UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Recommender Systems

Learning to Rank

Rodrygo L. T. Santos
rodrygo@dcc.ufmg.br

How to recommend?

Collaborative

- *Good accuracy and discovery*
- *Poor with sparse ratings*

Content-based

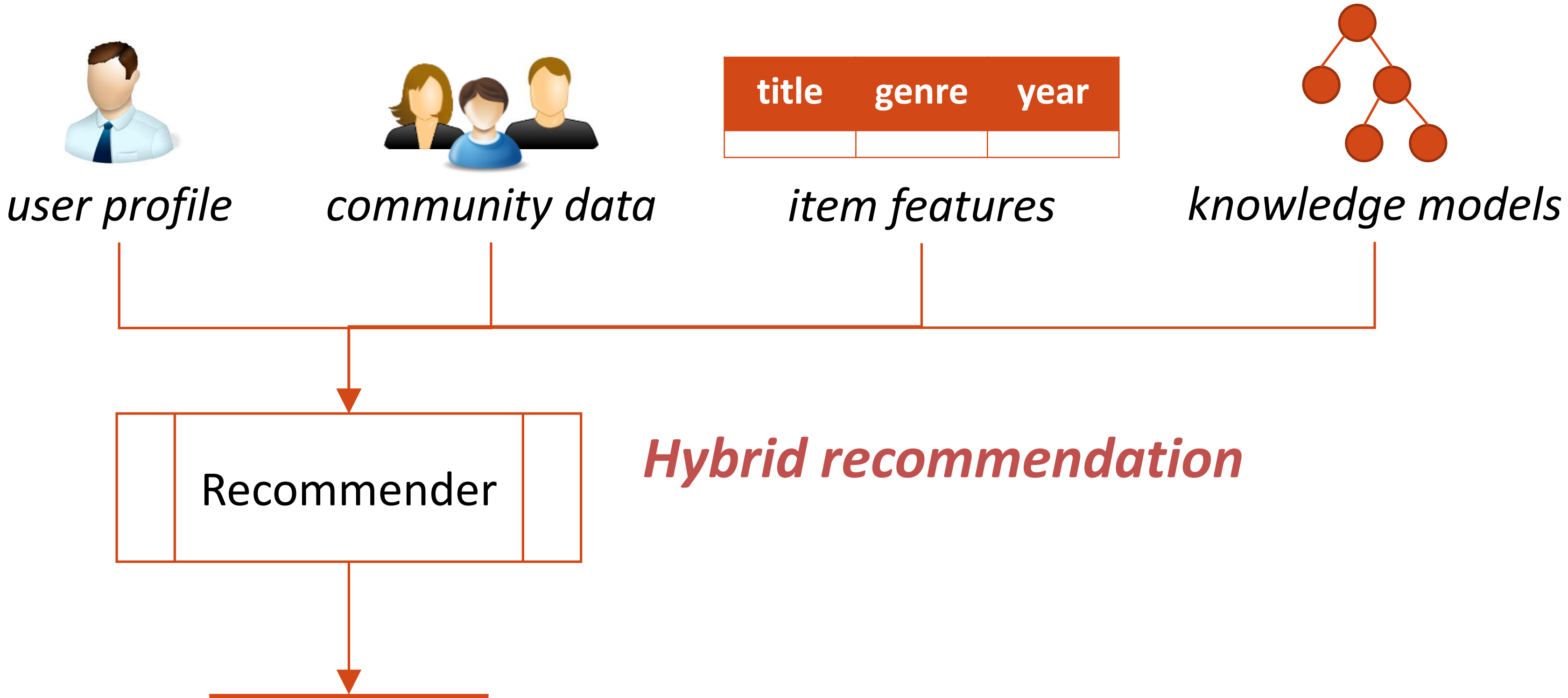
- *No need for item ratings*
- *Poor discovery*

Knowledge-based

- *No need for user or item ratings*
- *Costly knowledge acquisition*

NO
SILVER
BULLET

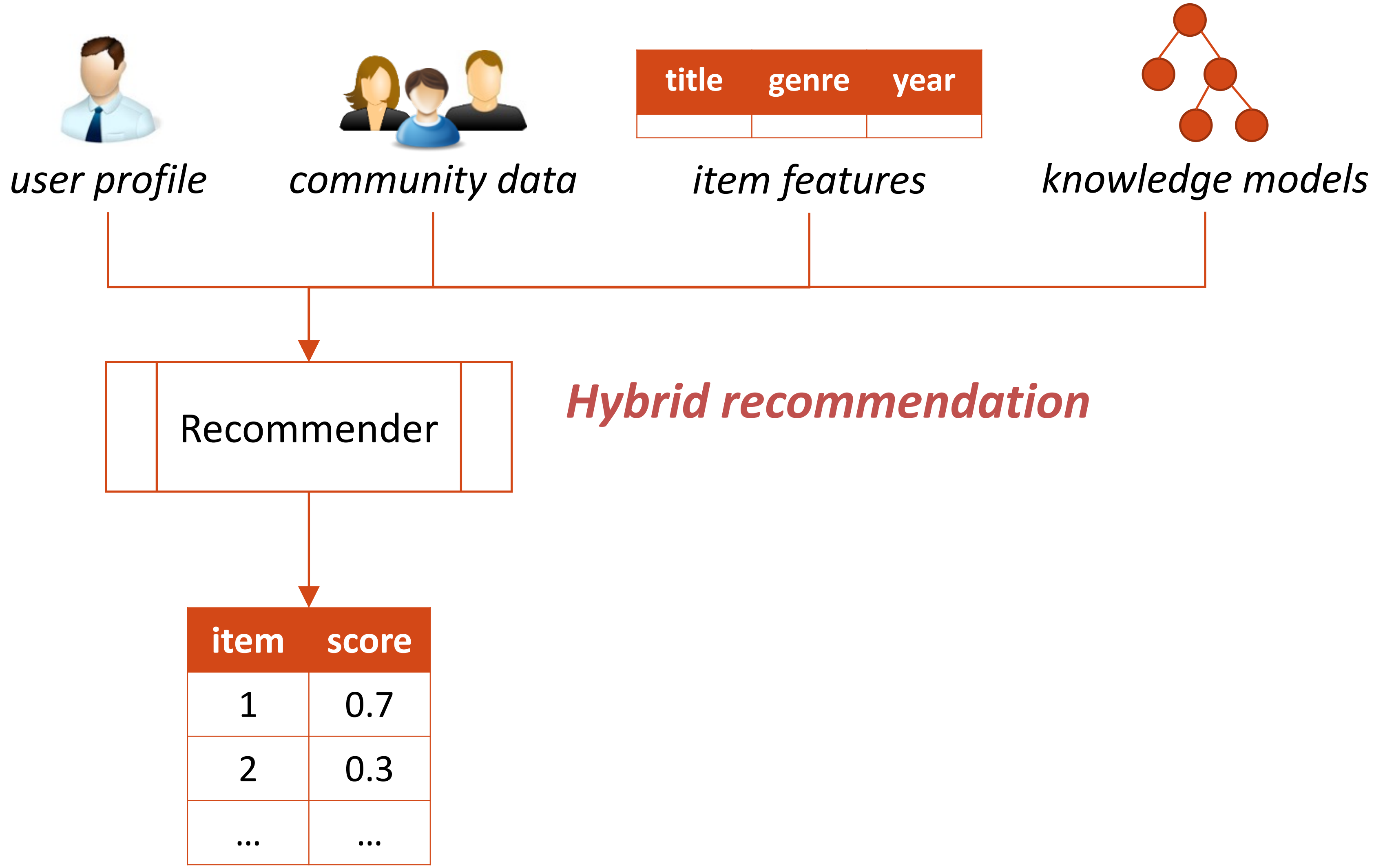
How to recommend?



Hybrid recommendation

item	score
1	0.7
2	0.3
…	…

How to recommend?



Hybrid recommendation

Learning hybrids

For a particular recommendation model

- Parameter tuning is usually difficult, especially when there are many parameters to tune

For a collection of models

- There are hundreds of models in the literature
  - Algorithms × instantiations

How to combine multiple models?

Ensembling the cues

Linear combination?

- $f(u,i) = \alpha_1 f_{\mathrm{CF}}(u,i) + \alpha_2 f_{\mathrm{CB}}(u,i)$

How to fit α_1 and $\alpha_2 = (1 - \alpha_1)$?

- $\{\alpha_1 = 0.3, \alpha_2 = 0.7\} \rightarrow \{\mathrm{MAP} = 0.2, \mathrm{nDCG} = 0.6\}$
- $\{\alpha_1 = 0.5, \alpha_2 = 0.5\} \rightarrow \{\mathrm{MAP} = 0.1, \mathrm{nDCG} = 0.5\}$
- $\{\alpha_1 = 0.7, \alpha_2 = 0.3\} \rightarrow \{\mathrm{MAP} = 0.4, \mathrm{nDCG} = 0.7\}$ ✓

What if we have thousands of models?

> “
>
> *Mr. Singhal has developed a far more elaborate system for ranking pages, which involves more than 200 types of information, or what Google calls “signals.”*
>
> ◦ Saul Hansell, New York Times, June 2007

The recommendation problem

$$f(u,i)$$

Learning to rank

$$f(\mathbf{x})$$

Learning to rank

Feature-based representation

- Individual models as ranking ”features”

Discriminative learning

- Effective models learned from data
- Aka machine-learned ranking

Learning to rank

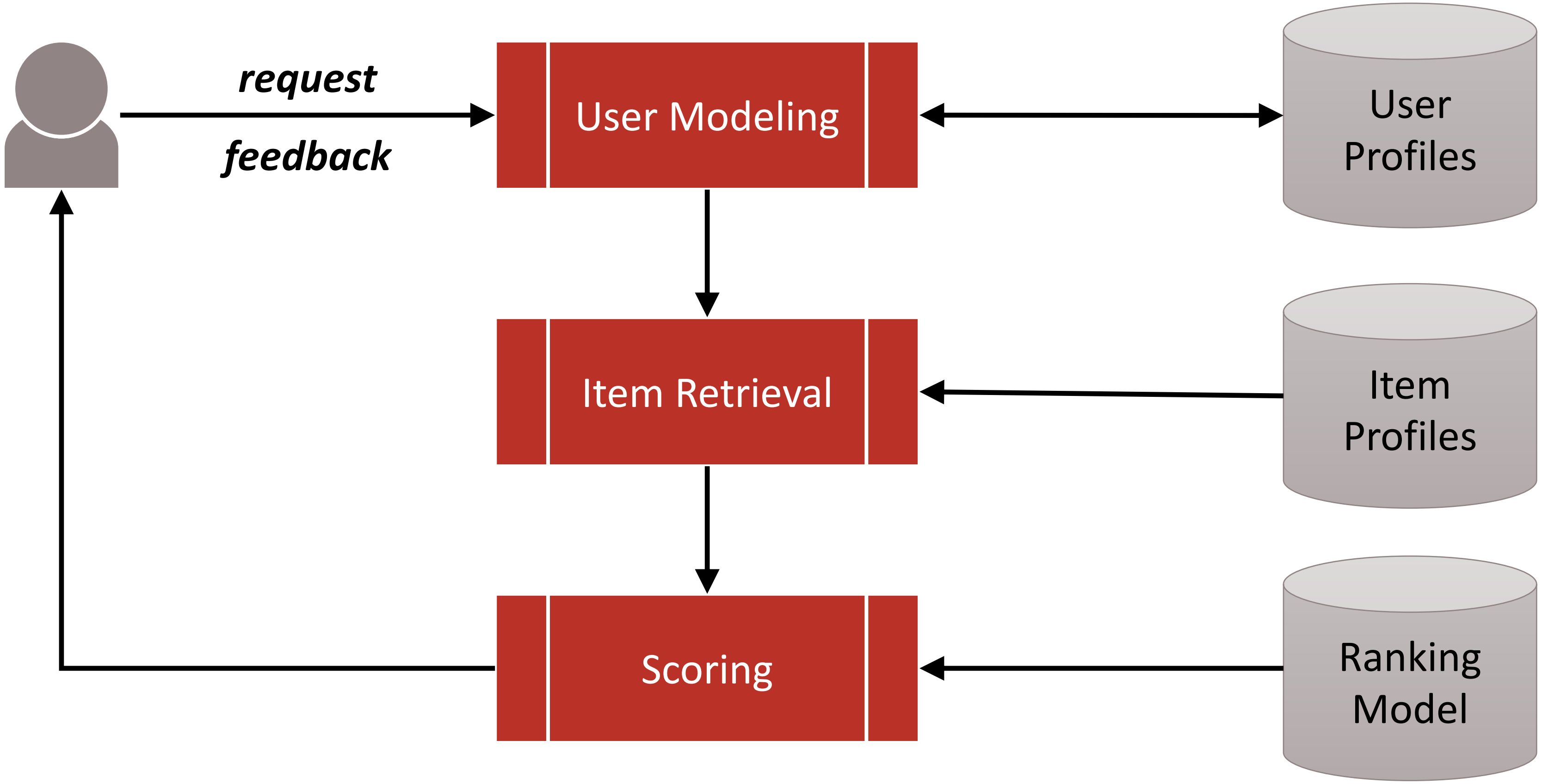
Actively researched over the last decade or so

- Both by academia as well as industry players

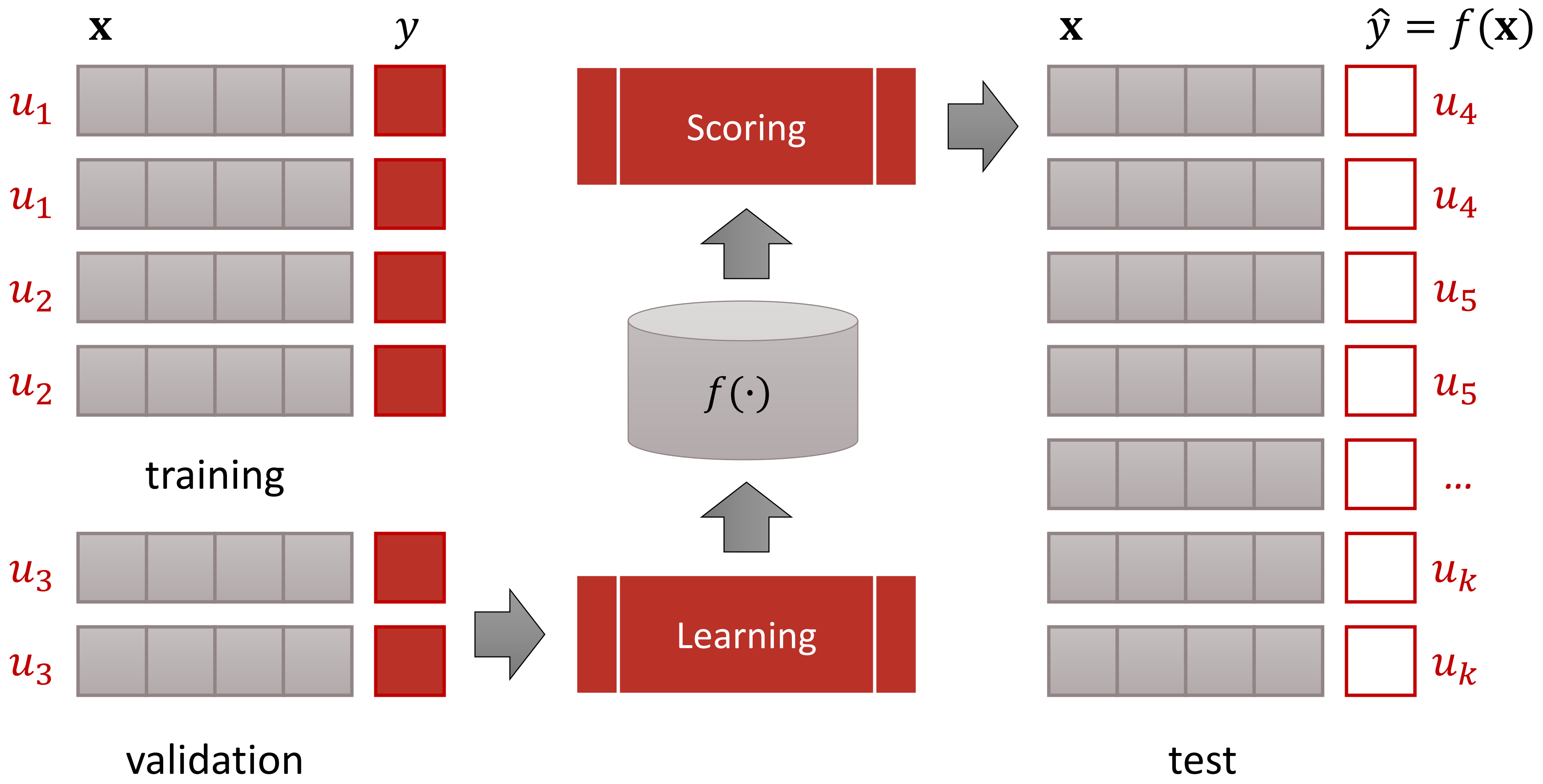
Why didn’t it happen earlier?

- Limited availability of training data
- Poor machine learning techniques
- Too few features to show value

Personalized recommendation



Discriminative learning framework



Building blocks

Goal is to learn a ranking model

- $f : \mathcal{X} \rightarrow \mathcal{Y}$

That minimizes some loss function

- $\mathcal{L} : f(\mathcal{X}) \times \mathcal{Y} \rightarrow \mathcal{R}$

$\mathcal{X}$: input space

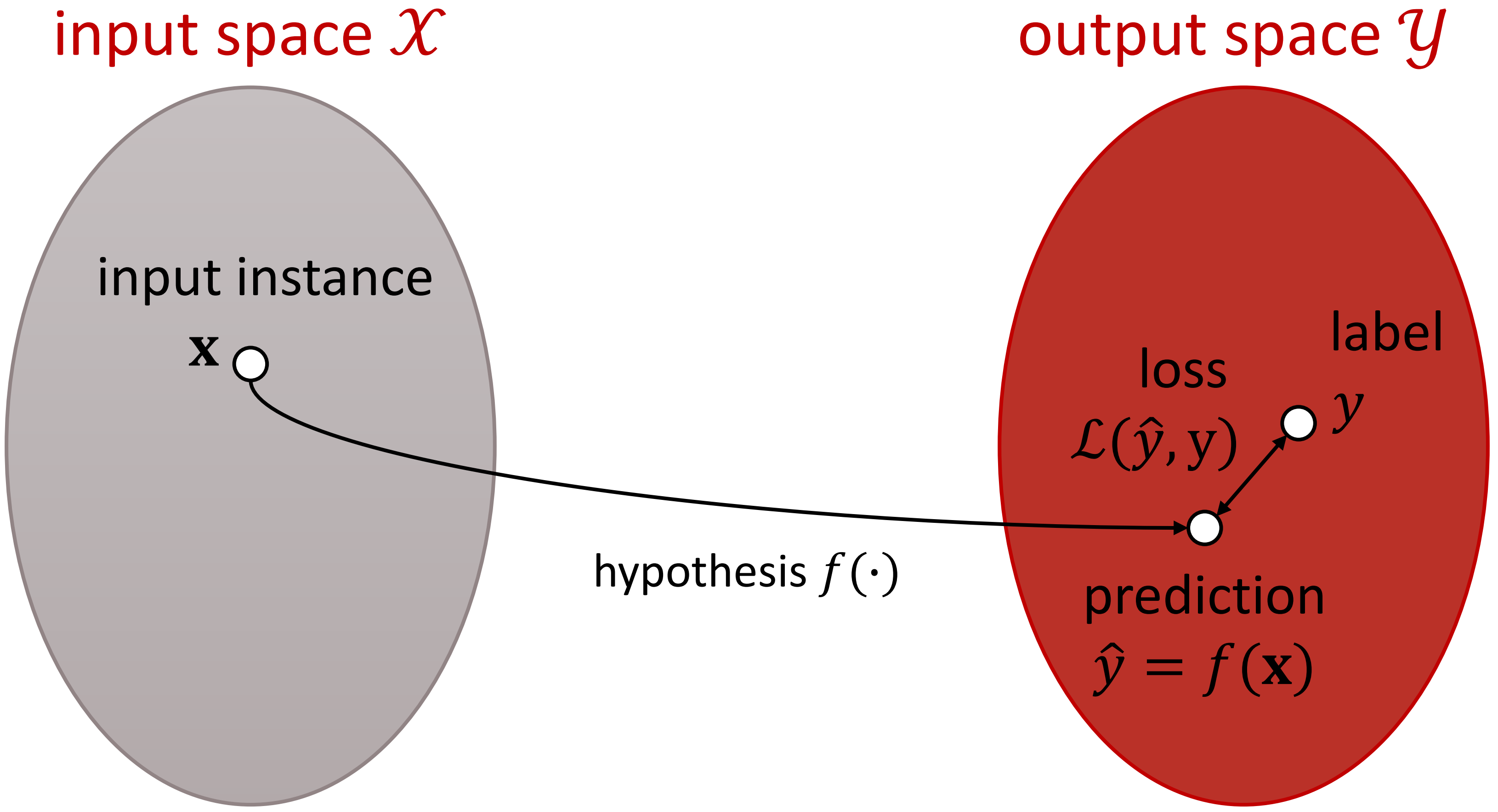
$\mathcal{Y}$: output space

$\mathcal{F}$: hypothesis space

$\mathcal{L}$: loss function

$\mathcal{O}$: optimizer

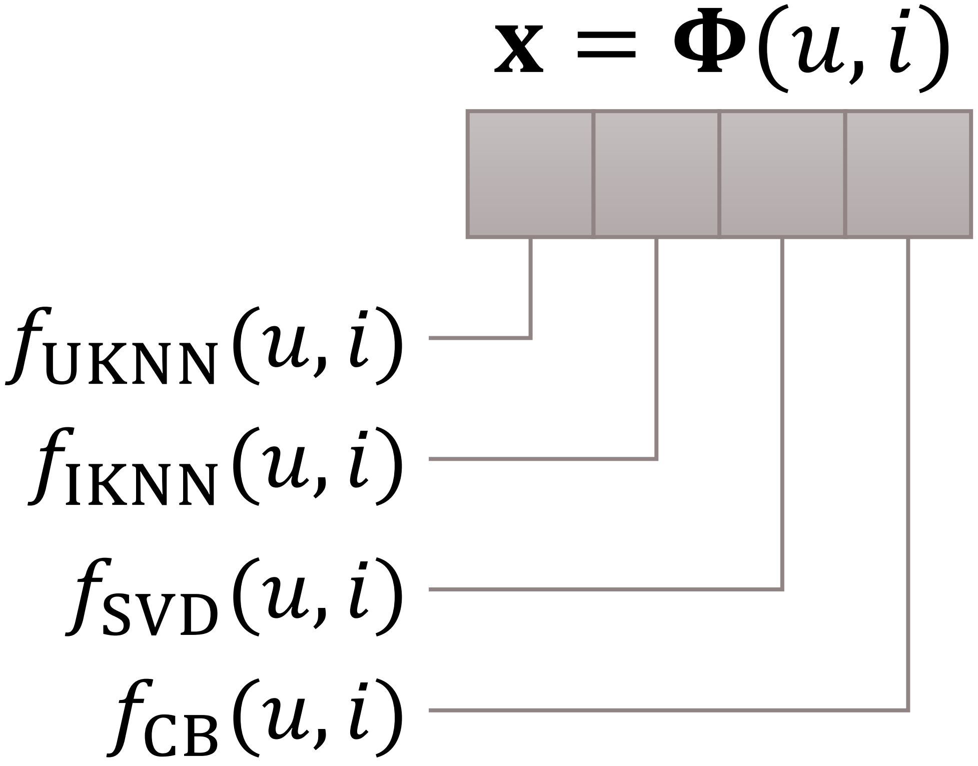
Building blocks



Input space ($\mathcal{X}$)

LTR takes as input feature vectors

- $\mathbf{x} \in \mathcal{X}$



Ranking features

User- or item-dependent (depend on $\boldsymbol{u}$ or $\boldsymbol{i}$)

- User demographics, item popularity

User-item dependent (depend on $\langle \boldsymbol{u}, \boldsymbol{i} \rangle$)

- Likelihood of click, similarity of categories

Context-dependent (further depend on context $\boldsymbol{c}$)

- Context prefiltering version of the above

Output space ($\mathcal{Y}$)

LTR may produce different outputs

- $y \in \mathcal{Y}$

Each scalar y labels a training instance

Data labeling alternatives

Labeling of individual items

- Binary judgments (non-rel, relevant)
  $y \in \{0,1\}$
- Graded judgments (non-rel, …, perfect)
  $y \in \{0,1,2,3,4,5\}$

Data labeling alternatives

Labeling of pairs of items

- Implicit judgments

i_1

i_2

i_3 ✓

i_4

i_5

$i_3 \succ i_1$

$i_3 \succ i_2$

$\cancel{i_3 \succ i_4}$

$\cancel{i_3 \succ i_5}$

Data labeling alternatives

Creation of list

- List (or permutation) of items is given
- Ideal, but difficult to implement

Hypothesis space ($\mathcal{F}$)

Goal is to learn a ranking model

- $f : \mathcal{X} \to \mathcal{Y}$

A hypothesis $f \in \mathcal{F}$ could be any function

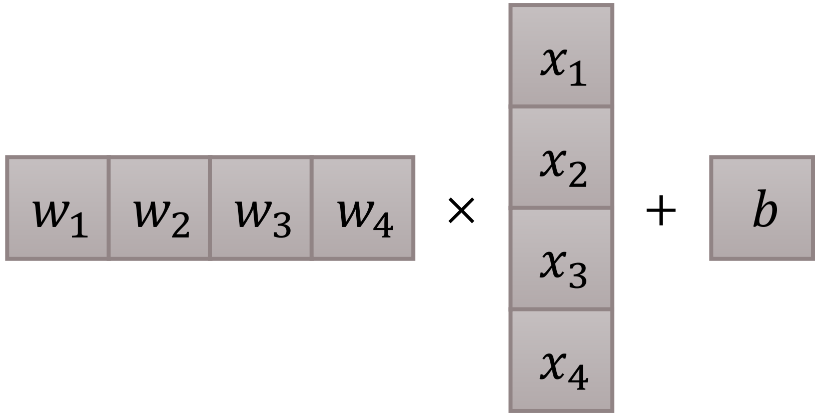
- Linear functions
- Non-linear functions (trees, networks)

Hypothesis space ($\mathcal{F}$)

Linear hypotheses

$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \mathrm{b}$

- $\mathbf{w}$ is a weight vector
- b is a scalar bias

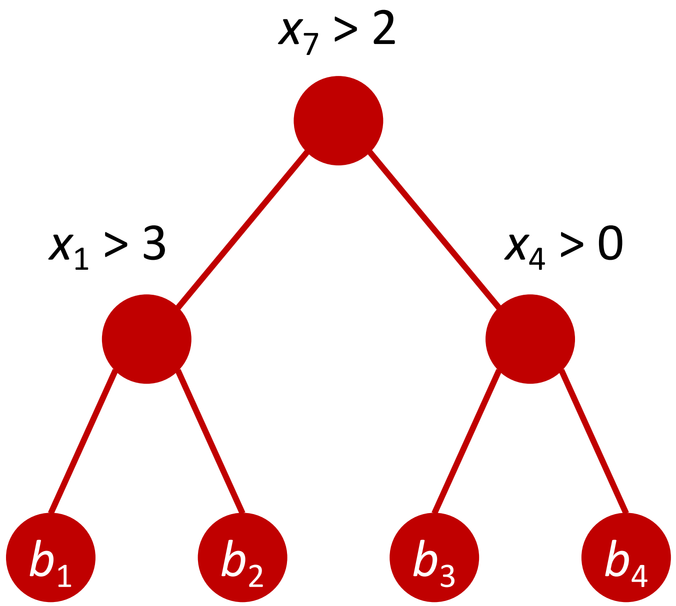


Hypothesis space ($\mathcal{F}$)

Tree-based hypotheses

$f(\mathbf{x}) = \sum_k b_k \mathbf{1}(\mathbf{x} \in R_k)$

- k is one of the leaves in the tree
- b_k is the value predicted in region R_k
- $\mathbf{1}(\cdot)$ is the indicator function

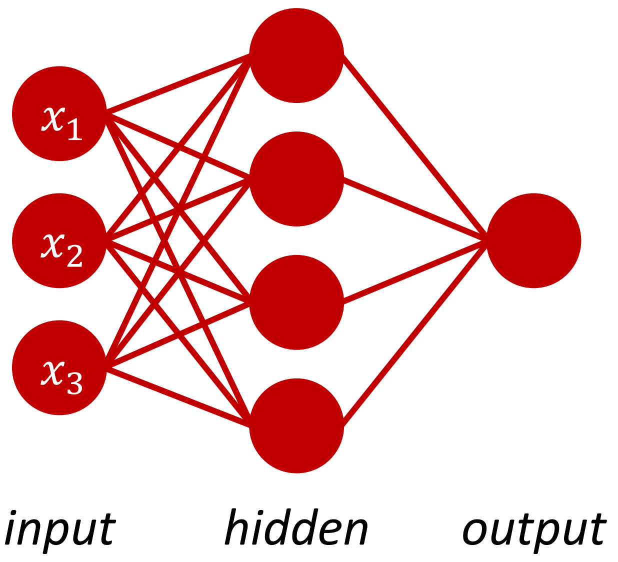


Hypothesis space ($\mathcal{F}$)

Neural network hypotheses

$$f(\mathbf{x}) = \sigma_2(\mathbf{W}_2 \underbrace{\sigma_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)}_{\text{1st layer output}} + \mathbf{b}_2)$$

$$\underbrace{\sigma_2(\mathbf{W}_2\sigma_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2)}_{\text{2nd layer output}}$$



- $\mathbf{W}_k$ is a weight matrix
- $\mathbf{b}_k$ is a bias vector
- σ_k is an activation function

Hypothesis space ($\mathcal{F}$)

Goal is to learn a ranking model

- $f : \mathcal{X} \to \mathcal{Y}$

A hypothesis $f \in \mathcal{F}$ could be any function

- Linear functions
- Non-linear functions (trees, networks)

There are infinitely many such functions

How to find the best $f(\mathbf{x})$?

Look for the one with minimum loss

Loss function ($\mathcal{L}$)

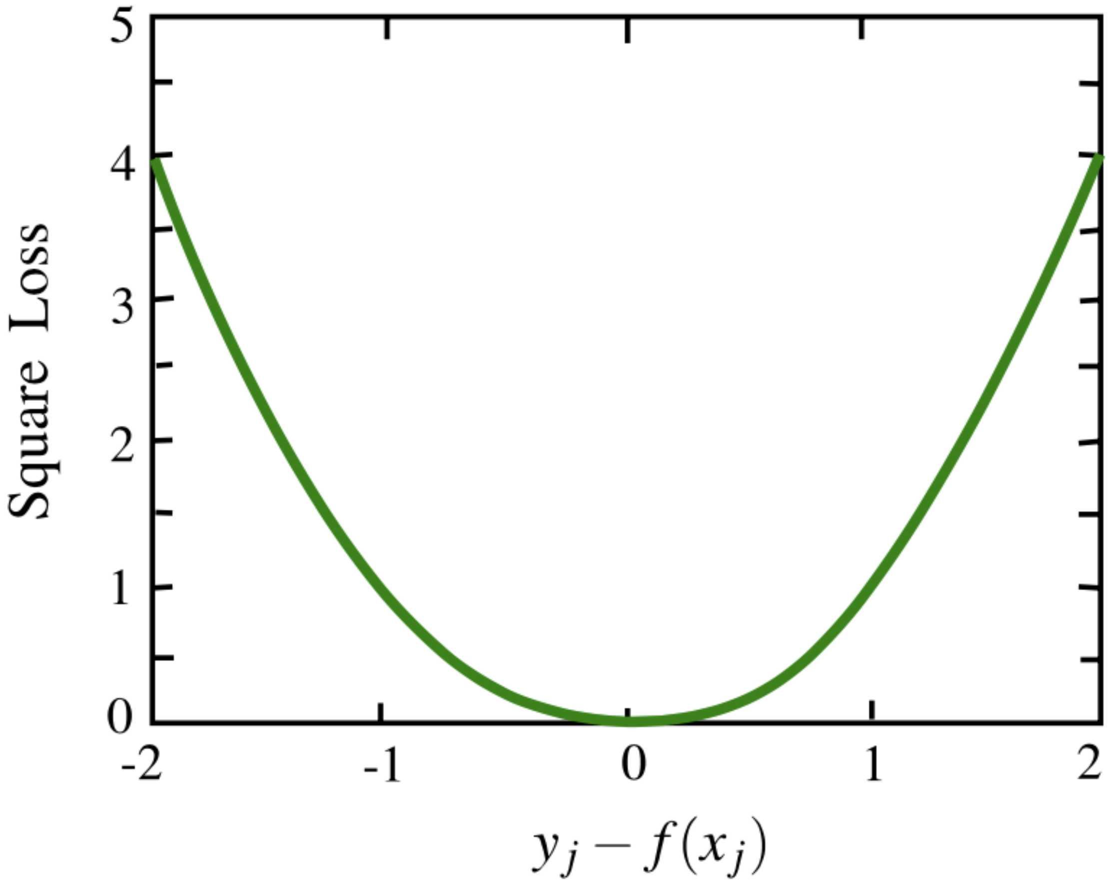
Loss as a measure of error

- $\mathcal{L}(\hat{y}, y) = \mathcal{L}(f(\mathbf{x}), y)$

Many options once again

- 0-1 loss: $\mathcal{L}(\hat{y}, y) = \mathbf{1}(y \neq f(\mathbf{x}))$
- Absolute loss: $\mathcal{L}(\hat{y}, y) = |y - f(\mathbf{x})|$
- Square loss: $\mathcal{L}(\hat{y}, y) = (y - f(\mathbf{x}))^2$

Example: square loss



How to find $f(\mathbf{x})$ that gives the minimum loss?

Hint:

- $f(\mathbf{x}; \mathbf{w})$ is actually parameterized by some $\mathbf{w}$

Optimizer

Coordinate methods

- Line search, one coordinate at a time

Gradient methods

- Walk downhill, all coordinates together

Boosting methods

- Upweight difficult examples

Learning algorithms

Query refinement (WWW 2008)

ListNet (ICML 2007)
SVM-MAP (SIGIR 2007)
Nested Ranker (SIGIR 2006)

LambdaRank (NIPS 2006)
Frank (SIGIR 2007)
MPRank (ICML 2007)

MHR (SIGIR 2007)
RankBoost (JMLR 2003)
LDM (SIGIR 2005)

RankNet (ICML 2005)
Ranking SVM (ICANN 1999)
IRSVM (SIGIR 2006)

Discriminative model for IR (SIGIR 2004)
SVM Structure (JMLR 2005)

GPRank (LR4IR 2007)
QBRank (NIPS 2007)
GBRank (SIGIR 2007)

McRank (NIPS 2007)
SoftRank (LR4IR 2007)
AdaRank (SIGIR 2007)

CCA (SIGIR 2007)
ListMLE (ICML 2008)
RankCosine (IP&M 2007)

Supervised Rank Aggregation (WWW 2007)
Relational ranking (WWW 2008)

Learning algorithms

Pointwise	$\mathcal{X}$: single items
	$\mathcal{Y}$: scores or class labels
Pairwise	$\mathcal{X}$: item pairs
	$\mathcal{Y}$: partial orders
Listwise	$\mathcal{X}$: item collections
	$\mathcal{Y}$: ranked item list

Pointwise approaches

Reduce ranking to regression or classification

- Assume relevance is user-independent

In practice, relevance is user-dependent

- Utility of a feature may also be user-dependent
- By putting items associated with different users together, the training process may be hurt

Pairwise approaches

Reduce ranking to classification on item pairs associated with the same user

- No longer assume independent relevance

Unique properties of ranking not fully covered

- Number of instance pairs varies across users
- Importance of errors varies across ranking positions

Listwise approaches

Perform learning directly on item list

- Treats ranked lists as learning instances

Two major approaches

- Define listwise loss functions
- Directly optimize ranking evaluation measures (current state-of-the-art)

Recap

Goal is to learn a ranking model

- $f : \mathcal{X} \rightarrow \mathcal{Y}$

That minimizes some loss function

- $\mathcal{L} : f(\mathcal{X}) \times \mathcal{Y} \rightarrow \mathcal{R}$

$\mathcal{X}$: input space

$\mathcal{Y}$: output space

$\mathcal{F}$: hypothesis space

$\mathcal{L}$: loss function

$\mathcal{O}$: optimizer

Summary

Learning to rank has been around for a few decades, but has only recently become hot

- More data, better resources, better algorithms

Machine learned ranking over many features easily beats hand-designed recommendation models

- Lots of open directions

Open directions

Deep learning

- Feature learning (vs. feature engineering)

Online learning

- Incremental, exploration-exploitation models

Structured learning

- Diversity, context-awareness

References

[Learning to rank for information retrieval]

Liu, FnTIR 2009

[Learning to rank for information retrieval]

Liu, 2011

[Learning to rank for information retrieval and natural language processing]

Li, 2014