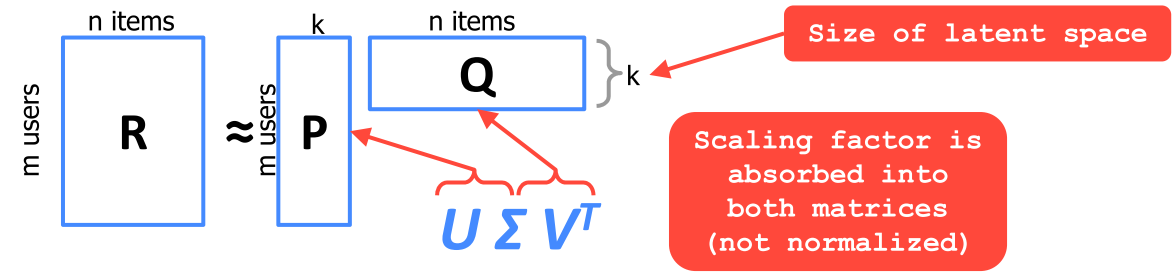
Recommendation using SVD



The predicted recommendation score for a user **u** to an item **i** is computed as:

$$\widehat{r_{ui}} = \overline{r_u} + U_k \cdot \sqrt{\Sigma_k}(u) \cdot \sqrt{\Sigma_k} V_k^T(i)$$

Low-rank Matrix Factorization



- No orthogonality requirement
- To learn the values in P and Q, we minimize a cost function
  - Weighted least squares (or others)

$$\min \sum_{ij} (R_{ij} - p_i q_j)^2$$

This optimization may overfit the equation if p_i/q_i are too many

Georgia Koutrika © 2018

49

49

Low-rank Matrix Factorization

Avoiding overfitting

- Regularization, e.g. L2, L1, etc

$$\min \sum_{ij} ((R_{ij} - p_i q_j)^2 + \underline{\lambda(||p|| + ||q||)})$$

- The system learns the model by fitting the previously observed ratings.
- However, the goal is to generalize those previous ratings so to predict future, unknown ratings.
- Thus, the system should avoid overfitting the observed data by regularizing the learned parameters, whose magnitudes are penalized.
- The constant λ controls the extent of regularization and is usually determined by cross-validation.

50

Georgia Koutrika © 2018

50

Training

Minimize $$\min \sum_{ij} ((R_{ij} - p_i q_j)^2 + \lambda(\|p\| + \|q\|))$$

- Alternating Least Squares
  - Assume p's are fixed, solve q's (becomes simple least squares), then alternate with q's fixed, and solve for p's, etc, etc
  - Could be viewed as coordinate descent on P and Q

- Or: SGD (Stochastic Gradient Descent)

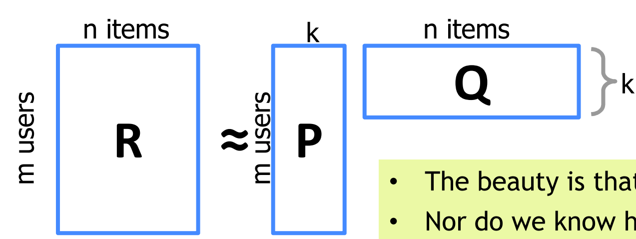
$$q_{ik} \leftarrow q_{ik} - \eta \frac{\partial L}{\partial q_{ik}}, \quad p_{ik} \leftarrow p_{ik} - \eta \frac{\partial L}{\partial p_{ik}}$$

$$\frac{\partial L}{\partial p_{ik}} = -2w_{ik}(R_{iu} - p_i q_u).q_{uk} + 2w_{iu}\alpha p_{ik}$$

51

51

Low-rank Matrix Factorization



- The beauty is that we do not know what these features are.
- Nor do we know how many *(k)* features are relevant.
- We simply pick a number for *k* and learn the relevant values for all the features for all the users and items

- No orthogonality requirement
- To learn the values in P and
  - Weighted least squares (for others)

- Of course, the product of the component matrices is never going to be exactly equal to the original matrix

- So we instead accept a good enough approximation at the cost of computing the ratings fast via a dot product.

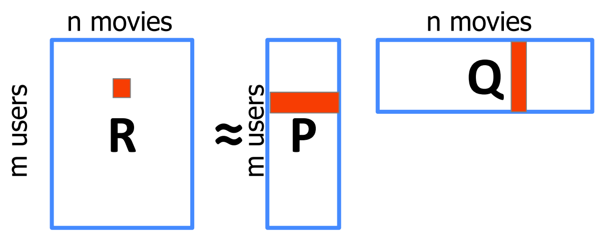
$$\min \sum_{ij} (R_{ij} - p_i q_j)$$

52

52

Recommendation with MF

User	Harry Potter	Batman	Spiderman
U1	5	3	4
U2	?	2	4
U3	4	2	?



Movie rating

A rating r_{ui} can be estimated by the dot product of user vector p_u and item vector q_i.

$$r_{ui} \approx q^T_i \, p_u$$

Recommendation with MF

User	Harry Potter	Batman	Spiderman
U1	5	3	4
U2	?	2	4
U3	4	2	?

P

Set k=5, only 5 latent factors

	F1	F2	F3	F4	F5
U1	0.2763678	-0.37747	-1.26192	-1.54754	0.4738
U2	0.3999	-0.52747	-0.28946	-1.51597	0.73743
U3	0.2252336	-0.29125	-1.06624	-1.22463	0.37353

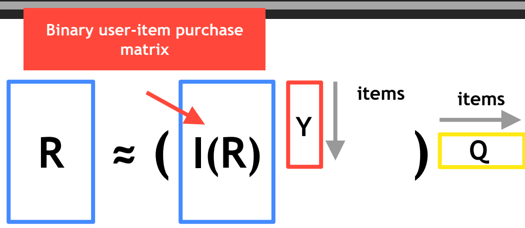
Q

	HarryPotter	Batman	Spiderman
F1	0.301758	0.14297	0.43409
F2	-0.405407	-0.20245	-0.57514
F3	-1.399694	-0.9232	-0.46807
F4	-1.677619	-0.94013	-1.72021
F5	0.5118556	0.23861	0.79576

Predicted Rating (U3, Spiderman) =
Dot product of the two Yellow vectors = 3.16822

https://www.slideshare.net/irecsys/matrix-factorization-in-recommender-systems

Asymmetric Matrix Factorization



- Replace user-vector with sum of item vectors for the items they rated

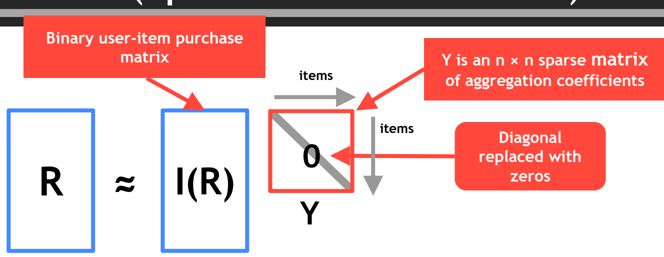
$$R_{ij} = \left(\frac{1}{\|N(u)\|} \sum_{k \in N(u)} \boxed{y_k} \right)^T \boxed{q_j}$$

N(u) is all items user u rated

If item space is << than your user space → nice trick to save computations

Factorization meets the neighborhood: a multifaceted collaborative filtering model. Y. Koren. KDD 2008

SLIM (sparse linear model)



- SLIM replaces low-rank approx by a sparse item-item matrix. Sparsity comes from L1 regularizer.
- Equivalent to constructing a regression using user's play history to predict ratings

$$R_{ij} = \sum_{k \in N(u), k \neq j} y_k$$

- Top-N recommendation for u_i is done by sorting u_i's non-rated items based on their recommendation scores in R in decreasing order and recommending the top-N items
- NB: Important that diagonal is excluded. Otherwise solution is trivial.

SLIM: Sparse Linear Methods for Top-N Recommender Systems. Xia Ning and George Karypis. ICDM 2011

Matrix Factorization

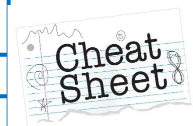
Matrix Factorization discovers latent features that determine how a user rates an item

Input: depends only on usage data (i.e., ratings)

Assumption: In trying to discover the different features, the assumption is that the number of features would be smaller than the number of users and the number of items.

Methods:
- Low-rank Matrix Factorization
- Asymmetric Matrix Factorization
- SLIM

Pros	**Cons**
• Dimension reduction • Superior performance both in terms of recommendation quality and scalability • Compact memory-efficient model	• Learnt latent space is not easy to interpret • Only uses information from the users- cannot generalize to completely unrated items



57

57

Multi-armed Bandits

58

58

Multi-armed Bandits

The **multi-armed bandit problem**:

A gambler at a row of slot machines has to decide which machines to play

When played, each machine provides a random reward.

Objective:
Maximize the sum of rewards earned through a sequence of lever pulls.

Search Tradeoff:
- **Explore** the different machines to learn their payoffs
- **Exploit** the machine with the greatest payoff.

Multi-armed Bandits

Machine 1	Machine 2	Machine 3	Machine 4
50%	70%	35%	45%

Reward probabilities are unknown.

Which machine to pick next?

Exploration vs. Exploitation Dilemma

Restaurant Selection

- Exploitation: Go to your favorite restaurant
- Exploration: Try a new restaurant

Oil Drilling

- Exploitation: Drill at the best known location
- Exploration: Drill at a new location

Game Playing

- Exploitation: Play the move you believe is best
- Exploration: Play an experimental move

Finance, Clinical trials, ….

MAB Problem

Set of "arms" A={a_1, … }.

The algorithm proceeds in discrete trials. In each trial $t \in \{1,2 \ldots\}$:

1. Consider the context x_t
2. Choose an arm a_t
3. Observe reward R_t
4. Improve reward knowledge for arm a_t with the new observation (x_t, a_t, R_t)

Contextual-bandit algorithm

Goal: minimize regret

$$Regret(T) = \underbrace{\sum_{t=1}^{T} E[R|a_t^*, X_t]}_{\text{Total optimal rewards}} - \underbrace{\sum_{t=1}^{T} R_t}_{\text{Total actual rewards received}}$$

MAB Problem

Set of "arms" A={a_1, … a_K}.

The algorithm proceeds in discrete trials. In each trial $t \in \{1,2 \ldots\}$:

For all t:

- The arm set A remains unchanged and contains K arms
- The context is the same

K-armed bandit or Context-free algorithm

MAB Problem

Example: Article/Ad Recommendation

Set of arms : a set of news articles, ads to recommend

Context x_t : user demographics, user location, …

Reward R_t : When a presented article is clicked, a payoff of 1 is incurred; otherwise, the payoff is 0.

Total Rewards : total CTR

Maximizing the expected number of clicks from users =
Maximizing the total expected payoff in our bandit formulation

MAB Components

Multi-armed Bandit algorithms have two components:

Arm Selection Strategy
How to choose an arm a_t

Reward Model
How to improve reward knowledge for arm a_t with the new observation (x_t, a_t, R_t)

Example:

$$\widehat{R_t}(a) \leftarrow \frac{1}{N(a)} \sum_{i=1}^{t} r_i \, 1[a_i = a]$$

Naïve approach: ε-greedy

Setting: Context-free K-armed bandit problem
Set of "arms" A={a_1, … a_K}.

In each trial $t \in \{1,2 \ldots T\}$:

1. Estimates the average payoff $\widehat{R_t}(a)$ of each arm a
2. With probability 1 - ε , it chooses the greedy arm (i.e., the arm with highest payoff estimate);
   With probability ε , it chooses a random arm

$\varepsilon = 0.05$ → chooses a random arm

explore

$\varepsilon = 0.95$ → chooses current best arm

exploit

ε-greedy

A simple bandit algorithm

Initialize, for each arm a = 1 to K:

$R(a) \leftarrow 0$ (the initial reward estimation)

$N(a) \leftarrow 0$ (the number of times a is selected)

Repeat forever:

$$a_t \leftarrow \begin{cases} argmax_a R(a) & with\ probability\ 1 - e\ \ (breaking\ ties\ randomly) \\ random\ action & with\ probability\ e \end{cases}$$

Arm selection Policy

Observe reward R for selected arm a_t

$N(a_t) \leftarrow N(a_t) + 1$

$$R(a_t) \leftarrow R(a_t) + \frac{1}{N(a_t)}(R - R(a_t))$$

Reward Model : Monte-Carlo evaluation

ε-greedy has linear total regret

ε-greedy

Pros:

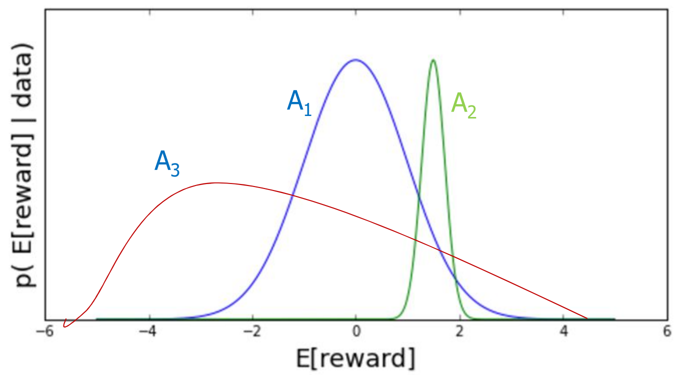
Easy to implement
Guaranteed level of exploration

Cons:

How to set epsilon? Decrease over time?
Unguided exploration

Optimism in the Face of Uncertainty

- The **more uncertain** we are about an action-value, the **more important** it is to explore that action
- It could turn out to be the best action
- Once we pick it, we are less uncertain about the value and more likely to pick another action
- Until we home in on the best action



69

Optimism in the Face of Uncertainty

Upper confidence bound algorithms

Estimate an upper confidence $U_t(a)$ for each action such that with high probability

$$\underbrace{R(a)}_{\text{Actual reward}} \leq \underbrace{\underbrace{\widehat{R_t}(a)}_{\text{Estimated mean}} + \underbrace{U_t(a)}_{\text{Estimated upper confidence}}}_{\textbf{Upper confidence bound}}$$

Select action maximizing Upper Confidence Bound (UCB) Arm selection Policy

$$a_t = argmax_{a\ in\ A}\ (\ \widehat{R_t}(a) + U_t(a)\ )$$

70

70

Upper confidence bound algorithms

UCB depends on the number of times $N(a)$ the item has been selected

- Small $N_t(a)$ → large $U_t(a)$ (estimated value is uncertain)
- Large $N_t(a)$ → small $U_t(a)$ (estimated value is accurate)

UCB1: An item's confidence interval is computed as

$$U_t(a) = \frac{\alpha}{\sqrt{N_t(a)}}$$ **(α** is a parameter)

UCB has logarithmic asymptotic total regret

Upper confidence bound algorithms

Pros:

Sampling focuses on most promising arms
Good theoretical regret bounds

Cons:

How to set alpha? Decrease over time?
Deterministic strategies suffer under delayed feedback

Bayesian Bandits

Let us assume that for an arm a, we are given 10 independent samples (say 1=click, 0=no click):
0; 0; 1; 1; 0; 1; 1; 1; 0; 0.

Frequentist Approach

Estimate that the true payoff for *a* is close to the average 0.5 with some confidence.

- e-greedy
- UCB

Bayesian learner

Maintains a probability distribution to represent his uncertainty about the parameter.

The probability distribution represents the chance that the parameter is of a certain value.

- Prior distribution $p(R|A_i)$
- Posterior distribution $p(R|A_i, H_t)$
  where H_t is the set of observations up to time t

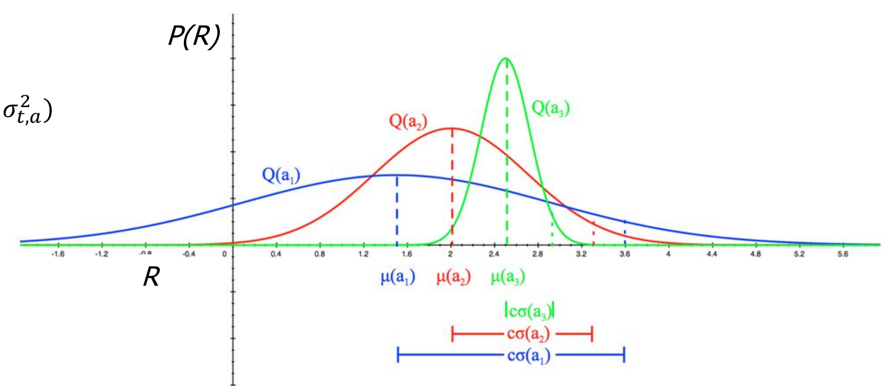
Use posterior to guide exploration
- Upper confidence bounds (Bayesian UCB)
- Probability matching (Thompson sampling)

73

Bayesian UCB

Assume no context and that arms are independent.
For each arm a, the reward distribution is **Gaussian**

prior:
$$p(R|a) \sim \mathcal{N}(\mu_0, \sigma_0^2)$$
posterior:
$$p(R|a, \mathcal{H}_t) \sim \mathcal{N}(\mu_{t,a}, \sigma_{t,a}^2)$$



74

Bayesian UCB

- Compute Gaussian posterior over $\mu_{t,a}$ and $\sigma^2_{t,a}$ (by Bayes law)

$$p(\mu_{t,a}, \sigma^2_{t,a} | H_t) \propto \prod_{t|a_t=a} \mathcal{N}(r_t;\ \mu_{t,a}, \sigma^2_{t,a}) p(\mu_{t,a}, \sigma^2_{t,a})$$

Reward Model

- Pick action that maximizes standard deviation

Arm selection Policy

$$a_t = argmax_{a\ in\ A}\ (\mu_{t,a} + c\ \frac{\sigma_{t,a}}{\sqrt{N_t(a)}})$$

75

75

MABs and Recommender Systems

(1) Popularity ranking

Balance exposure of new items (exploration) with old winners (exploitation).

(2) Model-based collaborative filtering

Score relevance using UCB or Thompson sampling.
Rank by these scores.

(3) Dueling bandits

Efficiently compare multiple rankers

76

76

Popularity Ranking

- A repository of content (e.g., for filtering news articles or for the display of advertisements)
- The content of the repository changes dynamically
- Content popularity changing over time
- A significant number of visitors are entirely new with no historical consumption record

These issues make traditional recommender-system approaches difficult to apply

Why use *MAB*?

- Naturally handles uncertainty due to item churn
- Learns directly from user feedback
- Randomness increases item coverage (# of items impressed)
- Faster experimentation

Challenge: how to optimally balance the two competing goals:

- maximizing user satisfaction in the long run,
- gathering information about goodness of match between user interests and content potentially reducing user satisfaction in the short term

Example: Personalized News

- The default "Featured" tab in the Today Module highlights one of four high-quality articles
- An hourly-refreshed article pool curated by human editors.
- Articles have short lifetimes (6-8 hours)
- The pool of articles is constantly changing
- The user population is dynamic
- Each article has different CTRs at different times of day or when shown in different slots

Rank available articles according to individual interests,
and highlight the most attractive article for each visitor at the story position

79

79

Example: Personalized News

How to apply a MAB approach

Split live traffic in two (or more) buckets:

- Learning bucket: use a MAB to learn the goodness of articles
- Serving bucket: articles that are currently the best are greedily shown in the serving bucket (e.g., blended with popular articles)

80

80

Example: Personalized News

How to apply a MAB approach

1. **No personalization.**
   - Use algorithms that make no use of features, such as ε-greedy or UCB1.
   - Apply one instance of the MAB algorithm running over the pool of curated content.
   - The content is refreshed every hour, so the duration of the trial is 1 hour.

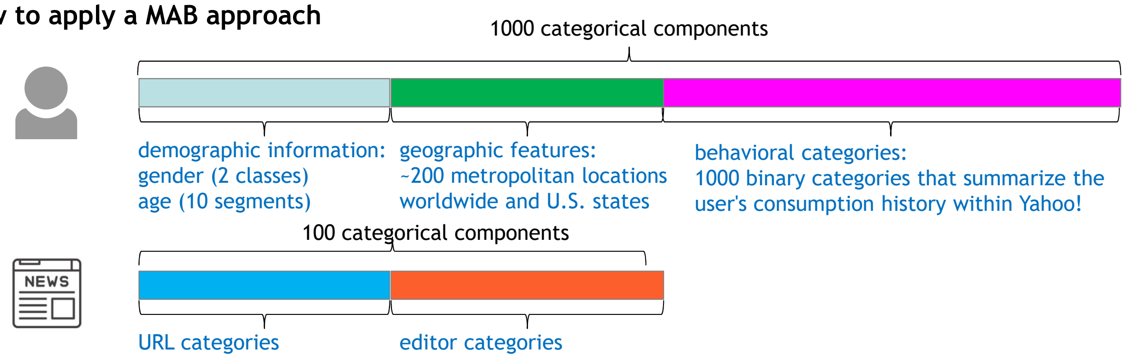
Example: Personalized News

How to apply a MAB approach

2. **Algorithm with warm start.**
   - Provide an offline-estimated user-specific adjustment on articles' context-free CTRs over the whole traffic.
   - The offset serves as an initialization on CTR estimate for new content, a.k.a."warm start".
   - The algorithm adds the user-specific CTR correction to the article's context-free CTR estimate.

Example: Personalized News

How to apply a MAB approach



3. Personalization with Context-Free Bandits.

- All users are partitioned into five groups
- All items are partitioned into five groups
- In each user group, a separate copy of the bandit algorithm is run.

83

83

Example: Personalized News

How to apply a MAB approach

4. Personalization with Contextual Bandits.

- All users are partitioned into five groups (a.k.a. user segments)
- All items are partitioned into five groups
- An algorithm such as LinUCB is run

84

84

LinUCB: Contextual Bandit Algorithm

Expected reward of item *a* at round *t*

$$\mathrm{E}[r_{t,a} | x_{t,a}] = x_{t,a}^{\top}\theta_a$$

context

Unknown co-efficient vector

Linear payoff

At trial t: D_a Matrix: contexts observed previously for item *a*

c_a Response vector

After ridge regression $\hat{\theta}_a = \underbrace{(\mathbf{D}_a^{\top}\mathbf{D}_a + \mathbf{I}_d)}_{A_a}{}^{-1}\underbrace{\mathbf{D}_a^{\top}\mathbf{c}_a}_{b_a}$

- Pick action that maximizes standard deviation

$$a_t = argmax_{a \ in \ A_t} (\underbrace{x_{t,a}^{\top}\hat{\theta}_a}_{\text{Predicted payoff}} + \alpha \underbrace{\sqrt{x_{t,a}^{\top}A_a^{-1}x_{t,a}}}_{\text{Standard deviation of payoff}})$$

LinUCB: Contextual Bandit Algorithm

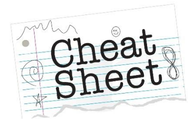
Pros:

- Complexity is linear in the number of arms and at most cubic in the number of features
- Works well for a dynamic arm set

Cons:

- Efficient as long as the size of the arm set is not too large

Multi-armed Bandits



Multi-armed bandit algorithms trade off *exploitation* and *exploration* in order to minimize regret.

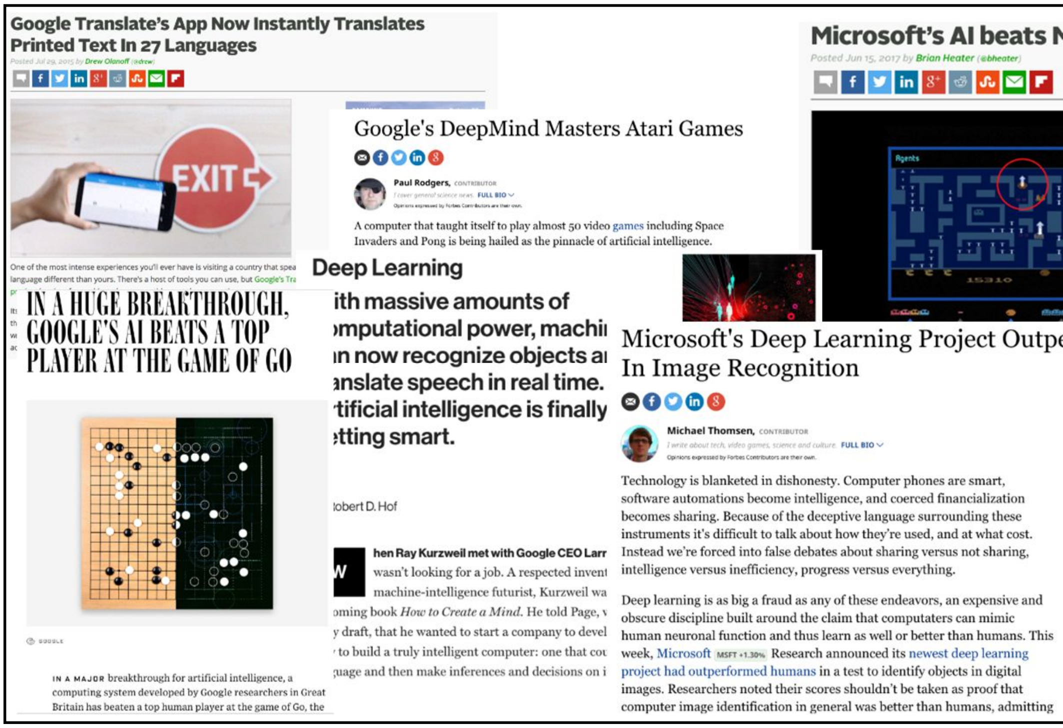
Problem setup:

(1) Observe user context
(2) Choose action (arm) to present user
(3) Observe reward
(4) Update p(reward|action, context)

Model types: contextual vs. non-contextual
Sampling strategies: epsilon-greedy, UCB, Thompson Sampling

Advantages	Concerns
• Gracefully handles item churn • Faster experimentation • Learns directly from user feedback • Randomness increases coverage	• **Size of repository** • **Lifetime of items** • **Performance**

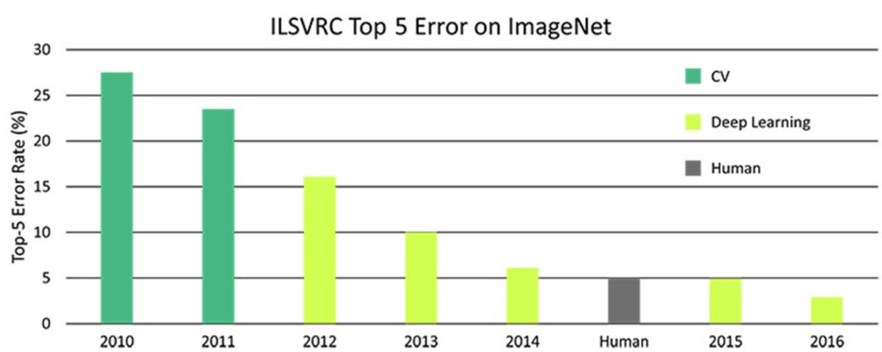
Deep learning



Deep Learning Is Making Waves Everywhere!

89

In many domains, deep learning is achieving near-human or super-human accuracy!



ILSVRC : ImageNet Large Scale Visual Recognition Challenge

However, application of Deep Learning in Recommender Systems is at its infancy.

90

So, what is Deep Learning?

A class of machine learning algorithms:

- that use a cascade of multiple non-linear processing layers
- and complex model structures
- to learn different representations of the data in each layer
- where higher level features are derived from lower level features to form a hierarchical representation.

Balázs Hidasi, RecSys 2016

91

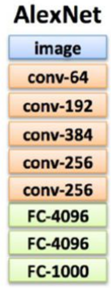
How deep is deep

Typically depth is driven by the complexity of the problem at hand. No hard and fast rule.

Deep Networks must be properly regularized.

(2012)

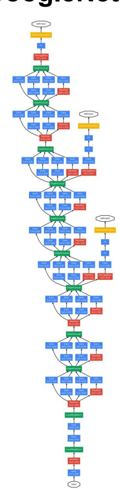
AlexNet



8 layers
16.4%

(2014)

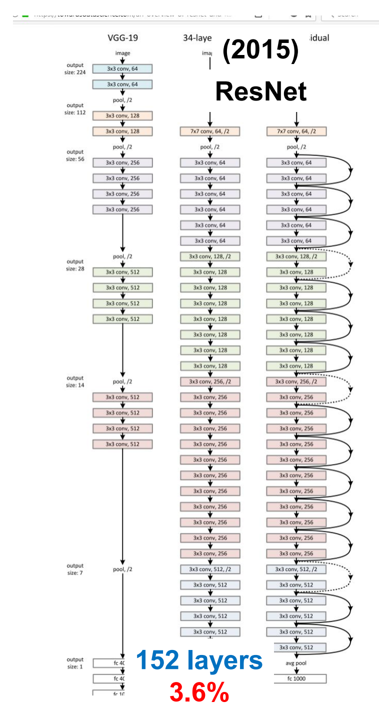
GoogleNet



22 layers
6.7%

(2015)

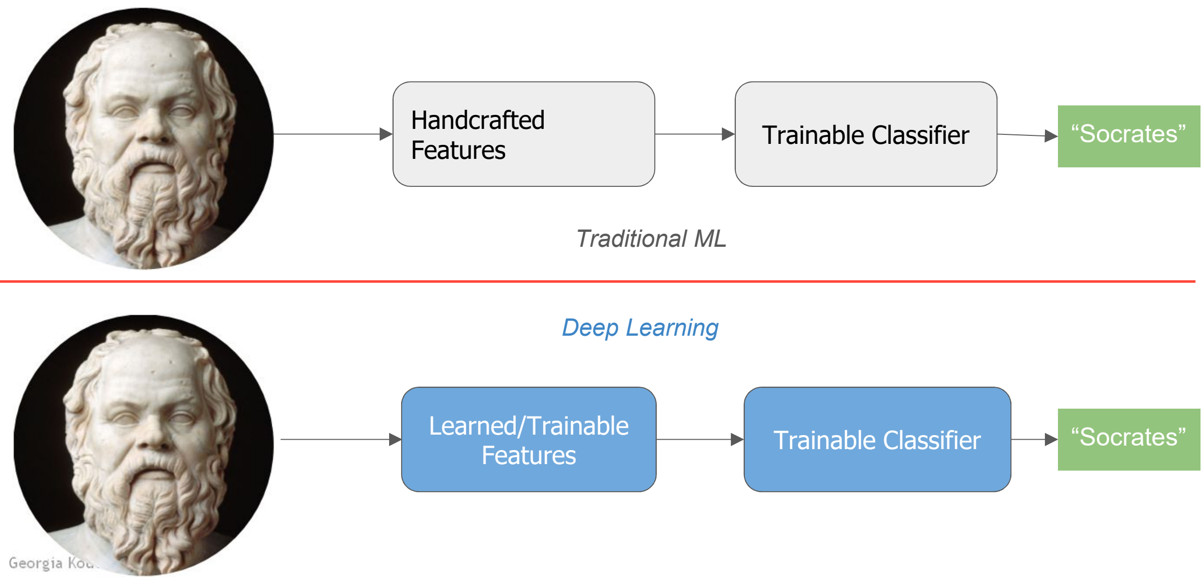
ResNet



152 layers
3.6%

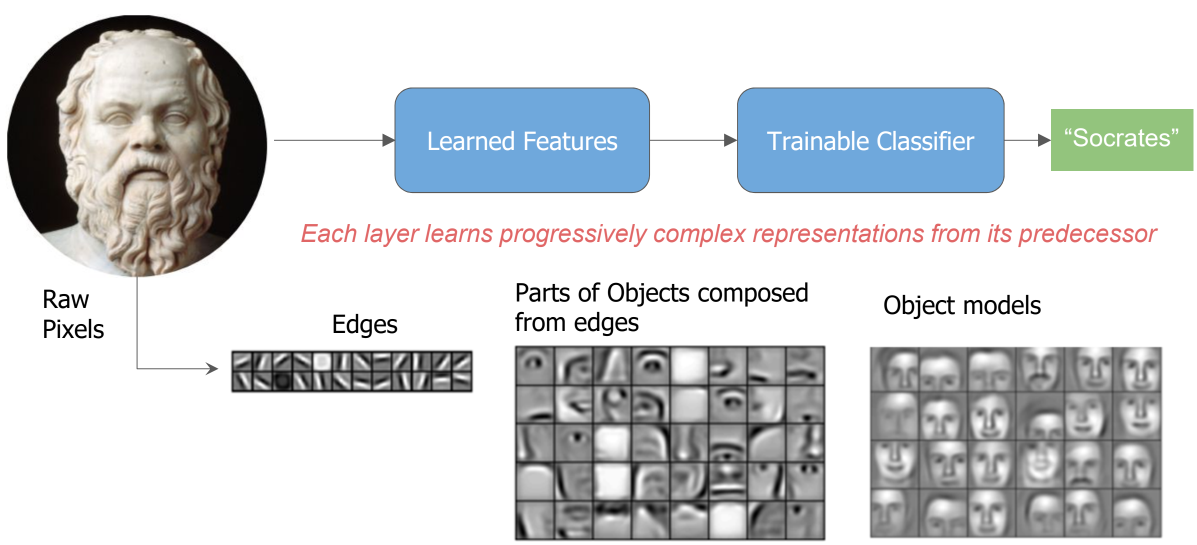
92

Traditional vs Deep



93

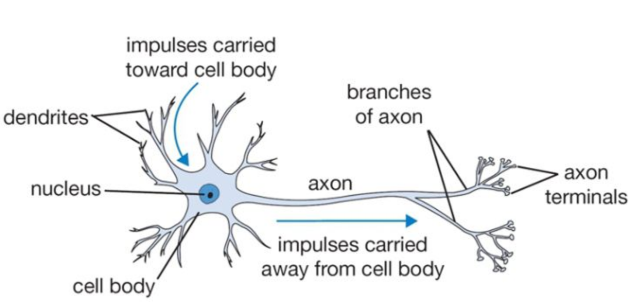
Learning hierarchical representations of data

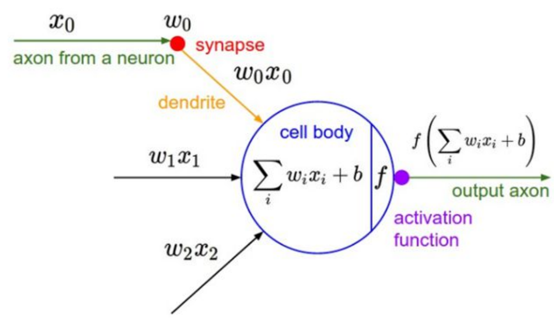


94

The *Neuron* is the fundamental unit of Deep Nets

It is extremely loosely inspired by the model of the biological neuron

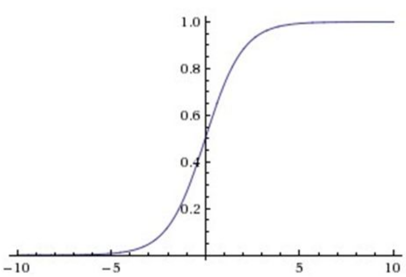




From Andrej Karpathy http://cs231n.github.io/neural-networks-1/

95

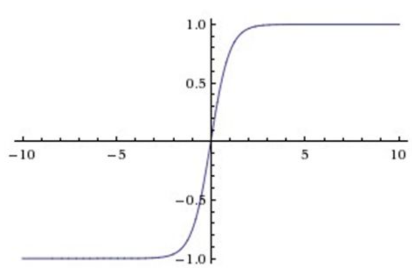
Common Activation Functions (or non-linearity)



sigmoid

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

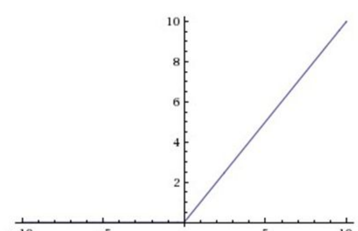
Rarely used - saturates and kills gradients



tanh

$$\tanh(x) = 2\sigma(2x) - 1$$

Zero-centered, preferred to sigmoid



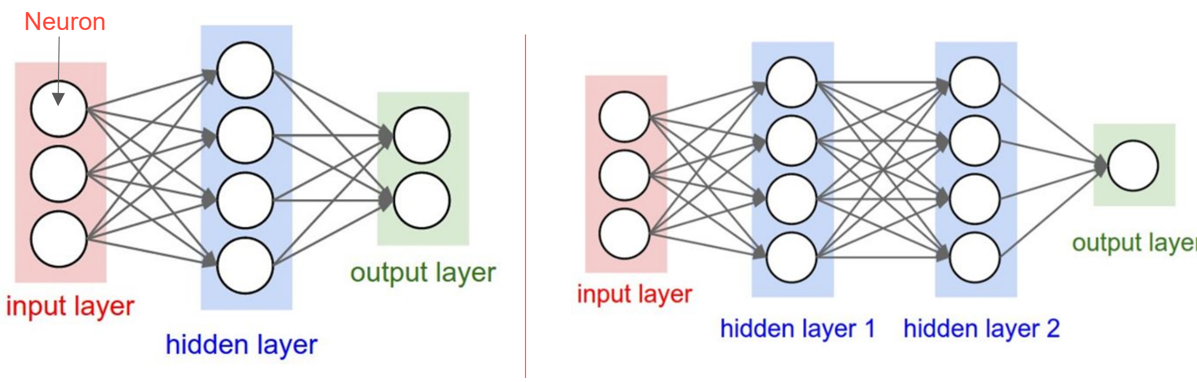
ReLU

$$f(x) = \max(0, x).$$

Very popular - faster, but needs careful setting of learning rate.

96

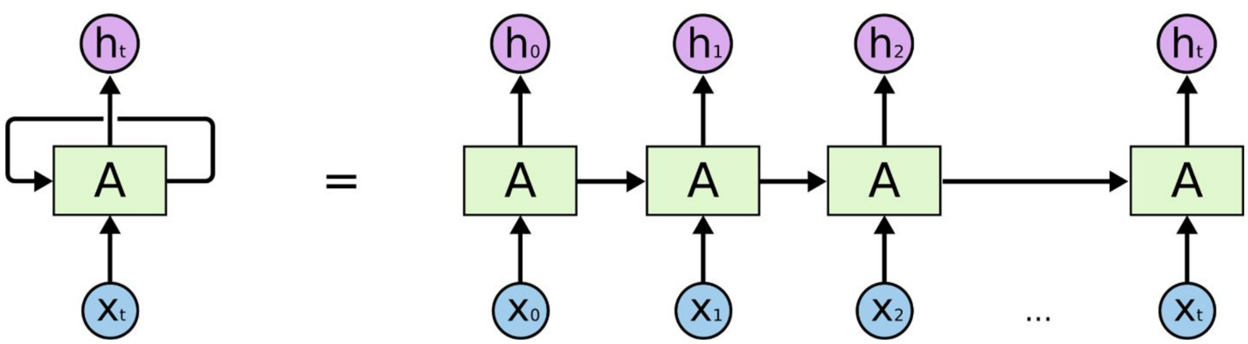
Fully Connected Layers - Feed-forward Networks



Note: There are connections between neurons across layers but not between neurons within a layer.

97

Recurrent Neural Networks - Sequence Modeling



A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor.

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

98

Recurrent Neural Networks can generate fake Shakespeare and LaTeX code that compiles!

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

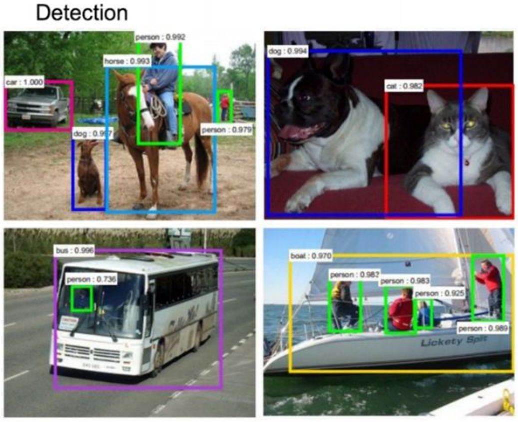
Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,\acute{e}tale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

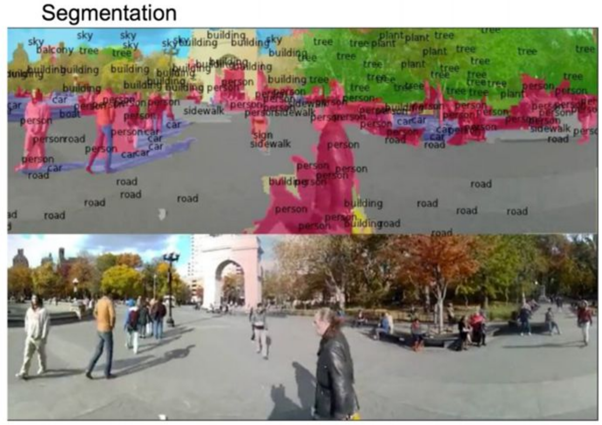
99

Convolutional Neural Nets

Detection



Segmentation



[Faster R-CNN: Ren, He, Girshick, Sun 2015]

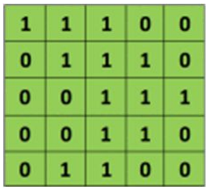
[Farabet et al., 2012]

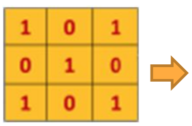
100

Convolutional Neural Nets

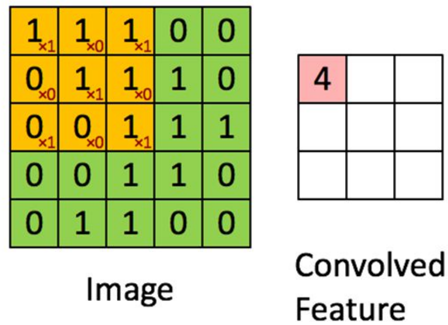
Step1: "Convolution" operator

Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data





Filter for:
- Edge detection
- Sharpen
- Box blur
- etc

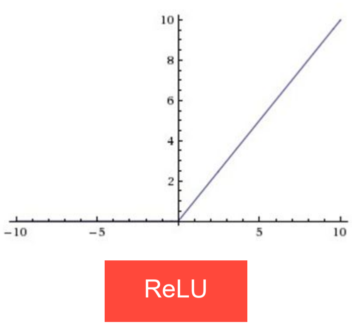


http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

101

Convolutional Neural Nets

Step2: Introducing Non Linearity (ReLU)



$$f(x) = \max(0, x).$$

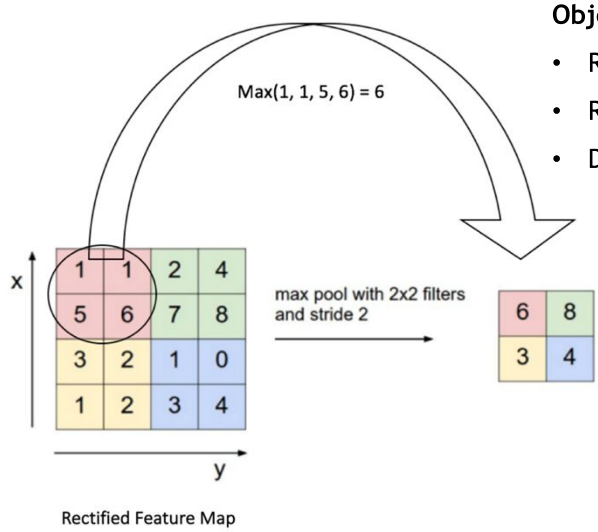
Objective: Introduce non-linearity in the ConvNet:

- Convolution is a linear operation – element wise matrix multiplication and addition
- But most of the real-world data would be non-linear
- So we account for non-linearity by introducing a non-linear function like ReLU).

102

Convolutional Neural Nets

Step3: Pooling



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

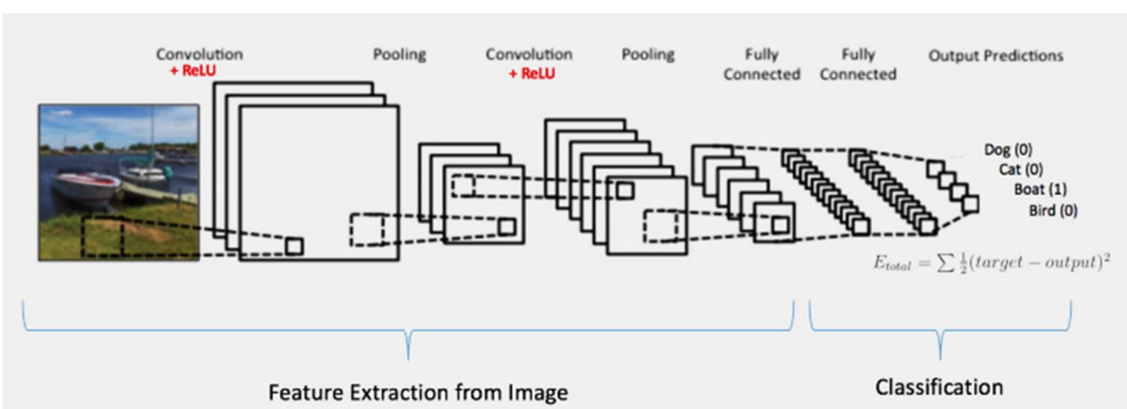
6	8
3	4

Objective:

- Reduces the dimensionality of each feature map
- Retains the most important information.
- Different types: Max, Average, Sum etc.

103

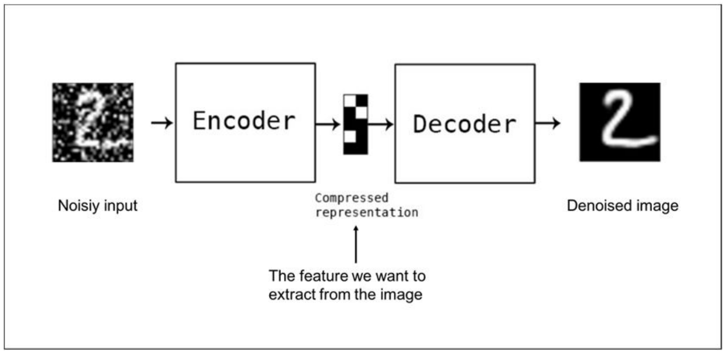
Convolutional Neural Nets

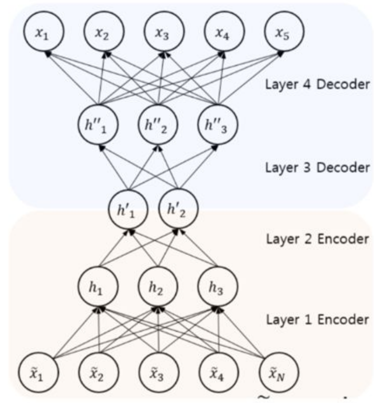


•Input Image = Boat
•Target Vector = [0, 0, 1, 0]

104

Denoising autoencoders





- The principle is to be able to reconstruct data from an input of corrupted data.
- After giving the autoencoder the corrupted data, we force the hidden layer to learn only the more robust features, rather than just the identity.
- The output will then be a more refined version of the input data

Why use deep learning in recommender systems?

- Better generalization beyond linear models for user-item interactions.
- Unified representation of heterogeneous signals (e.g. add image/audio/textual content as side information to item embeddings via convolutional NNs).
- Exploitation of sequential information in actions leading up to recommendation (e.g. LSTM on viewing/purchase/search history to predict what will be watched/purchased/searched next).
- DL toolkits provide unprecedented flexibility in experimenting with loss functions (e.g. in toolkits like TensorFlow/MxNet/Keras etc. switching the loss from classification loss to ranking loss is trivial. The optimization is taken care of.)

Collaborative Denoising Auto-Encoder

- Treats the feedback on items **y** that the user **U** has interacted with (input layer) as a noisy version of the user's preferences on all items (output layer)
- Introduces a user specific input node and hidden bias node, while the item weights are shared across all users.

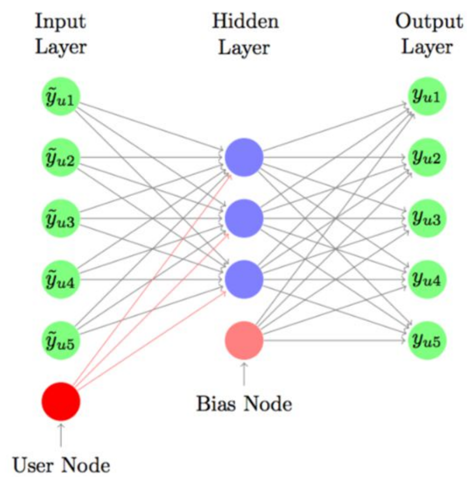


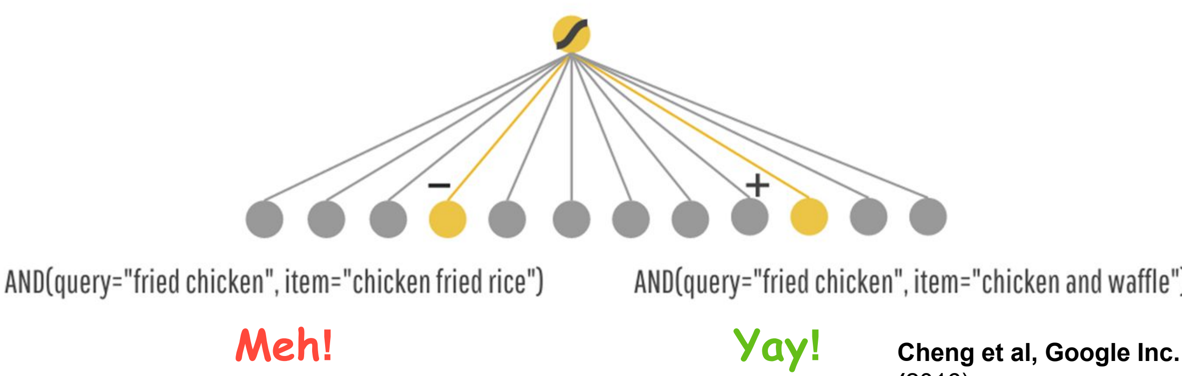
Figure 1: A sample CDAE illustration for a user u. The links between nodes are associated with different weights. The links with *red* color are user specific. Other weights are shared across all the users.

Collaborative Denoising Auto-Encoders for Top-N Recommender Systems, Wu et.al., WSDM 2016

107

Wide + Deep Models for Recommendations

Memorization
In a recommender setting, you may want to train with a wide set of cross-product feature transformations , so that the model essentially memorizes these sparse feature combinations (rules):

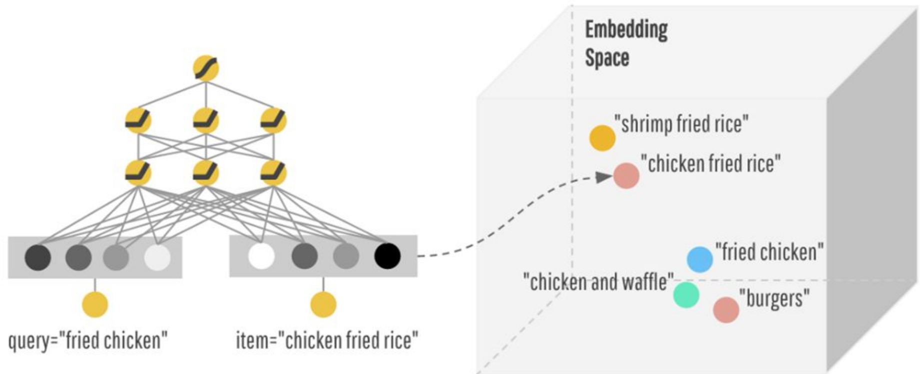


Cheng et al, Google Inc. (2016)

108

Wide + Deep Models for Recommendations

Generalization
On the other hand, you may want the ability to generalize using the representational power of a deep network. But deep nets can overgeneralize.



Cheng et al, Google Inc.
(2016)

109

Wide + Deep Models for Recommendations

One challenge in recommender systems is to achieve **both memorization and generalization.**

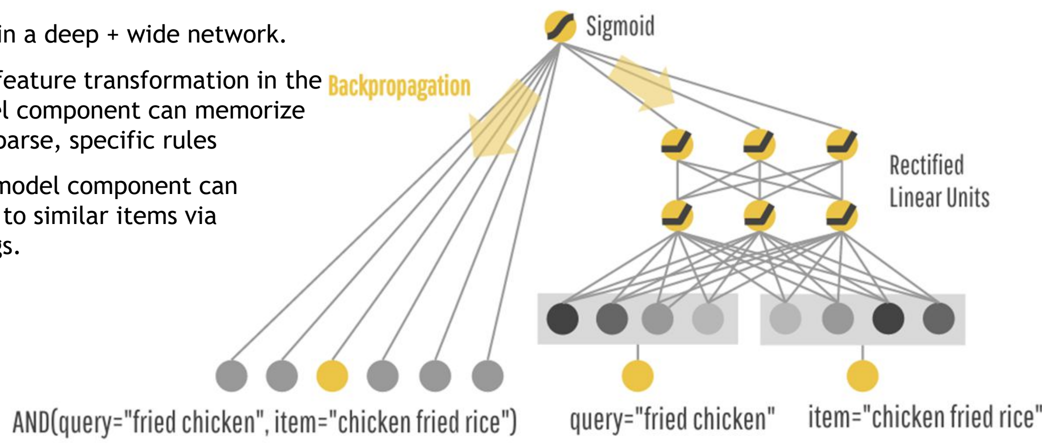
- Recommendations based on memorization tend to be
  more topical and directly relevant to the items which users have already interacted with.
- Generalization tends to improve the diversity of the recommended items.

110

Wide + Deep Models for Recommendations

Best of both worlds:

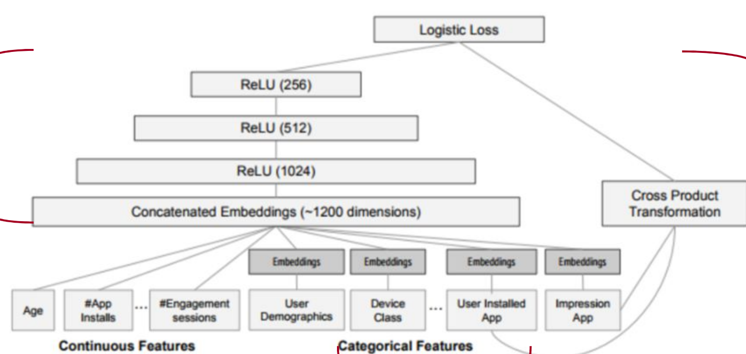
- Jointly train a deep + wide network.
- The cross-feature transformation in the wide model component can memorize all those sparse, specific rules
- The deep model component can generalize to similar items via embeddings.



111

Google app store recommendations

- The two models are combined using a weighted sum of their output log odds as the prediction
- This is fed to one common logistic loss function for joint training



The deep model is a feed-forward NN

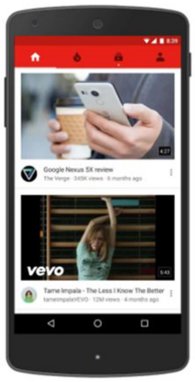
The wide model is a linear model
$y = \mathbf{w}^T \mathbf{x} + b$

Each sparse, high-dimensional categorical feature is converted into a low-dimensional and dense real-valued embedding vector

Cheng et al, Google Inc. (2016)

112

The Youtube Recommendation model



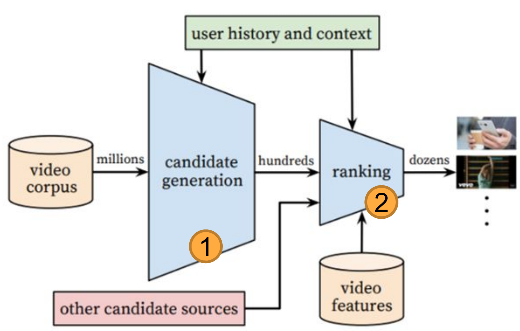
Recommending YouTube videos is extremely challenging:

- **Scale**:
  - Many existing recommendation algorithms fail to handle YouTube's massive user base and corpus.
- **Freshness**:
  - Many hours of video are uploaded per second.
  - The recommendation system should be responsive enough to model newly uploaded content as well as the latest actions taken by the user.
- **Noise**:
  - Rarely obtain the ground truth of user satisfaction and instead model noisy implicit feedback signals.
  - Metadata associated with content is poorly structured without a well defined ontology.

Covington et al., Google Inc. (2016)

113

The Youtube Recommendation model



A two-stage approach with two deep networks:

Candidate generation network

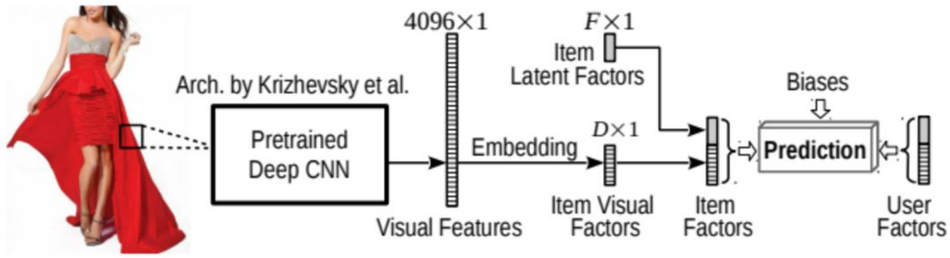
- takes events from the user's activity history as input
- retrieves a small subset (hundreds) of videos
- These candidates are intended to be generally relevant to the user with high precision.
- Provides broad personalization via collaborative filtering.

Ranking network

- scores each video using a rich set of features describing the video and user.
- The highest scoring videos are presented to the user, ranked by their score

114

VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback



Helping cold start with augmenting item factors with visual factors

- Create an item Factor that is a sum of two terms: An Item Visual Factor which is an embedding of a Deep CNN on the item image, and the usual collaborative item factor.

He *et al.*, AAAI (2016)

115

The Pinterest Application: **Pin2Vec** Related Pins

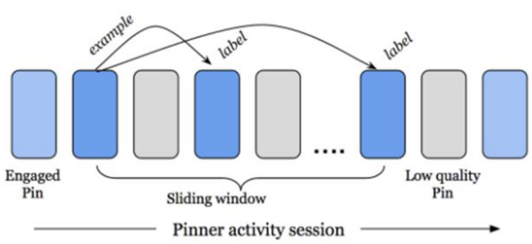


Figure 3. Extract Pin2Vec training pairs from Pinner activity history.

Learn a 128 dimensional compressed representation of each item (embedding). Then use a similarity function (cosine) between them to find similar items.

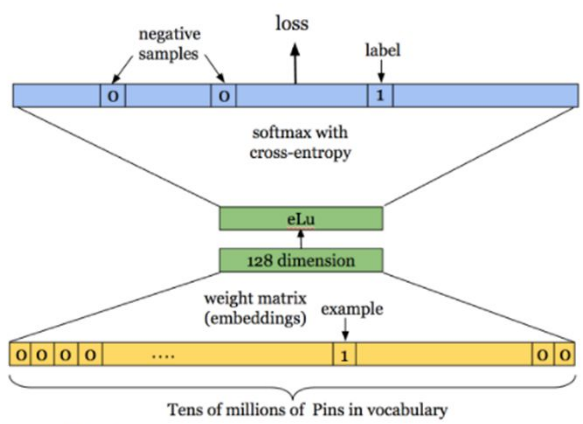


Figure 4. Feedforward neural network architecture of training Pin2Vec.

Liu et al (2017)

https://medium.com/the-graph/applying-deep-learning-to-related-pins-a6fee3c92f5e

116

The Pinterest Application: **Pin2Vec** Related Pins

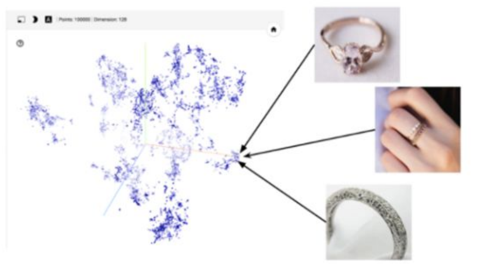


Figure 5. 3D presentation of 128-dimension embeddings of Pins using t-SNE. Each point is a Pin. Related pins are clustered together. The figure shows "an island of wedding rings."

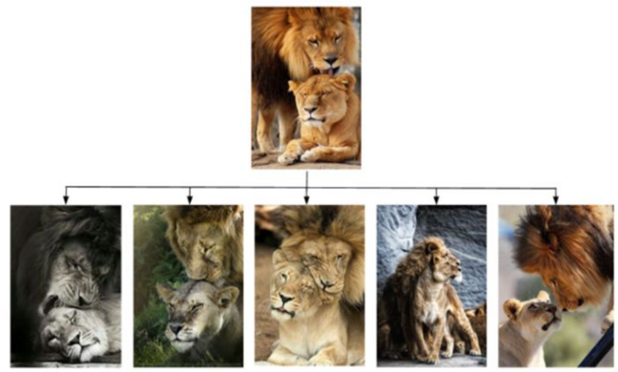


Figure 6. Related Pins generated using Pin2Vec. These are top Pins ranked by their euclidean distances to the query Pin in an ascending order.

https://medium.com/the-graph/applying-deep-learning-to-related-pins-a6fee3c92f5e

Liu et al (2017)

117

The Pinterest Application: **Pin2Vec** Related Pins

Bridging the Gap:

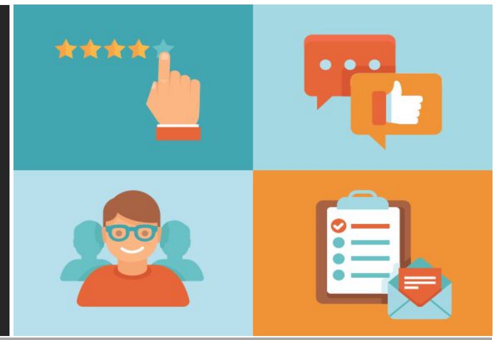
Board co-occurrence only found images of bottled wines, whereas Pin2Vec found recommendations for drinks made with wine. This suggests Pinners actually saved the bottled wine Pin and the wine cocktail Pins in the same *time series*.

More relevant recommendations are possible because of the representations learned by the model.

Figure 7. The Related Pins generated using board co-occurrence and Pin2Vec.

118

THANK YOU



119

119