Escuela de Verano de Inteligencia Artificial

Asociación Española para la Inteligencia Artificial (**AEPIA**)

Variables latentes para aprender a ordenar

Antonio Bahamonde.
Universidad de Oviedo. AEPIA

UNIVERSIDAD DE OVIEDO

A Coruña, Septiembre, 2014

Contents

- What is this about?
- Application: learning to order things
    - How to do this?
- Take-home messages

What is this about?

- Latent variables
  - Hidden variables (not observables) but inferred from the others
  - Reduce dimensionality
- Examples: quality of life, happiness, …

Netflix Prize Winner

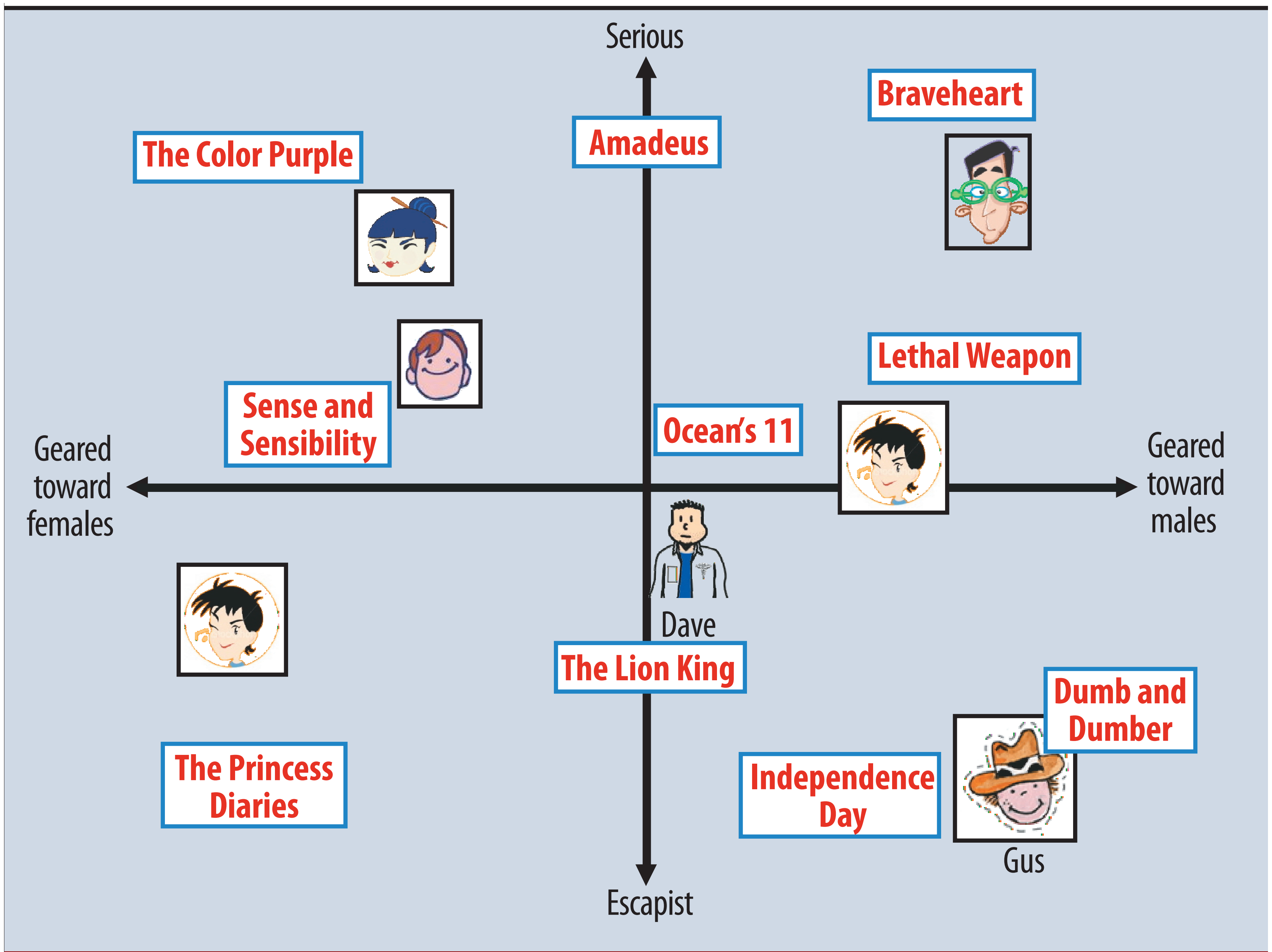


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

What is this about?

- Methods for inferring latent variables
  - Hidden Markov models
  - Factor analysis
  - PCA (Principal Component Analysis)
  - LSA (Latent Semantic Analysis)
  - Bayesian methods

What is this about?

- Latent variables
  - Embedding in Euclidean Spaces
  - Factorization
- Machine Learning tool for regression, classification or ranking

Where is it useful?

- Tasks where the interaction of two factors is essential:
    - Consumer and items
    - Queries and users
    - Images and sets of labels
    - …
- Each factor is described by a set of variables

The core formula

$$(\boldsymbol{x}^T, \boldsymbol{y}^T) = \Big((x_1, \ldots, x_{|\boldsymbol{x}|}), (y_1, \ldots, y_{|\boldsymbol{y}|})\Big)$$

$$\sum_{i,j} m_{ij} x_i y_j$$

This includes

$$\sum_{i,j} a_{ij} x_i y_j + \sum_i b_i x_i + \sum_j c_j y_j + d$$

$$(\boldsymbol{x}^T, \boldsymbol{y}^T) \leftarrow \Big((x_1, \ldots, x_{|\boldsymbol{x}|}, 1), (y_1, \ldots, y_{|\boldsymbol{y}|}, 1)\Big)$$

The core formula

Tensor product

$$(\boldsymbol{x}^T, \boldsymbol{y}^T) = \Big((x_1, \ldots, x_{|\boldsymbol{x}|}), (y_1, \ldots, y_{|\boldsymbol{y}|})\Big)$$

$$\sum_{i,j} m_{ij} x_i y_j = \langle (m_{ij} : i, j), \boldsymbol{x} \otimes \boldsymbol{y} \rangle$$

The core formula

Bilinear presentation

$$\sum_{i,j} m_{ij} x_i y_j = \boldsymbol{x}^T \boldsymbol{M} \boldsymbol{y}$$

Needs

$$|\boldsymbol{M}| = |\boldsymbol{x}| \times |\boldsymbol{y}|$$

parameters: usually too much!

The core formula

$$\sum_{i,j} m_{ij} x_i y_j = \boldsymbol{x}^T \boldsymbol{M} \boldsymbol{y} = \boldsymbol{x}^T (\boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{V}) \boldsymbol{y}$$

The set of parameters is *factorized* in two matrices

$$\boldsymbol{M} = \boldsymbol{W}^T \boldsymbol{V}$$

$$\sum_{i,j} m_{ij} x_i y_j = \boldsymbol{x}^T \boldsymbol{M} \boldsymbol{y} = \boldsymbol{x}^T (\boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{V}) \boldsymbol{y}$$

$$= (\boldsymbol{W}\boldsymbol{x})^T \boldsymbol{V}\boldsymbol{y} = \langle \boldsymbol{W}\boldsymbol{x}, \boldsymbol{V}\boldsymbol{y} \rangle$$

The core formula

$$\sum_{i,j} m_{ij} x_i y_j = \langle \boldsymbol{W}\boldsymbol{x}, \boldsymbol{V}\boldsymbol{y} \rangle$$

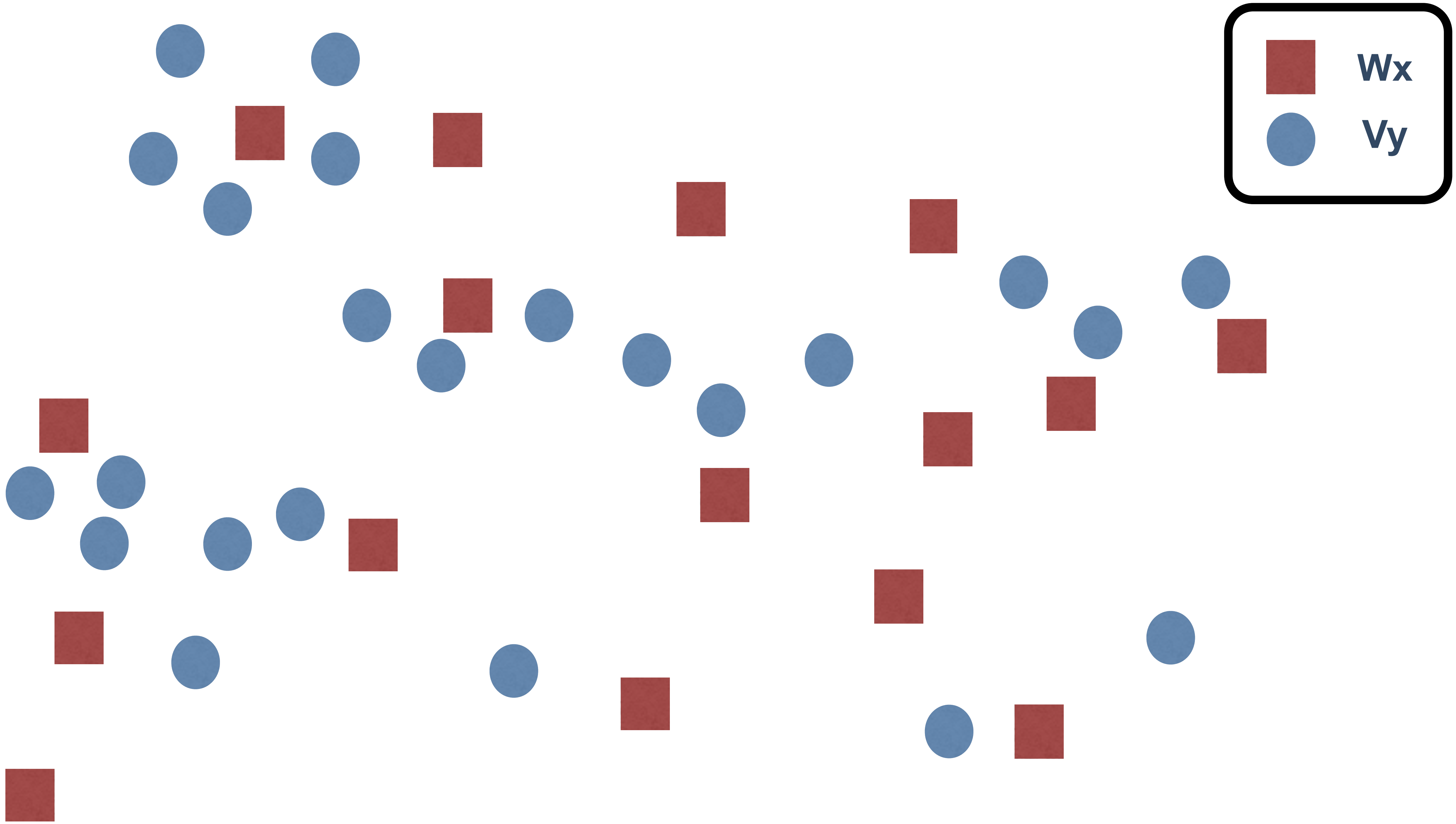
Geometric meaning: similarity of representations in a common Euclidean space

$$\mathbb{R}^{|\boldsymbol{x}|} \longrightarrow \mathbb{R}^k, \quad \boldsymbol{x} \rightsquigarrow \boldsymbol{W}\boldsymbol{x},$$

$$\mathbb{R}^{|\boldsymbol{y}|} \longrightarrow \mathbb{R}^k, \quad \boldsymbol{y} \rightsquigarrow \boldsymbol{V}\boldsymbol{y},$$

Latent variables

The core formula



The core formula

$$\sum_{i,j} m_{ij} x_i y_j = \langle \boldsymbol{W}\boldsymbol{x}, \boldsymbol{V}\boldsymbol{y} \rangle$$

Needs

$$|\boldsymbol{W}| + |\boldsymbol{V}| = |\boldsymbol{x}| \times k + |\boldsymbol{y}| \times k = \Big(|\boldsymbol{x}| + |\boldsymbol{y}| \Big) \times k$$

parameters instead of

$$|\boldsymbol{M}| = |\boldsymbol{x}| \times |\boldsymbol{y}|$$

Warning: it is not the same solution. But it is enough good. Better, if we have to learn it!

The core formula

$$\sum_{i,j} m_{ij} x_i y_j = \langle \boldsymbol{W}\boldsymbol{x}, \boldsymbol{V}\boldsymbol{y} \rangle$$

Columns of ***W*** and ***V*** are *latent* variables in the sense used in Information Retrieval (LSI) or Statistics (PCA)

Therefore, factorization

- Useful when variables
  - can be split in two parts
  - interaction is relevant to make predictions
- Needs less parameters to learn
- Has a geometric semantics
- Learns latent variables that filter noisy information

Contents

- What is this about?
- Application: learning to order things
- Take-home messages

Ordering is important

- Which document is the most relevant for this query?
- Which movies are likely to be enjoyed by a user?
- Which kind of (...food product...) is going to be preferred by consumers?
- Which assignment deserves a higher grade?
- Which diet is better for me?

Applications

- Recommender Systems:
    - Netflix Prize
    - News recommendations
- Information Retrieval
    - Music annotations
    - Image tagging
- Analysis of consumer preferences of food products
- Assessment in MOOCs

RS: Definitions

RS are software agents that elicit the interests and preferences of individual consumers and make recommendations accordingly.

RS help to match users with items

- Ease information overload
- *Sales* assistant (guidance, advisory, persuasion,...)

Different system designs / paradigms based on availability of exploitable data

- Implicit or explicit user feedback
- Domain characteristics

Popular RS

- Google
- Genius (Apple)
- last.fm
- Amazon
- Netflix
- TiVo

Collaborative Filtering

- To relate users and items
  - explicit feedback (ratings)
  - implicit (purchase or browsing history, search patterns, ...)
  - sometimes items descriptions by feature (content based)
- Approaches:
  - neighborhood
  - latent factor

Naïve Neighborhood Approach

item-item user-user

	item1	item2	item3	item4	item5
alice	5	3	4	4	?
user1	3	1	2	3	3
user2	4	3	4	3	5
user3	3	3	1	5	4
user4	1	5	5	2	1

Compute similarity → prediction

Naïve Neighborhood Approach

user-user

	item1	item2	item3	item4	item5
alice	5	3	4	4	?
user1	3	1	2	3	3
user2	4	3	4	3	5
user3	3	3	1	5	4
user4	1	5	5	2	1

Naïve Neighborhood Approach

item-item

	item1	item2	item3	item4	item5
alice	5	3	4	4	?
user1	3	1	2	3	3
user2	4	3	4	3	5
user3	3	3	1	5	4
user4	1	5	5	2	1

Naïve Neighborhood Approach

- not all neighbors should be taken into account (similarity thresholds)
- not all items are rated (co-rated)
- not involved the loss function

The New York Times

Netflix Prize

(Sep, 21, 2009):

Netflix Awards $1 Million Prize and Starts a New Contest

[...]try to predict what movies particular customers would prefer

"Predicting the movies Netflix members will love is a key component of our service," said Neil Hunt, chief product officer (Netflix)

Netflix Prize

The Netflix dataset

More than 100 million movie ratings (1-5 stars)

Nov 11, 1999 and Dec 31, 2005

- about 480,189 users and n = 17,770 movies
- 99% of possible ratings are **missing**
  - movie average 5,600 ratings
  - user rates average 208 movies

Training and quiz (test-prize) data

Netflix Prize

The loss function: root mean squared error (RMSE)

$$RMSE = \sqrt{\frac{1}{|Quiz|} \sum_{(u,i) \in Quiz} (r(u,i) - b(u,i))^2}$$

Netflix had its own system, Cinematch, which achieved 0.9514.

The prize winner had to reach RMSE below 0.8563 (10% improvement)

Netflix Prize Winner

For example, suppose that you want a first-order estimate for user Joe's rating of the movie *Titanic*. Now, say that the average rating over all movies, μ, is 3.7 stars. Furthermore, *Titanic* is better than an average movie, so it tends to be rated 0.5 stars above the average. On the other hand, Joe is a critical user, who tends to rate 0.3 stars lower than the average. Thus, the estimate for *Titanic*'s rating by Joe would be 3.9 stars (3.7 + 0.5 - 0.3).

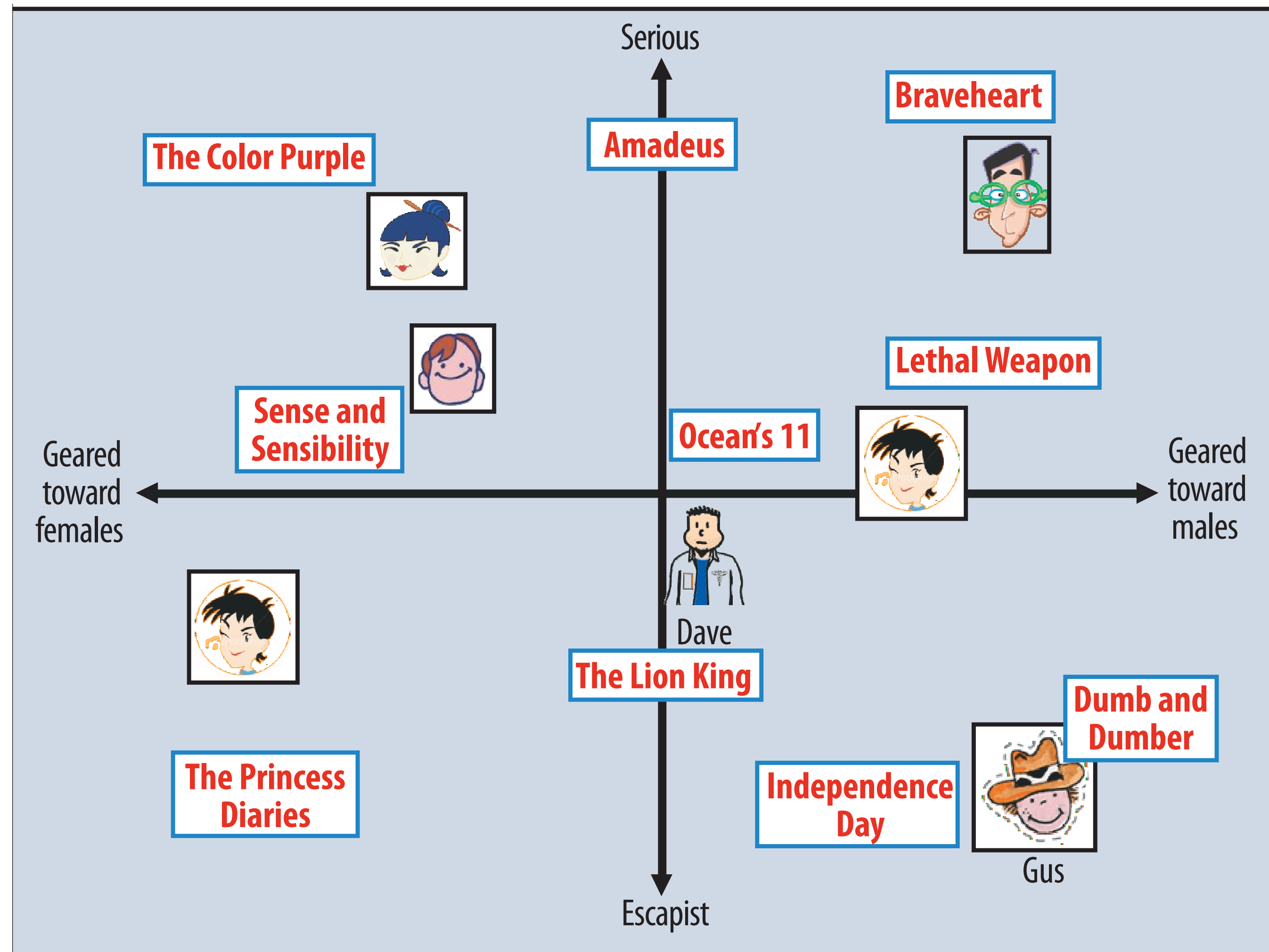


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

$$f(u, i) = \mu + \boldsymbol{bU}_u + \boldsymbol{bI}_i + \boldsymbol{P}_u\boldsymbol{Q}_i$$

Netflix Prize Winner

Koren and Bell, set an optimization problem that admits an efficient solution and avoids the problem of missing values

$$\min_{b_*,q_*,p_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_i - b_u - q_i^T p_u)^2 + \lambda_4 (b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2)$$

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

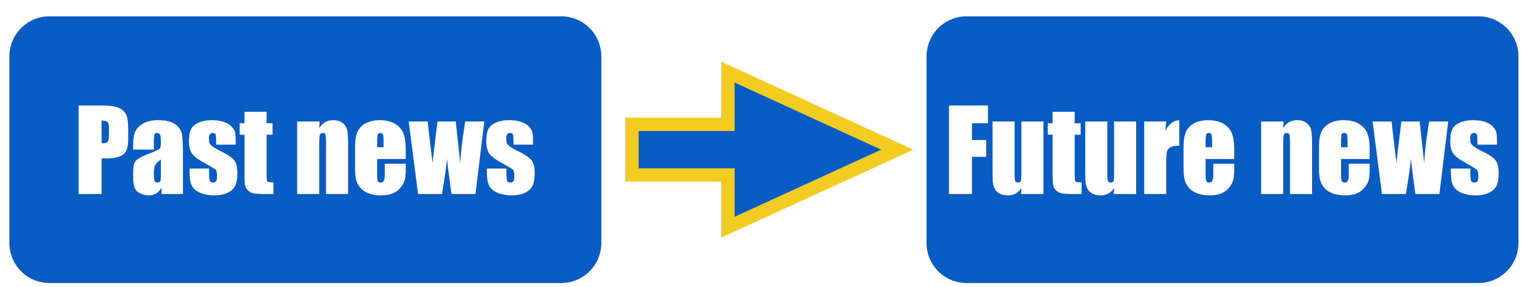
q_i and p_u are vectors of k components

News Recommendations

- The aim is to keep readers online with personalized recommendations to read next
- There are already a number of implicit or explicit recommendations in digital newspapers
- A news recommender should suggest news of interest for readers that are not explicitly linked by other recommenders

News Recommendations

- Learning task: find a function to map from trajectories of already read news to news to be read in the future. It is **multilabel classification task**



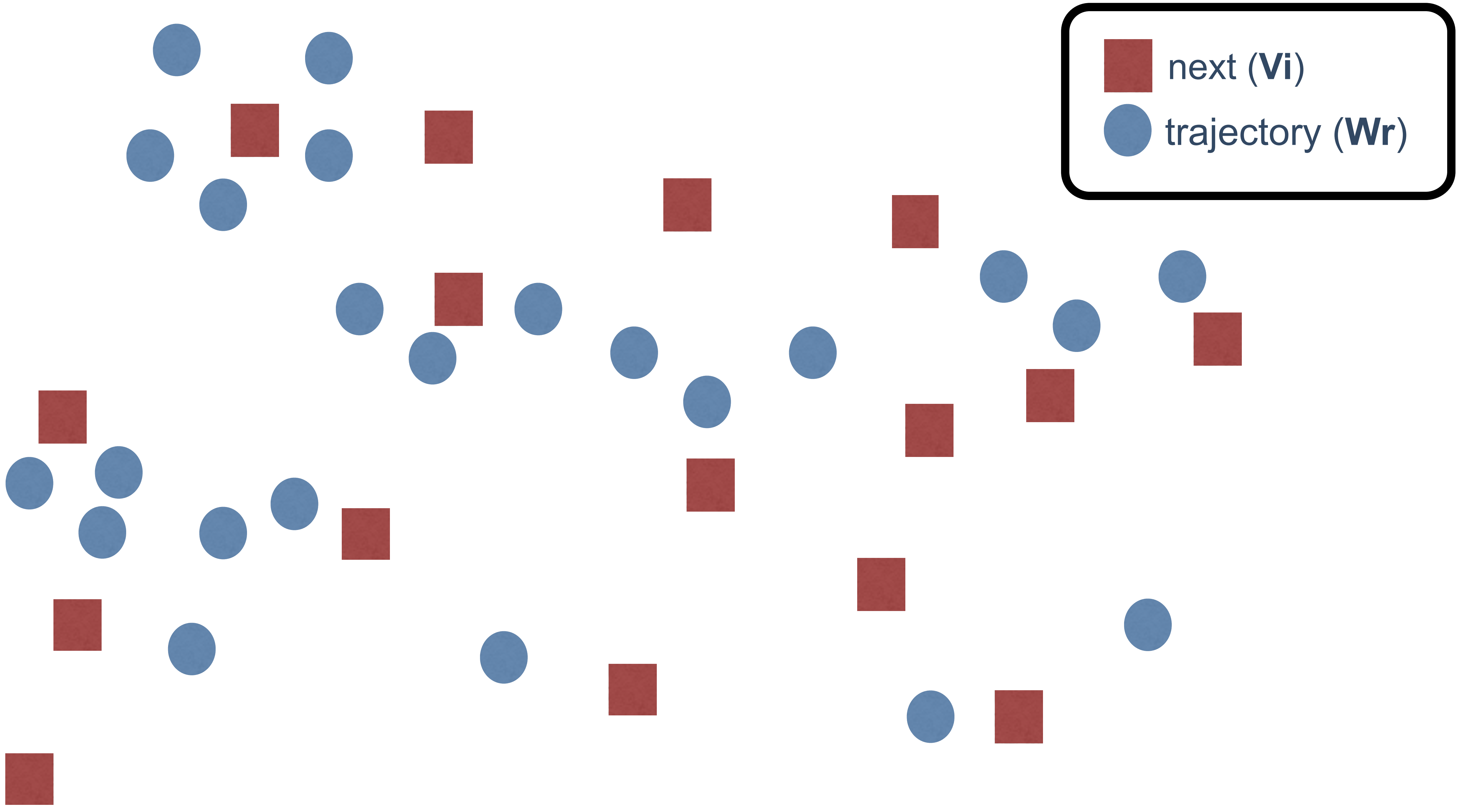
- Both sets of news are going to be embedded in a common Euclidean space

News Recommendations

- Learning task
  - represent reading trajectories
  - represent news
  - in such a way that interesting news for readers are near to their trajectories

$$
\begin{aligned}
f(\boldsymbol{r}, i) &= -||\phi_{trajectory}(\boldsymbol{r}) - \phi_{art}(i)||^2 \\
&= 2(\boldsymbol{W}\boldsymbol{r})^T \boldsymbol{V}_i - (\boldsymbol{W}\boldsymbol{r})^T (\boldsymbol{W}\boldsymbol{r}) - (\boldsymbol{V}_i)^T \boldsymbol{V}_i
\end{aligned}
$$

News Recommendations



News Recommendations

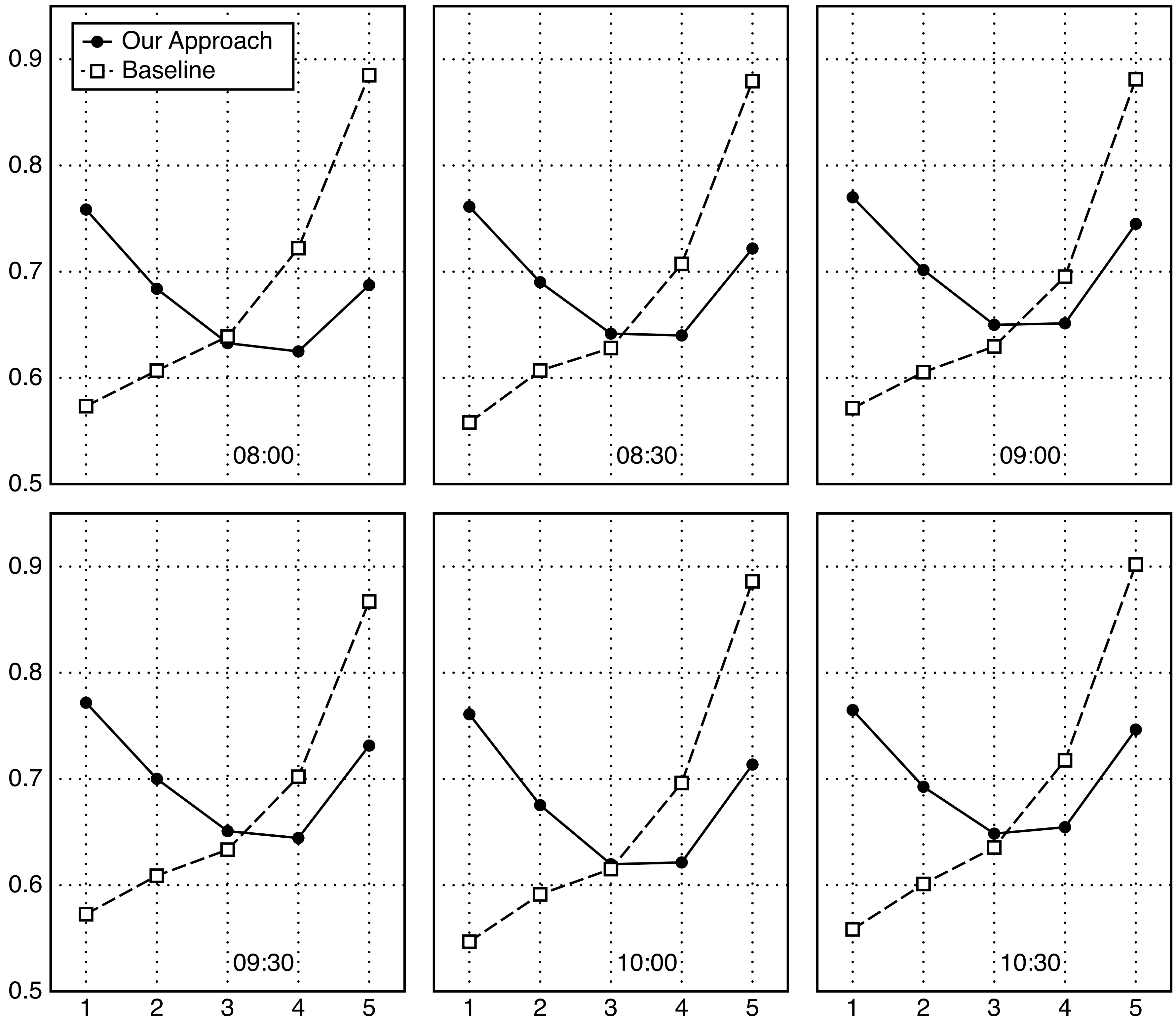
Optimize ranking loss:

WARP (Weighted Approximately Ranked Pairwise)

$$WARP_{error} = \sum_j L(\alpha) \max(0, 1 - f(\boldsymbol{r}_j, p) + f(\boldsymbol{r}_j, n))$$

where *p* is a positive example and *n* a negative one

News Recommendations



MOOCs assessment

- MOOCs are expensive:
  - $ 50,000 filming
  - $ 50,000 hosting
  - Intelligent services (answering questions, evaluation of assignments)
- Prestigious universities are interested because:
  - Open: Kind of ad to attract students for regular courses
  - Licensing courses: for Universities without specialist in high level courses. These universities provide TA (cheaper than Professors that are not really available)

MOOCs: peer evaluation

- Assignments are difficult to be evaluated by computers. Open-responses. Essay. Graphical illustrations. Pictures.
- Metaphor of Conference papers
    - Students submit assignments as papers.
    - Students will serve as reviewers
- Students receive a *rubric* (a set of rules to uniform grades)

MOOCs: peer evaluation

- Someone or a simple software will assign assignments (papers) to other students (reviewers).

- Students will be advised that they are going to be evaluated as authors and as reviewers.

MOOCs: peer evaluation

Each grader g receives a subset of assignments and provides a grade

$$g(i) \in [0, 10]$$
$$\forall g \in \mathcal{G}, g(i) > g(j) \Rightarrow [g, i, j] \in \mathcal{PJ}$$

Embedding of graders and assignments

$$\phi_{gr}(g) : \mathcal{G} \to \mathbb{R}^k, \quad g \mapsto \boldsymbol{W}_g,$$
$$\phi_a(i) : \mathcal{A} \to \mathbb{R}^k, \quad i \mapsto \boldsymbol{V}_i.$$

MOOCs: peer evaluation

The ranking will be given by the average of learned grades

$$f(\mathcal{G}, i) = -\frac{1}{|\mathcal{G}|} \left\| \sum_{g \in \mathcal{G}} \phi_{gr}(g) - \phi_a(i) \right\|^2$$

This rank should be as coherent as possible with the ranks of graders

MOOCs: peer evaluation

Define error in order to maximize the margin

$$err(\boldsymbol{W}, \boldsymbol{V}) = \sum_{[g,i,j] \in \mathcal{PJ}} \max\Big(0, 1 - f(\mathcal{G}, i) + f(\mathcal{G}, j)\Big)$$

regularization

$$r(\boldsymbol{W}, \boldsymbol{V}) = \|\boldsymbol{W}\|_F^2 + \|\boldsymbol{V}\|_F^2$$

Then we need to optimize

$$\operatorname*{argmin}_{\boldsymbol{W}, \boldsymbol{V}} \ (err(\boldsymbol{W}, \boldsymbol{V}) + \nu r(\boldsymbol{W}, \boldsymbol{V}))$$

MOOCs: peer evaluation

	a1	a2	a3	a4	a5
g1		6	8	4	
g2	10		9		10
g3		4	6	3	
g4	8	5		5	8

MOOCs: peer evaluation

	a1	a2	a3	a4	a5
g1	★	6	8	4	★
g2	10	★	9	★	10
g3	★	4	6	3	★
g4	8	5	★	5	8

MOOCs: peer evaluation

	a1	a2	a3	a4	a5
g1	★	★	★	★	★
g2	★	★	★	★	★
g3	★	★	★	★	★
g4	★	★	★	★	★

MOOCs: peer evaluation

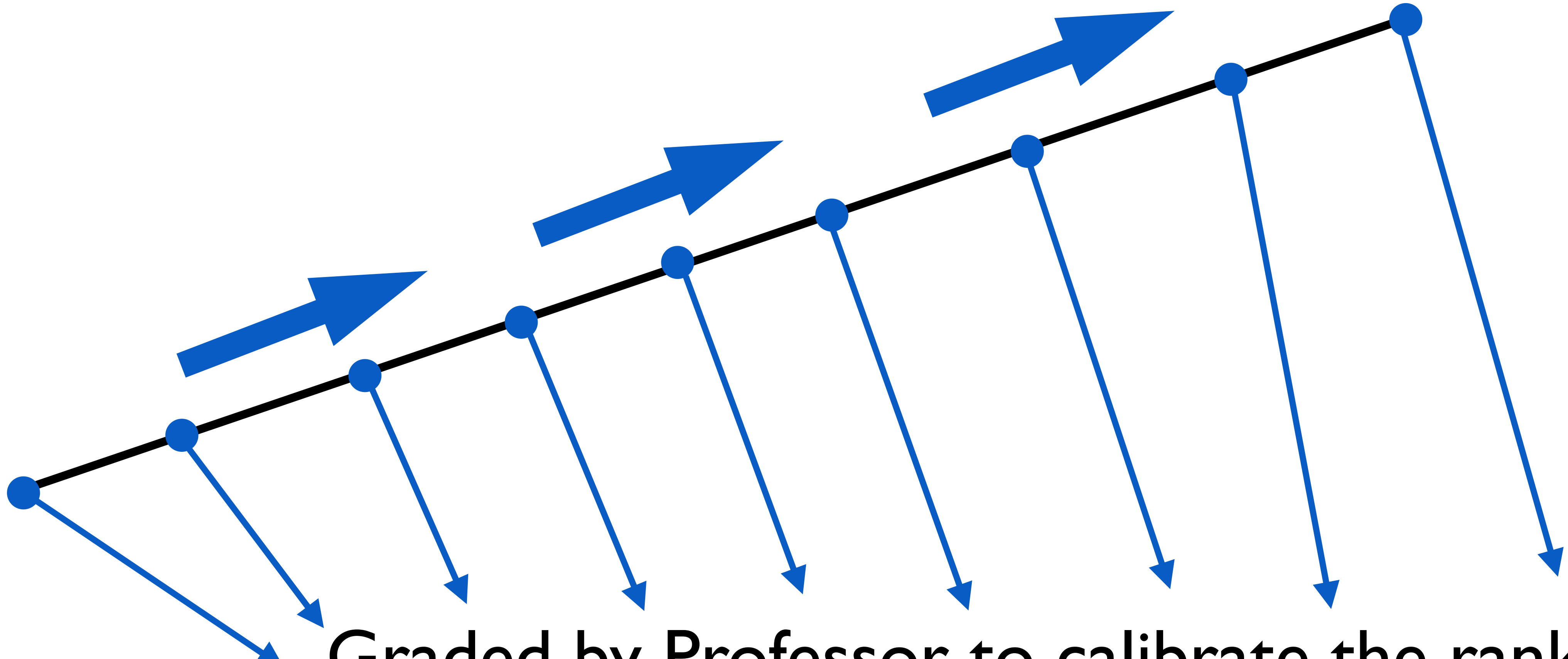
	a1	a2	a3	a4	a5
g1	★	★	★	★	★
g2	★	★	★	★	★
g3	★	★	★	★	★
g4	★	★	★	★	★
$f(\mathcal{G}, i)$	★	★	★	★	★

MOOCs: peer evaluation

- The output of the learning process is a ranking of the assignments
- Calibration by Professor
    - Some of the assignments will be evaluated by the Professor to find a way to convert ranking into assessments

MOOCs: peer evaluation

Order assignments by $f(\mathcal{G}, i)$



MOOCs: peer evaluation

The final grade of a student: weighted sum of evaluation as author (calibrated percentile) and as reviewer (AUC)

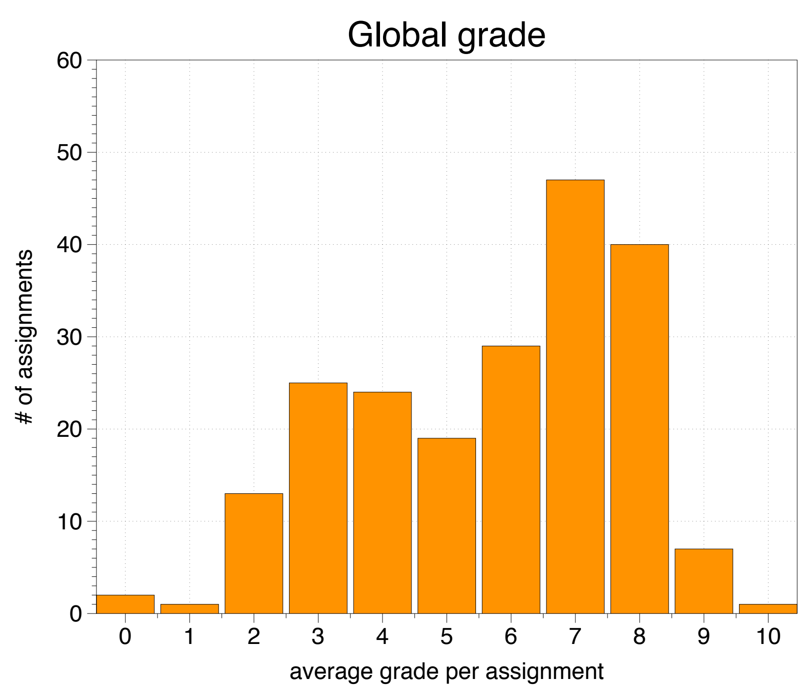
$$grade(i) = 0.7 \cdot \mathrm{calibrated}(f(\mathcal{G}, i)) + 0.3 \cdot AUC(f, g_i)$$

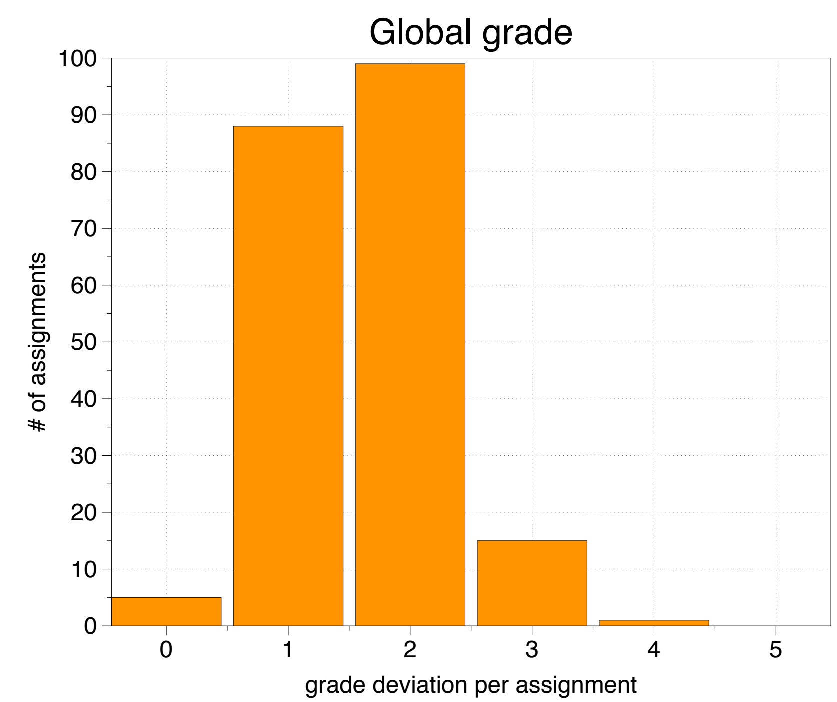
MOOCs: peer evaluation

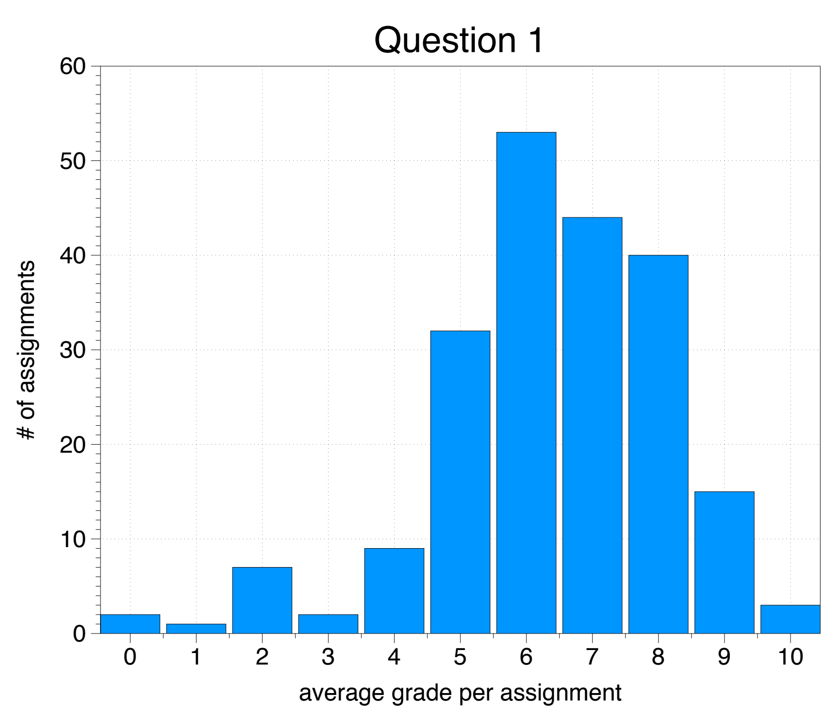
Table 1: Datasets description

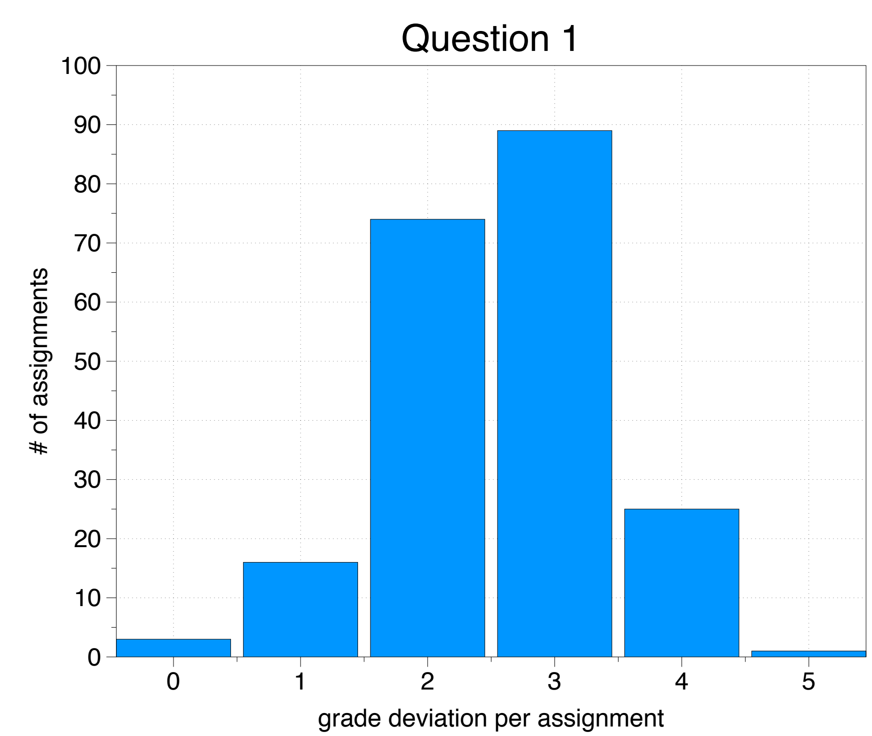
# of assignment	208
# of graders	188
# of evaluations	1882
evaluations per grader	10.01 ± 0.77
evaluations per assignment	9.05 ± 1.71

MOOCs: peer evaluation

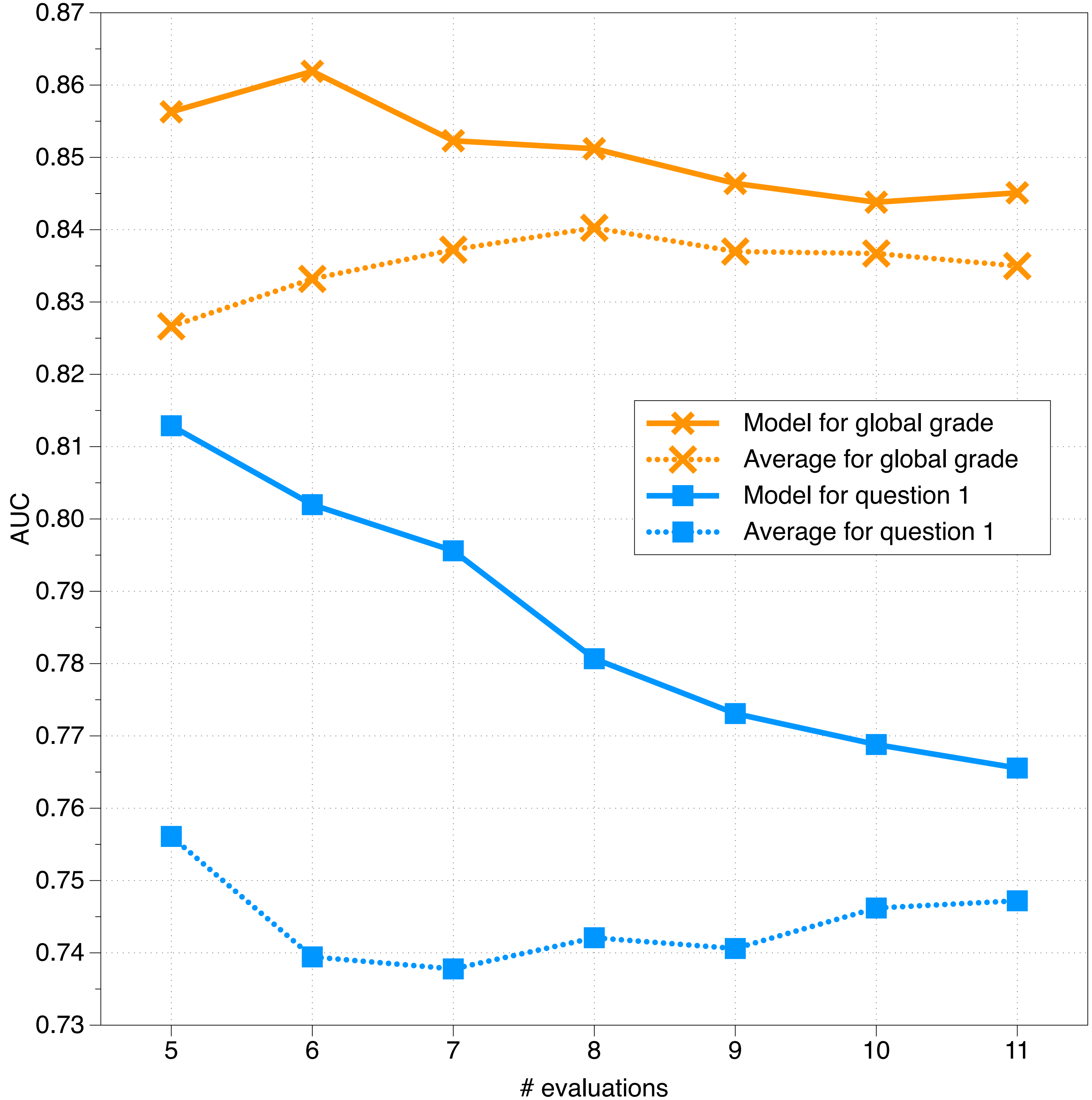








MOOCs: peer evaluation



Take-home messages

- Learning task
    - Split variables. Interaction.
    - Classification, regression, ranking

Take-home messages

- Procedure
  - Set embedding equations
  - Find optimal matrices for loss function and regularization
  - Use your favorite optimizer (SGD, proximal)

Take-home messages

- Advantages
  - Clean, principled
  - Noise-tolerant
  - Fast. Scalable to Big Data

Take-home messages

- Bibliography

  - Neal Parikh (Department of Computer Science Stanford University),Stephen Boyd (Department of Electrical Engineering Stanford University): Proximal Algorithms. Foundations and Trendsin Optimization Vol. 1, No. 3 (2013) 123–231.
  - Kaare Brandt Petersen, Michael Syskind Pedersen: The Matrix Cookbook. Technical University of Denmark, 2012

Escuela de Verano de
Inteligencia Artificial

Asociación Española para la Inteligencia Artificial (**AEPIA**)

Variables latentes para aprender a ordenar

Antonio Bahamonde.
Universidad de Oviedo. AEPIA

UNIVERSIDAD DE OVIEDO

A Coruña, Septiembre, 2014