A Fast Learning Recommender Estimating Preferred Ranges of Features

Takuya Watanabe
Yuji Nakazato — Edirium K.K.
Hiroaki Muroi
Takuya Hashimoto — Acroquest Technology Co., Ltd.
Toru Shimogaki — NTT DATA Corporation
Takeshi Nakano — Recruit Technologies Co., Ltd.
Tsutomu Kumazawa — Software Research Associates, Inc.

12th Joint Conference on Knowledge-Based Software Engineering (JCKBSE 2018)

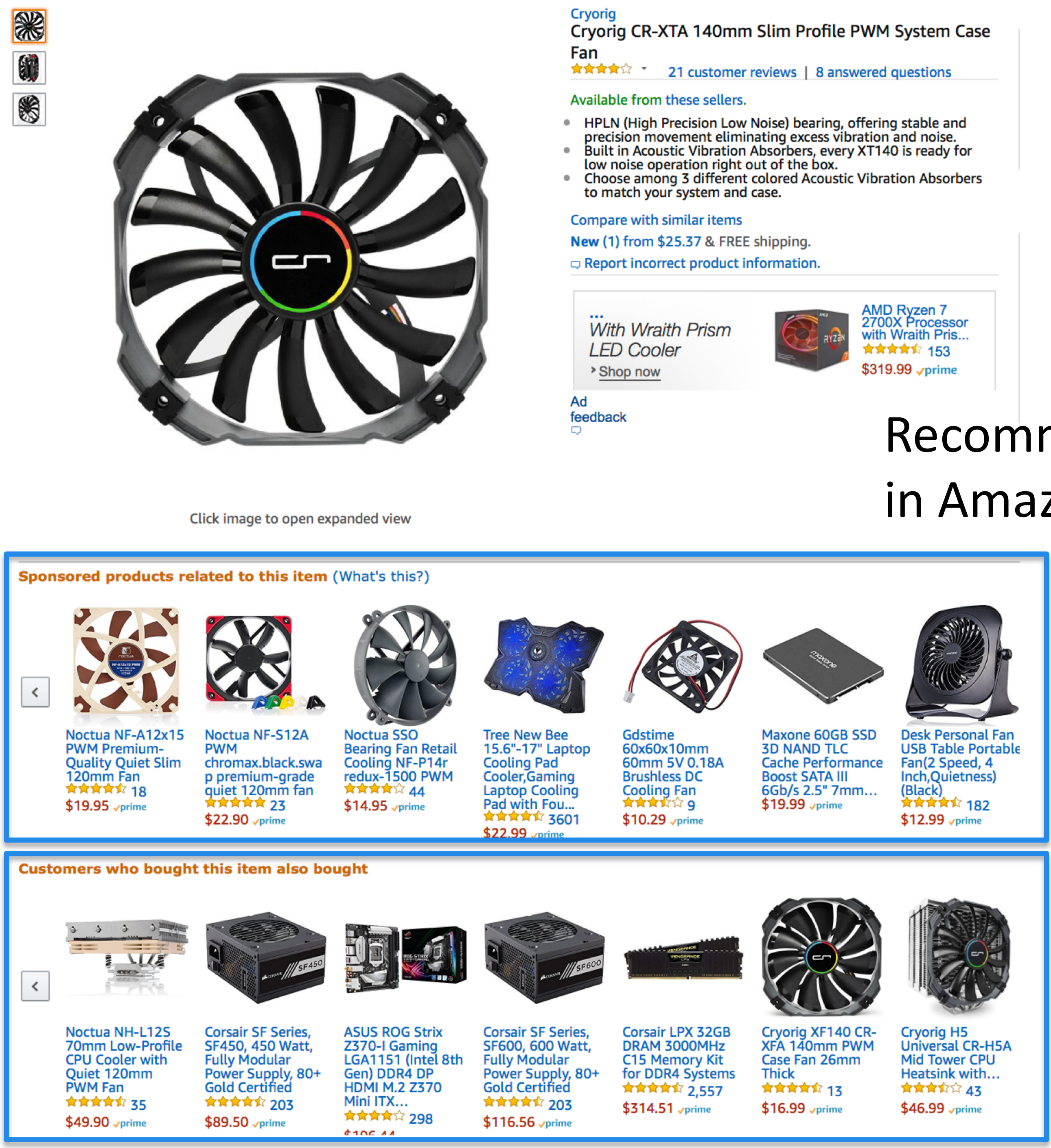
Outline

- recommender basics
- motivation
- basic ideas
- algorithm details
- experimental evaluation
- deployment as a web service
- summary

What Are Recommenders?

- to recommend items for users
  - inferring items which might be preferable for targeting users
- two types of recommenders
  - content-based recommender
    - based on item features: descriptions, images, prices etc.
  - collaborative filtering
    - based on records of user behaviour
    - e.g. users who bought this item also bought...
- general procedure
  - find a target point in a vector space
  - calculate similarity/proximity of items
    - extensive use of *k*-nearest neighbour search
  - create item ranking
    - can be a raw similarity ordered list
    - various re-ranking methods

Recommendation Example in Amazon.com



Content-based?

Collaborative filtering

Motivation

- making a recommender
  - equipped with web service like interactive user interface
    - to help users recognise their own preferences
  - whose users are mostly new or not registered
- want to estimate new users' preferences
  - *cold-start* problem
    - no user attributes
    - no behavioural history -> collaborative filtering cannot be applied
    - need to know users' preferences to match item features
  - estimate feature-wise?
    - needs high degree of user effort
    - what about interaction between features?
- objectives
  - infer users' preferable ranges of item features
  - reduce number of user interactions as far as possible

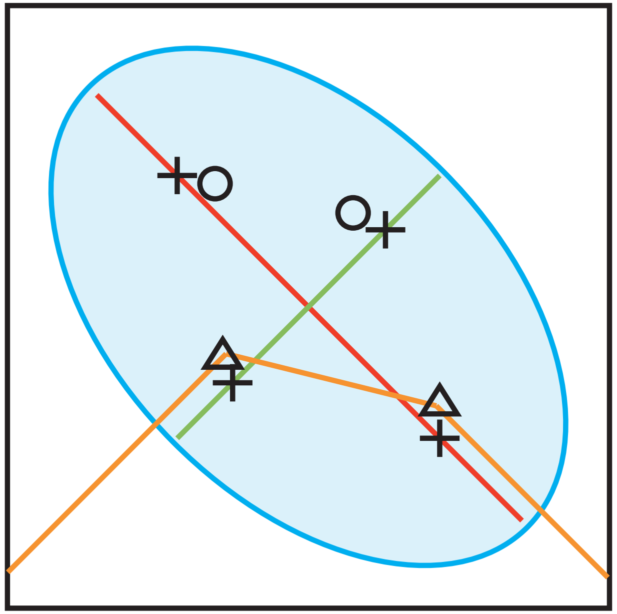
Basic Ideas

- binary search like active learning
  - aggressively cut the search space out
  - change the search area according to user responses
- dimensionality reduction of the item space
  - to make binary search applicable
  - by multidimensional scaling (MDS)

Binary Search Like Active Learning

- binary search (actually not exactly)
  - estimate a preferable range within [0, 1]
    - present a pair (0.25, 0.75) to the user
    - discard (0.5, 1] if she prefers 0.25, [0, 0.5) otherwise
    - next iteration starts with the remaining range
- can be two-dimensional
  - present a set of pairs along two axes
  - need to determine two axes
    - want to choose the most "effective" axes
    - choose axes spanning the most widely distributed directions
    - by principal component analysis (PCA)
  - discard an area around not preferable ones
  - next iteration starts with the remaining area

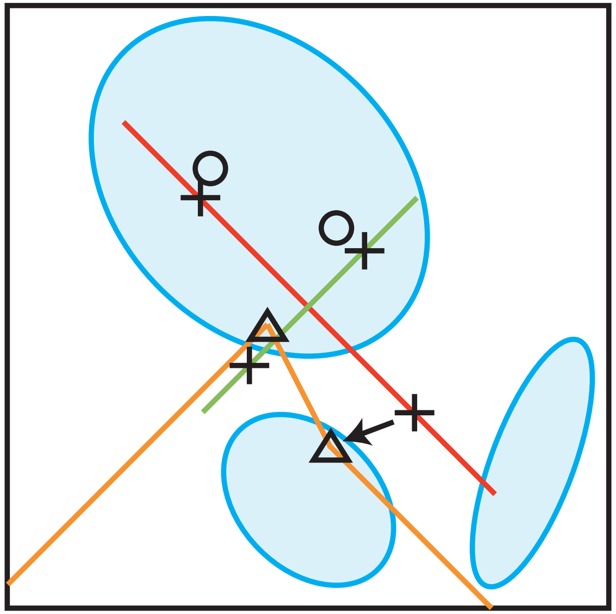
Geometrical Exclusion



pivot points are on-axis reference points, and items to be presented to the user are nearest items to the pivot points

area to be discarded is determined by line segments through not preferable items

Convex case



- cyan circle: item distribution
- red and green line: axis
- cross: pivot point
- black circle: preferable item
- triangle: not preferable item
- orange line: to segment the area to be discarded

Concave case

Dimensionality Reduction

- how about more than two dimensions?
  - possible, but impractical; needs 2^{d-1} pairs
- reduce dimensionality without loss of information as far as possible
  - by multidimensional scaling (MDS)
  - using classical MDS, which is effectively the same as linear PCA
    - non-linear MDS can be used, but *regularisation* not applicable (next page)
- what is MDS?
  - calculate a low-dimensional representation from a distance matrix
  - preserving between-item distances as well as possible

Regularisation of the Learning Process

- a problem caused by dimensionality reduction
  - searching is done in a reduced space, pick items to present a user
  - can infer features in the original item space by linear regression
  - but we want to present a user actual item features
  - what if magnitude relations of a feature inferred by linear regression and original items' are different?
- a simple solution
  - select a pair of items near the pivot point
  - check if magnitude relations of features are *coincide*
    - coincide means relations (which is larger, in this case) are the same
  - if not, pick next pair

	Item A	Rel	Item B
inferred	0.25	<	0.75
actual	0.21	<	0.83

coincide!

	Item A	Rel	Item B
inferred	0.25	<	0.75
actual	0.83	>	0.21

not coincide

Algorithm Details

Initialise search space S
repeat
  Fill S by colour k_1
  Analyse S by principal component analysis
  $a_{1,2} \leftarrow$ axes corresponding to two eigenvalues

determine axes

  for $i \leftarrow 1, 2$ **do**
    Select pivot points $p_{1,2}$ from axis a_i
    repeat
      Select candidates $c_{1,2}$ around $p_{1,2}$ respectively
    until *coincide?*(S, $c_{1,2}$)

choose appropriate items

    Display candidates $c_{1,2}$ and wait response
    $r_i^p \leftarrow$ candidate chosen, $r_i^n \leftarrow$ candidate not chosen

get a user response

  end for
  Place line segments $l_{1,2,3}$ based on the position of $r_{1,2}^n$
  Draw $l_{1,2,3}$ on S by colour k_3

place line segments

  Select an arbitrary tile t not belonging to $l_{1,2,3}$
  Flood-fill S from t by colour k_2
  $k \leftarrow$ colours of $r_{1,2}^p$ and the mean
  $u \leftarrow$ subscript of the majority of colours in k
  Exclude k_{3-u} and k_3 coloured tiles from S

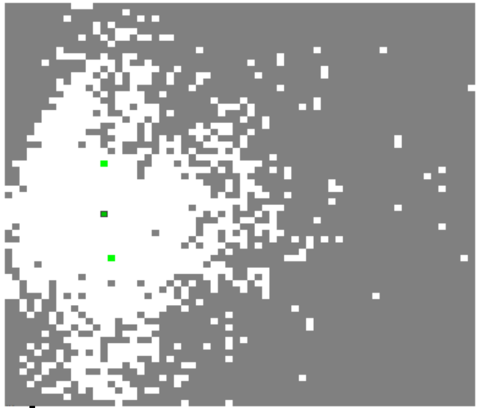
discard a non-preferable area

until *converged?*(S)

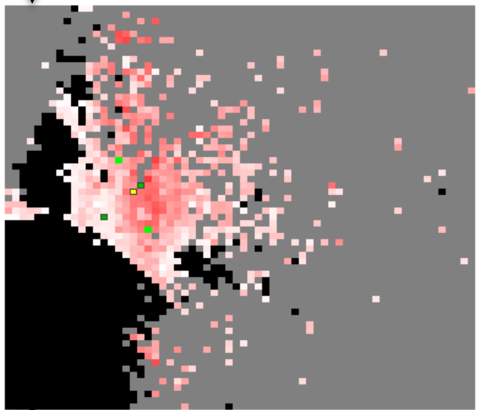
An Actual Progress of Geometrical Exclusion

- white tile: at least one item exists
- gray tile: no item exists
- black tile: excluded
- green tile: pivot point
- red tile: received positive responses
- blue tile: received negative responses

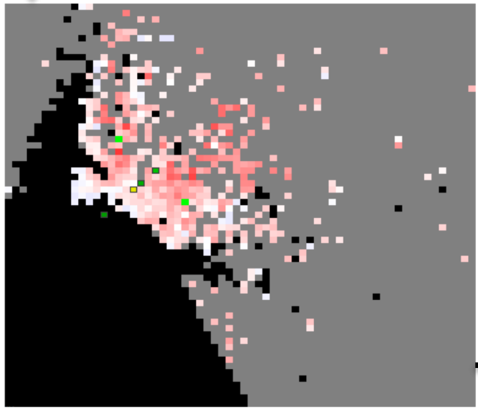
positive/negative judgement is by using a linear classifier (not covered by the paper)



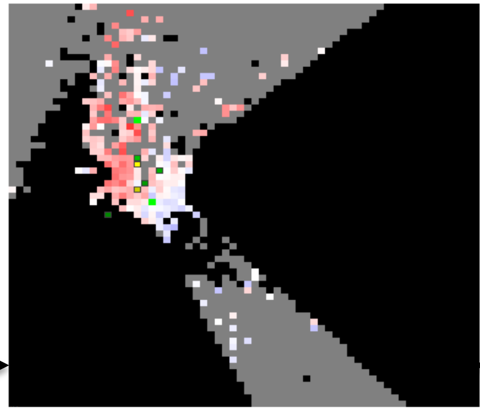
Initial state



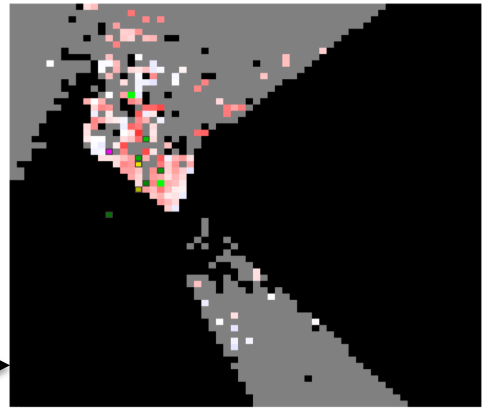
After 2 responses



After 4 responses



After 6 responses

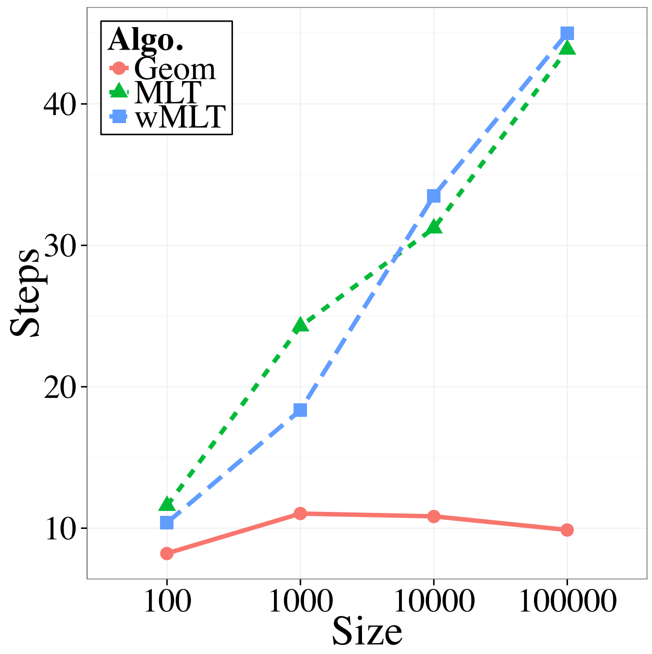


After 8 responses

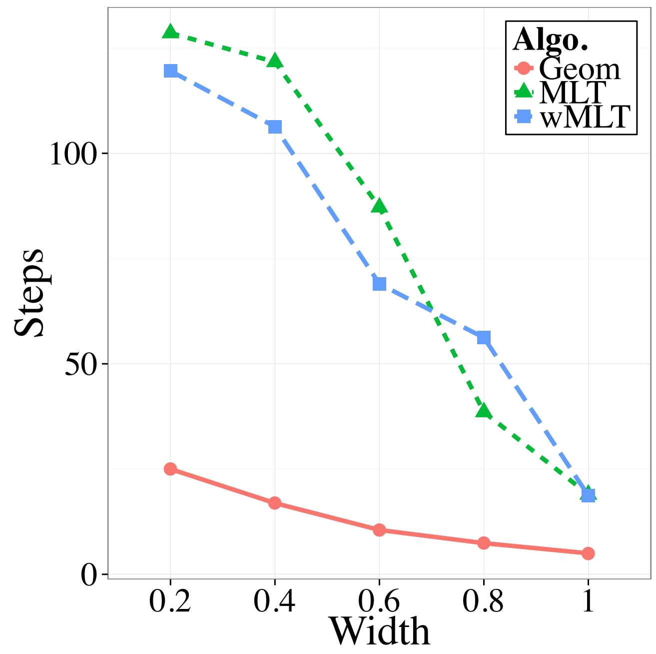
Experimental Evaluation

- compared algorithms
  - more like this (MLT) by L. McGinty and B. Smyth
    - has an internal feature-wise query
    - next query is created by incorporating all features of a selected item
    - baseline
  - a variant, weighted more like this (wMLT)
    - each feature has its weight
    - weight is a ratio of unique feature values
    - performed best in their experiment
- simulated experiments
  - randomly select a target item (user's most preferable item) at first
    - its surrounding region is an "answer" region
  - a "user" knows which item is closer to the target item and will select it
  - when reached the "answer" region, simulation stops

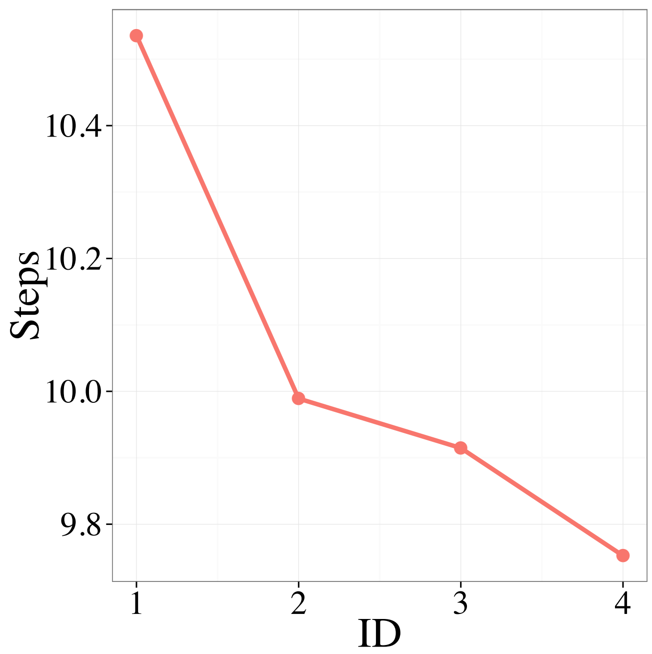
L. McGinty and B. Smyth, “Comparison-based recommendation”, ECCBR ’02



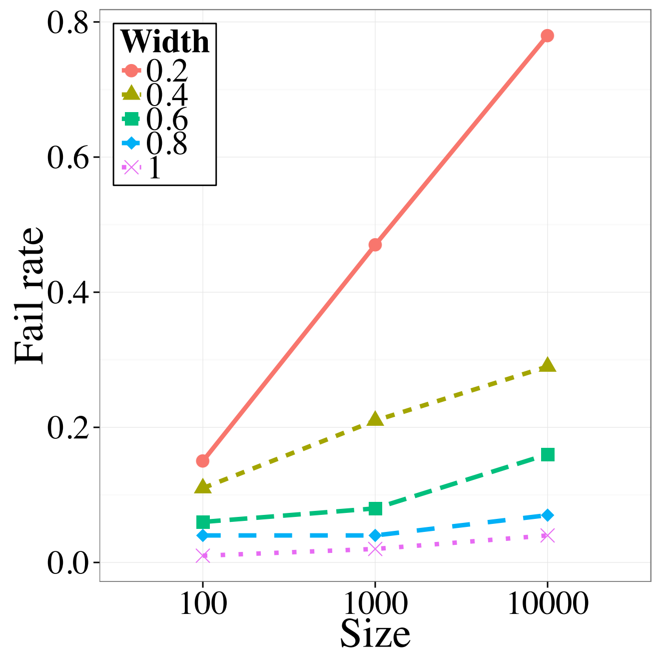
Varying data sizes



Varying "answer" region width (stddev)



Varying correlations between features



Fail rate analysis

Deployment as a Web Service

- implemented an online recommender system
  - for rental residence recommendation
  - UI is designed for smartphones
- deployed five weeks
  - within a commercial web service
  - received favourable responses
- features
  - monthly rent
  - floor area
  - age
  - distance from the nearby station

Summary

- proposed an algorithm to infer preferred ranges of features
  - fast learning inspired by binary search
- showed its effectiveness by simulated experiments
- implemented the algorithm as an online recommender
  - as a part of a commercial web service