Web Mining and Recommender Systems

Personalized Models of Visual Data

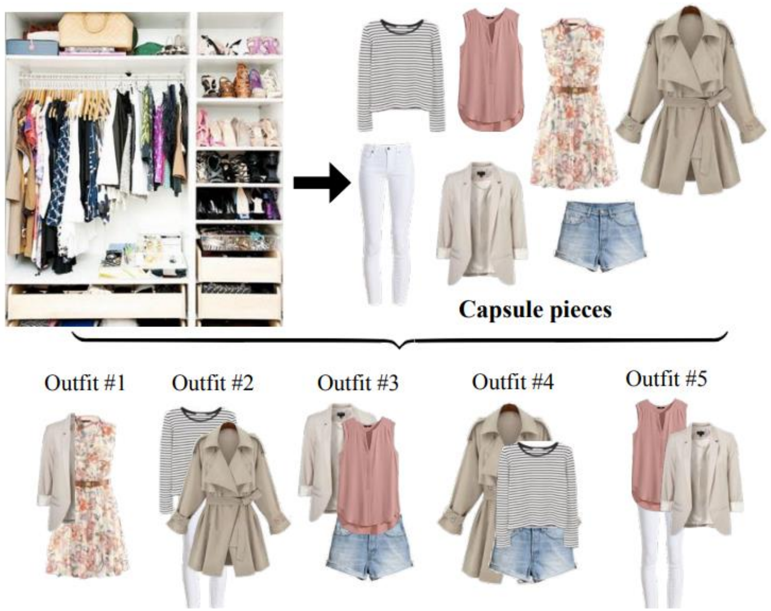
Learning Goals

- How can we do recommendation in "visual" settings (mostly fashion)
- Why is visual data different from any other sort of feature or side-information?
- What are some different tasks for visual/fashion recommendation?

Visual recommendation

What are some tasks in visual recommendation?

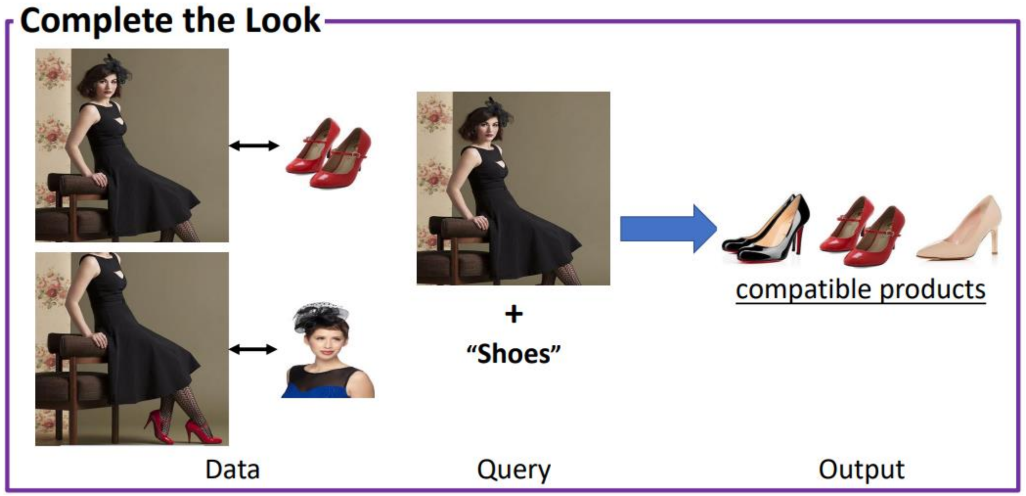
- Recommending sets of compatible items



Visual recommendation

What are some tasks in visual recommendation?

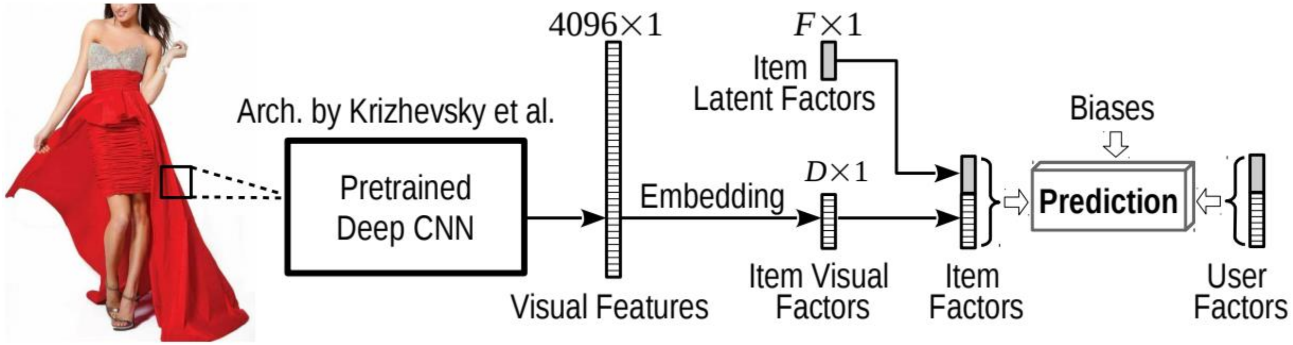
- Identifying recommendations based on a user image



Visual recommendation

What are some tasks in visual recommendation?

- Regular old recommendation with visual data



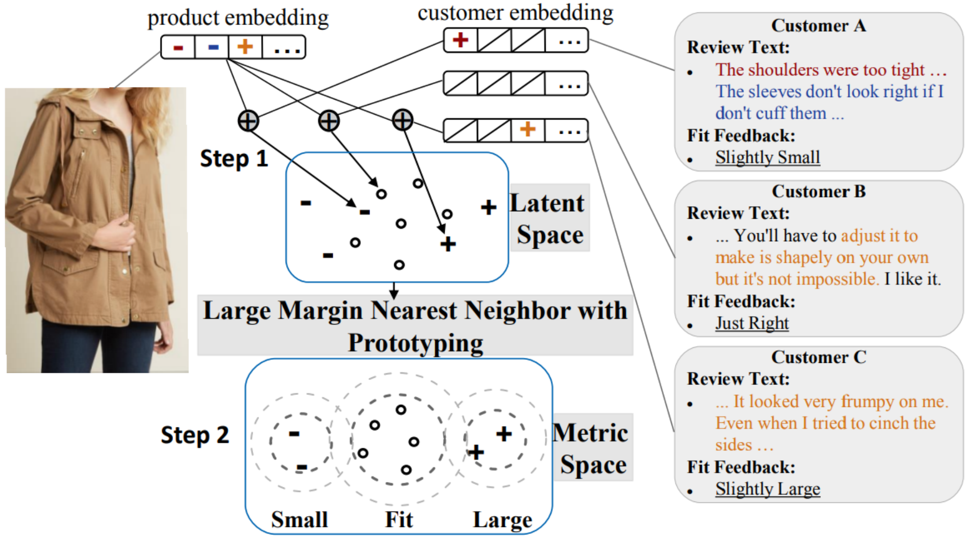
Visual recommendation

What are some tasks in visual recommendation?

- Other related tasks, e.g. fit prediction

Jacket in Maplewood (Size: Medium)

https://www.modcloth.com/shop/outerwear/woods-you-be-mine-jacket-in-maplewood/118317.html



Customer A

Review Text:

- The shoulders were too tight … The sleeves don't look right if I don't cuff them ...

Fit Feedback:

- Slightly Small

Customer B

Review Text:

- ... You'll have to adjust it to make is shapely on your own but it's not impossible. I like it.

Fit Feedback:

- Just Right

Customer C

Review Text:

- ... It looked very frumpy on me. Even when I tried to cinch the sides …

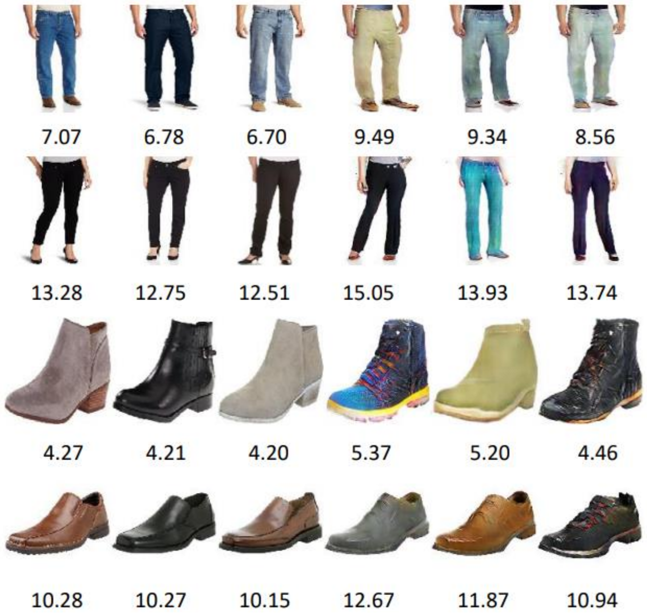
Fit Feedback:

- Slightly Large

Visual recommendation

What are some tasks in visual recommendation?

- Personalized design



Visual recommendation

Why are these tasks any different from other recommendation tasks?

- Cold-start issues: sparse, long-tailed datasets
- Visual data is high-dimensional, can't easily be "plugged in" to feature-based frameworks
- Complex combination of temporal dynamics etc. in fashion
- Tasks with unusual semantics (e.g. fit prediction) for which standard models don't work

Web Mining and Recommender Systems

Estimating compatibility among items

Visual compatibility estimation

Goal is to implement a feature like "people who bought X also bought Y". What makes this different from "traditional" recommendation problems?

- How can we establish "groundtruth"? Previously when implementing "people who bought X also bought Y"-style features we used co-purchases, but do co-purchases really indicate visual compatibility?
- Cold-start problems (may be even worse in fashion scenarios)
- Features that make items compatible could be incredibly subtle (and are some combination of visual and non-visual features)
- Compatibility is quite different from visual similarity

Initial attempts…

- Collect a bunch of co-purchase links from Amazon
- Collect product images of each product
- Estimate a compatibility function of the form:

$$p(i \text{ co-purchased with } j) = \sigma(c - d(i, j)).$$

Ref.: Image-based recommendations on styles and substitutes

Initial attempts…

- This is accomplished with a simple projection matrix:

Ref.: Image-based recommendations on styles and substitutes

Initial attempts…

- This is accomplished with a simple projection matrix:

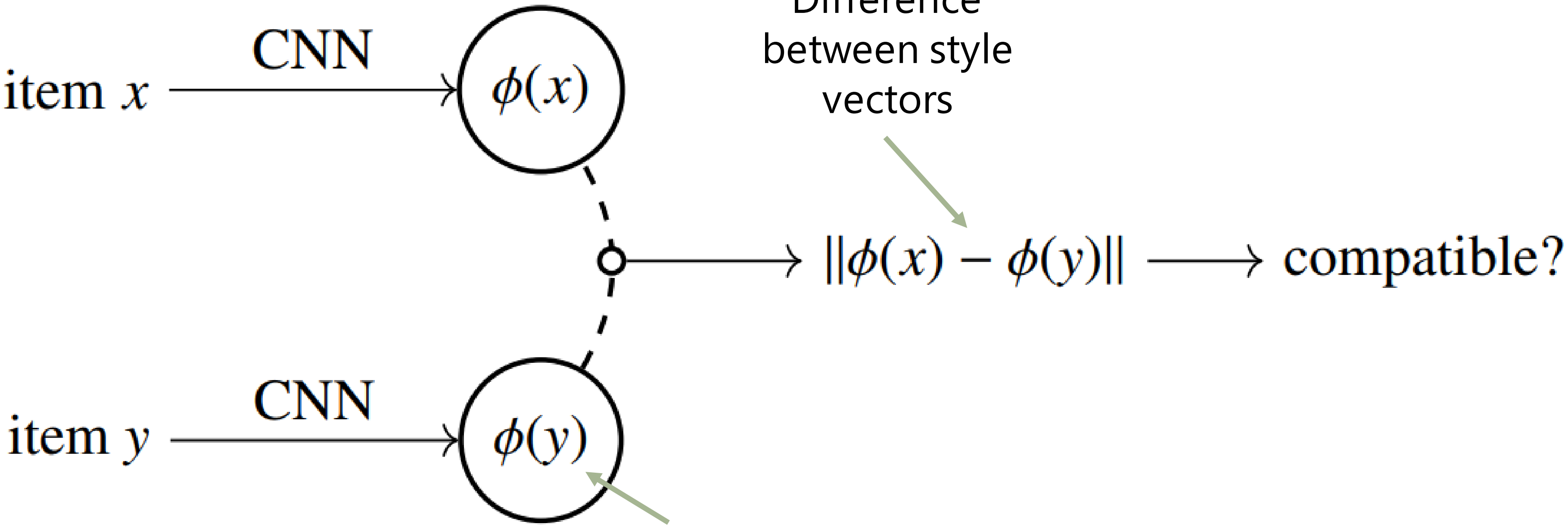
$$d(i, j) = \|s_i - s_j\|_2^2; \quad \text{where} \quad s_i = E \times f_i.$$

"Style-space" embedding of *i*

Ref.: Image-based recommendations on styles and substitutes

A little fancier... (Veit et al. 2015)

- Can do the same thing end-to-end (Siamese network):



Difference between style vectors

"Style-space" embedding of *y*

Learning visual clothing style with heterogeneous dyadic co-occurrences

A little fancier...

- Above models are not personalized, but can be personalized by learning user-specific embeddings:

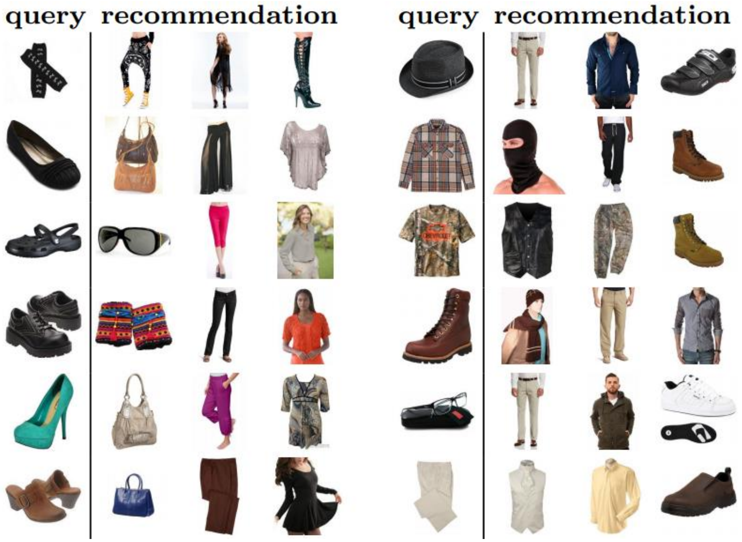
$$d_u(i, j) = \sum_k (\gamma_{u,k} s_{ik} - \underset{\uparrow}{\gamma_{u,k}} s_{jk})^2$$

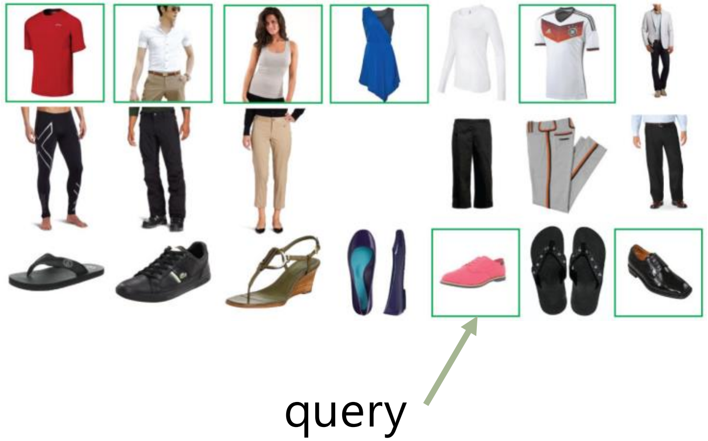
Weighting of style components by user *u*

- (of course, this is data hungry and requires several training samples per user)

(example recommendations)

- Both work okay (can estimate compatibility accurately)





- Experiments can verify (somewhat) that compatibility learned from (e.g.) Amazon co-purchases matches compatible items "in the wild"

Improvements: non-metric relationships

- Non-metric relationships (Wang et al. 2018, and others)

$$\underbrace{\sum_{(i,j)\in C} \log\sigma(\gamma_i\cdot\gamma'_j)}_{\text{complementary pairs}} + \overbrace{\sum_{(i,j)\in C^-} \log\sigma(\gamma_i\cdot\gamma'_j)}^{\text{non-complementary pairs}} .$$

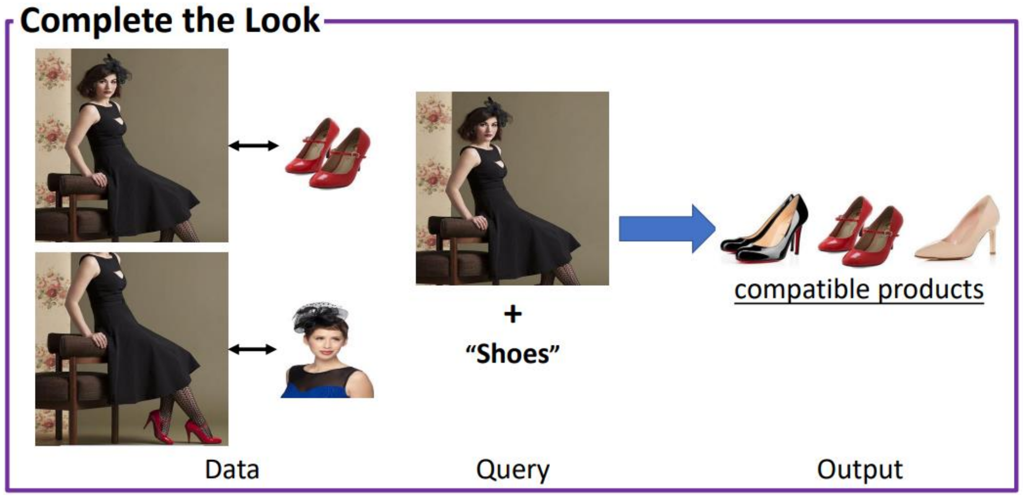
"Source" and "target" style spaces

(or just $d(i,j) = \|\gamma_i - \gamma'_j\|_2^2.$)

e.g. a path-constrained framework for discriminating substitutable and complementary products in e-commerce

Improvements: "wild" datasets

- Compatibility estimates can also be harvested from "wild" or from curated images



Complete the Look: Scene-based complementary product recommendation

Further thoughts

- How can we construct better groundtruth?
  - What are the problems with "harvested" groundtruth from Amazon?
  - Is groundtruth harvested from (e.g.) curated images of models ("wild" data) really "better"?

Further thoughts

- To what extent are compatibility functions explainable from images? What other data would be useful?
- Does personalization actually matter in this scenario?
- Are pairwise compatibility functions "enough"?

Further thoughts

- Does personalization actually matter in this scenario?
- Are pairwise compatibility functions "enough"?

Web Mining and Recommender Systems

Visually-aware recommendation

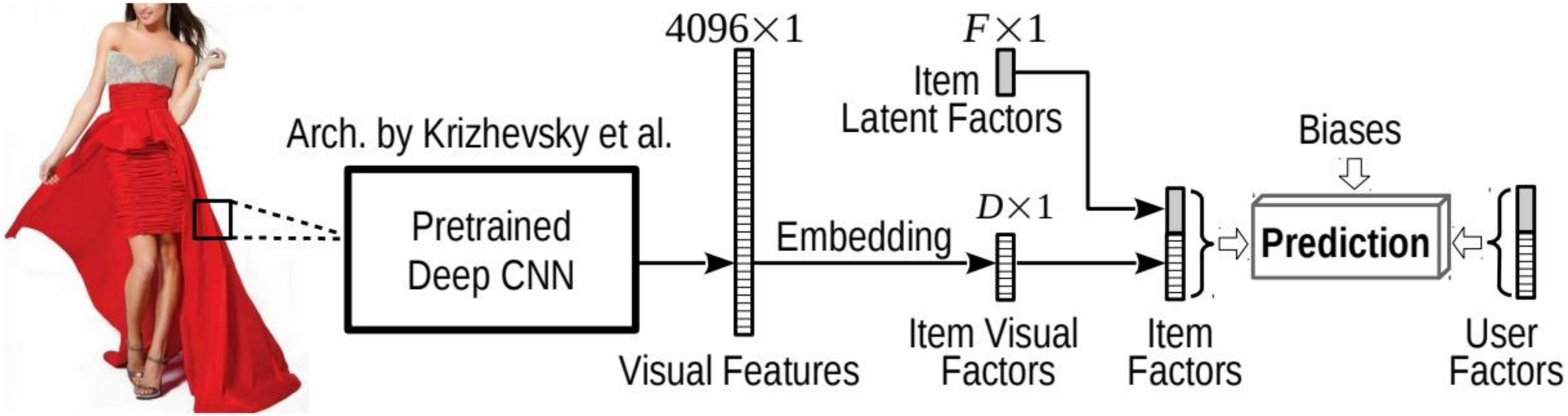
Visual recommendation

Goal is to extend "traditional" recommender systems to incorporate information from product images

- Fashion choices are guided by visual signals, so fashion recommendation should consider visual information
- Visual data could help in "long-tail" or cold-start scenarios
- Could also make recommender systems more interpretable
- Other dynamics are at play, e.g. how does fashion change over time?

Initial attempts…

- Visual data is treated much like any other feature (VBPR)



- Main challenge is how to deal with high-dimensional image data, how to handle cold-start, etc.

Visual Bayesian Personalized Ranking

Initial attempts…

- Image embeddings are incorporated into a latent-factor model:

$$x_{u,i} = \alpha + \beta_u + \beta_i + \gamma_u \cdot (E f_i)$$

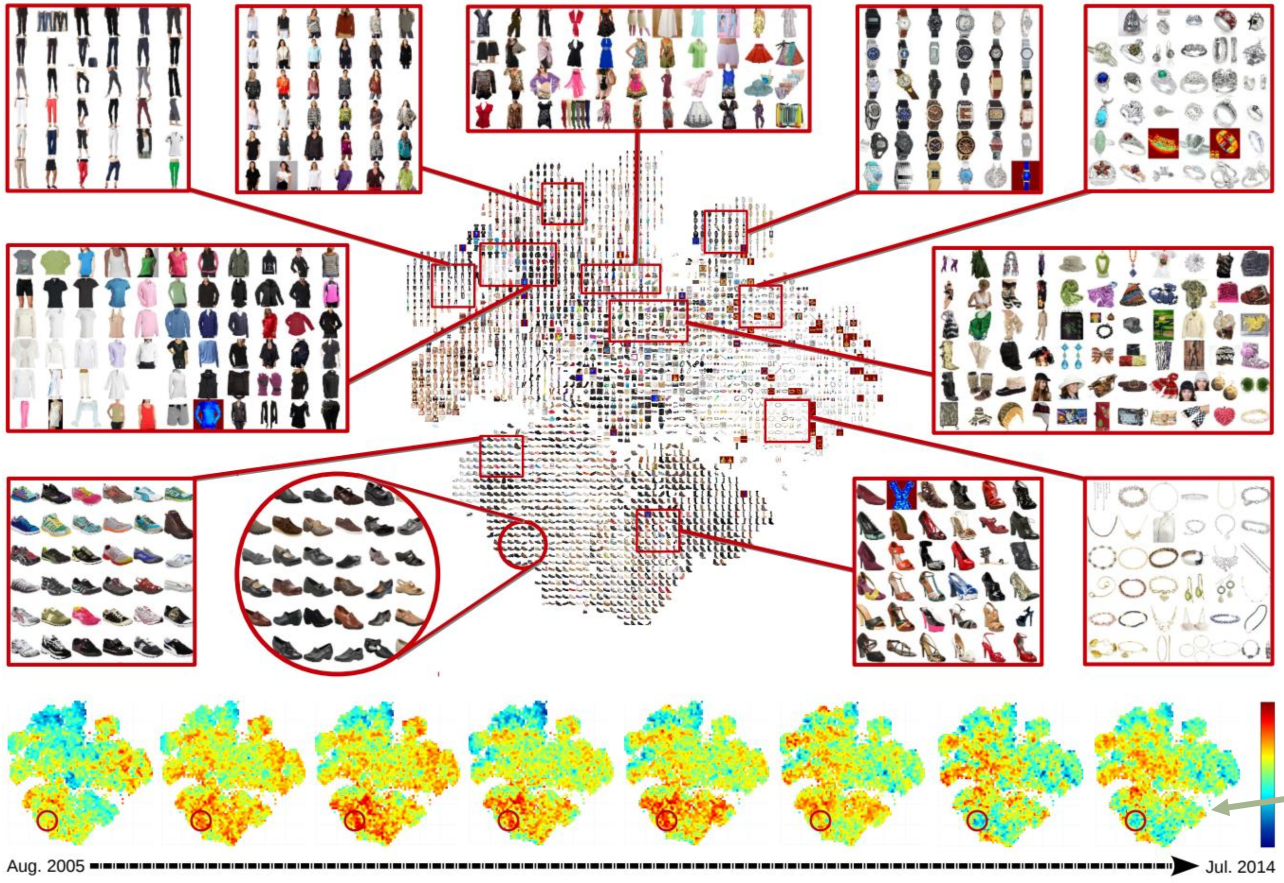
Extensions: Incorporating temporal dynamics

- How can we handle the dynamics of fashion over time?
- One solution is essentially to learn a sequence of temporal "bins" describing different periods (see also e.g. temporal models on Netflix)

$$\underbrace{\widehat{x}_{u,i}(t)}_{\substack{\text{preference of user } u\\ \text{towards item } i \text{ at time } t}} = \underbrace{\alpha + \beta_u + \underbrace{\beta_i(t) + \beta_{C_i}(t)}_{\text{temporal non-visual biases}} + \underbrace{\langle \overbrace{\beta(t)}^{\text{defined by Eq. 10}}, f_i \rangle}_{\text{temporal visual bias}}}_{\text{bias terms}} + \underbrace{\underbrace{\langle \gamma_u, \gamma_i \rangle}_{\text{non-visual interaction}} + \underbrace{\langle \overbrace{\theta_u(t)}^{\text{defined by Eq. 9}}, \overbrace{\theta_i(t)}^{\text{defined by Eq. 7}} \rangle}_{\text{temporal visual interaction}}}_{\text{user-item interactions}}.$$

Ups and Downs: Modeling the Visual Evolution of
Fashion Trends with One-Class Collaborative Filtering

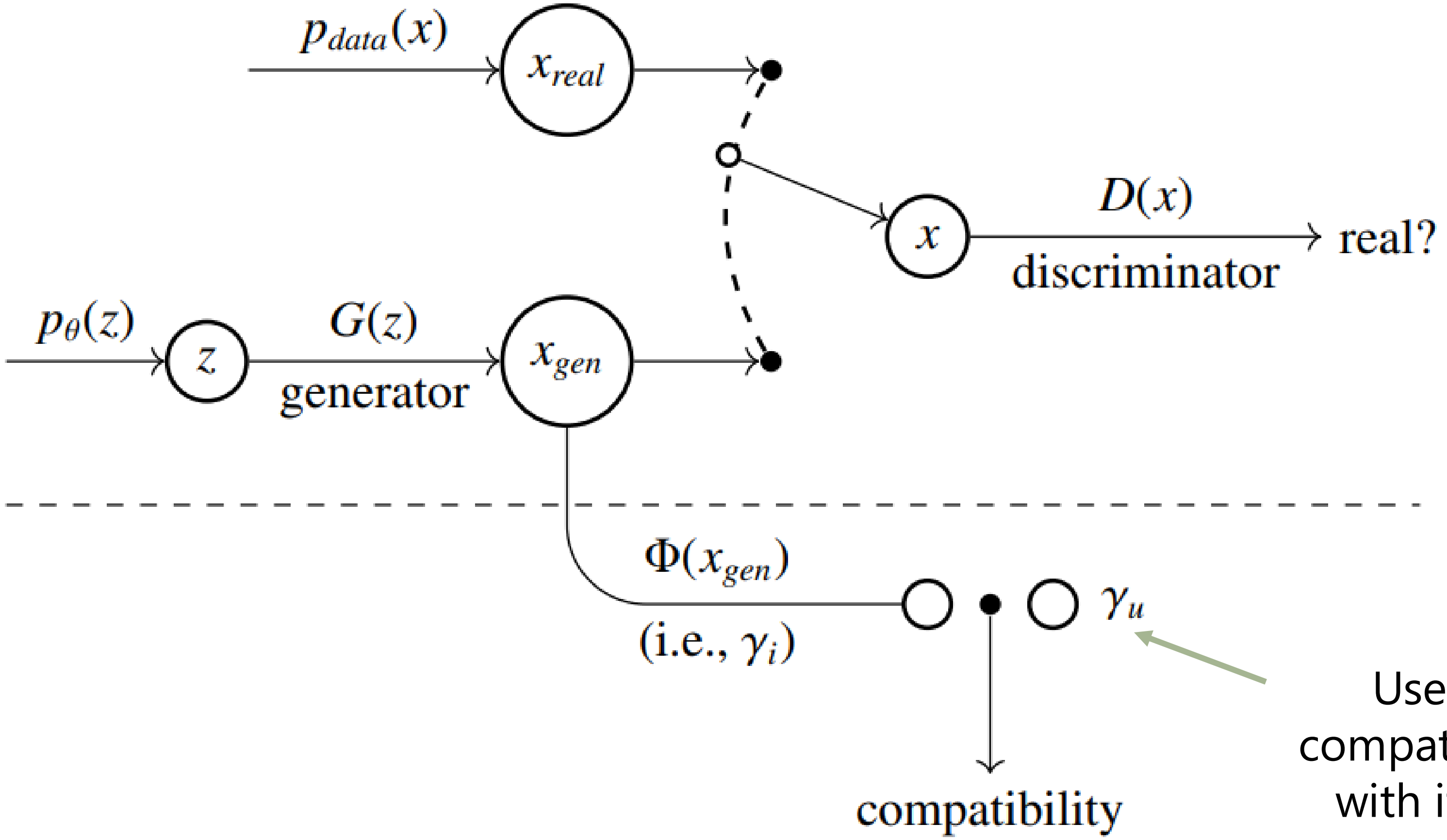
Extensions: Incorporating temporal dynamics



Which style components are "hot" during a particular year

Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering

End-to-end ("deep" models)



User's compatibility with item

Visually-Aware Fashion Recommendation and Design with Generative Image Models

Further thoughts…

- Such models are mostly useful for cold-start settings; unclear whether "real" datasets are as "cool" as academic data
- Image-based models mainly learn fine-grained product categories (i.e., they classify the image). Do they really learn "fashion" dimensions or just categories? Could we do the same with high-quality category data?
- Likewise, temporal models mostly learn sales volumes rather than the subtleties of what's really "fashionable"

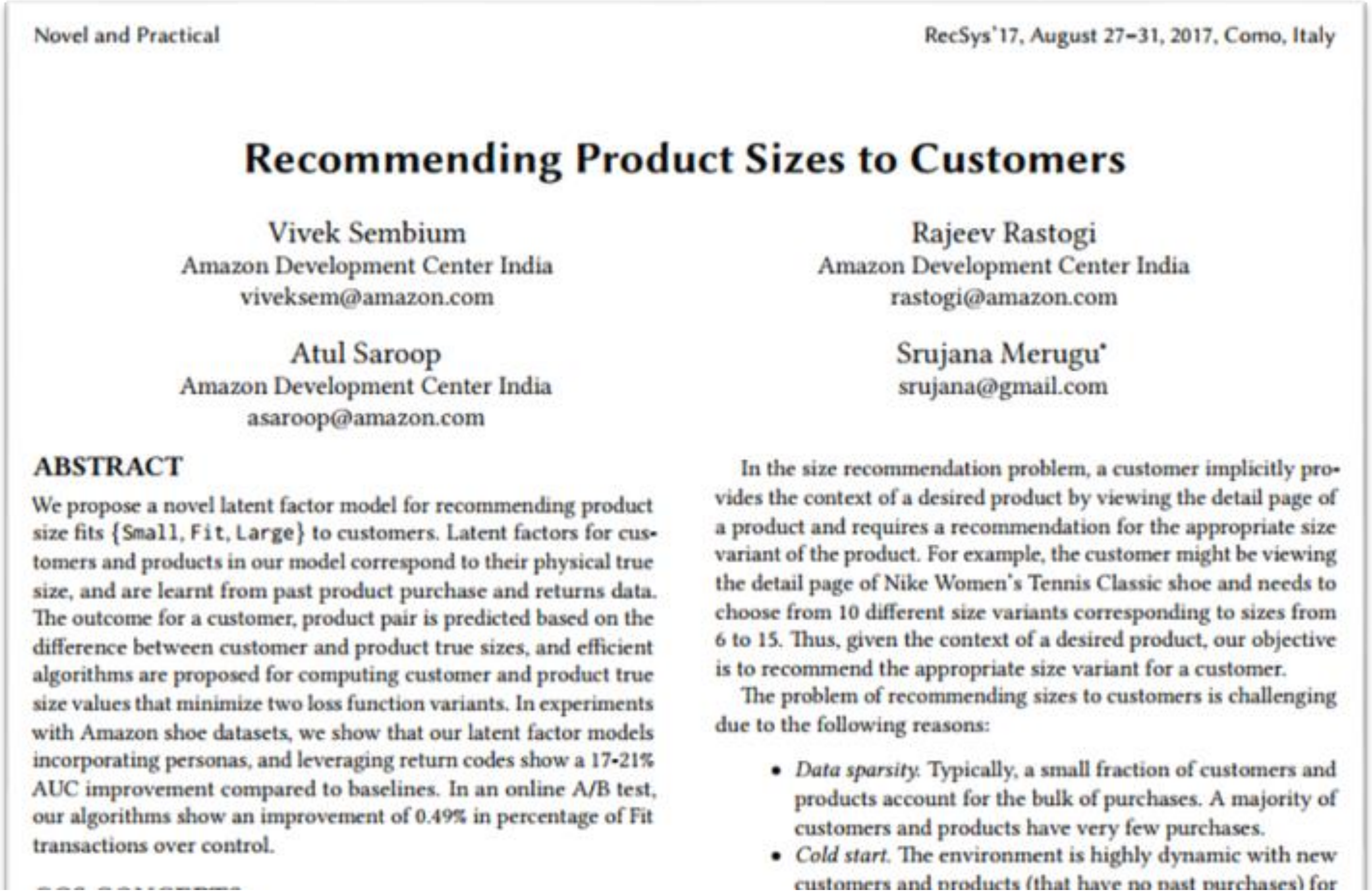
Web Mining and Recommender Systems

Fit prediction

Fit prediction – some Qs

- What do fit compatibility functions really look like? Inner product spaces? Metric spaces? Something else?
- How to deal with the fact that items don't have consistent sizes, but even that *instances of the same item* may not be the same size?
- Users also share accounts, have dynamic sizes, and may want different sizes in different contexts

Recommending product sizes to customers



Recommending Product Sizes to Customers

Vivek Sembium
Amazon Development Center India
viveksem@amazon.com

Rajeev Rastogi
Amazon Development Center India
rastogi@amazon.com

Atul Saroop
Amazon Development Center India
asaroop@amazon.com

Srujana Merugu*
srujana@gmail.com

ABSTRACT

We propose a novel latent factor model for recommending product size fits {Small, Fit, Large} to customers. Latent factors for customers and products in our model correspond to their physical true size, and are learnt from past product purchase and returns data. The outcome for a customer, product pair is predicted based on the difference between customer and product true sizes, and efficient algorithms are proposed for computing customer and product true size values that minimize two loss function variants. In experiments with Amazon shoe datasets, we show that our latent factor models incorporating personas, and leveraging return codes show a 17-21% AUC improvement compared to baselines. In an online A/B test, our algorithms show an improvement of 0.49% in percentage of Fit transactions over control.

In the size recommendation problem, a customer implicitly provides the context of a desired product by viewing the detail page of a product and requires a recommendation for the appropriate size variant of the product. For example, the customer might be viewing the detail page of Nike Women's Tennis Classic shoe and needs to choose from 10 different size variants corresponding to sizes from 6 to 15. Thus, given the context of a desired product, our objective is to recommend the appropriate size variant for a customer.

The problem of recommending sizes to customers is challenging due to the following reasons:

- *Data sparsity.* Typically, a small fraction of customers and products account for the bulk of purchases. A majority of customers and products have very few purchases.
- *Cold start.* The environment is highly dynamic with new customers and products (that have no past purchases) for

Recommending product sizes to customers

Goal: Build a recommender system that predicts whether an item will "fit":

$$(u, i) \rightarrow \{\text{small}, \text{fit}, \text{large}\}$$

Recommending product sizes to customers

Challenges:

- **Data sparsity:** people have very few purchases from which to estimate size
- **Cold-start:** How to handle new customers and products with no past purchases?
- **Multiple personas:** Several customers may use the same account

Recommending product sizes to customers

Data:

- Shoe transactions from Amazon.com
- For each shoe *j*, we have a reported size *c_j* (from the manufacturer), but this may not be correct!
- Need to estimate the customer's size *(s_i*), as well as the product's **true** size *(t_j)*

Recommending product sizes to customers

Loss function:

$$\begin{tikzcd}
f_w(s_i, t_j) + b \arrow[d] \\
f_w(s_i, t_j) = w \cdot (s_i - t_j)
\end{tikzcd}$$

Recommending product sizes to customers

Loss function:

$$L(y_{ij}, f_w(s_i, t_j)) = \begin{cases} L^{bin}(+1, f_w(s_i, t_j) - b_2) & \text{if } y_{ij} = \texttt{Small} \\ (L^{bin}(-1, f_w(s_i, t_j) - b_2) & \\ +L^{bin}(+1, f_w(s_i, t_j) - b_1)) & \text{if } y_{ij} = \texttt{Fit} \\ L^{bin}(-1, f_w(s_i, t_j) - b_1) & \text{if } y_{ij} = \texttt{Large} \end{cases}$$

$$L(y_{ij}, f_w(s_i, t_j)) = \begin{cases} \log(\frac{1}{1+e^{-f_w(s_i,t_j)+b_2}}) & \text{if } y_{ij} = \texttt{Small} \\ (\log(\frac{1}{1+e^{f_w(s_i,t_j)-b_2}}) & \\ +\log(\frac{1}{1+e^{-f_w(s_i,t_j)+b_1}})) & \text{if } y_{ij} = \texttt{Fit} \\ \log(\frac{1}{1+e^{f_w(s_i,t_j)-b_1}}) & \text{if } y_{ij} = \texttt{Large} \end{cases}$$

Recommending product sizes to customers

Loss function:

$$L(y_{ij}, f_w(s_i, t_j)) = \begin{cases} \max\{0, 1 - f_w(s_i, t_j) + b_2\} & \text{if } y_{ij} = \texttt{Small} \\ (\max\{0, 1 + f_w(s_i, t_j) - b_2\} & \\ + \max\{0, 1 - f_w(s_i, t_j) + b_1\}) & \text{if } y_{ij} = \texttt{Fit} \\ \max\{0, 1 + f_w(s_i, t_j) - b_1\} & \text{if } y_{ij} = \texttt{Large} \end{cases}$$

Recommending product sizes to customers

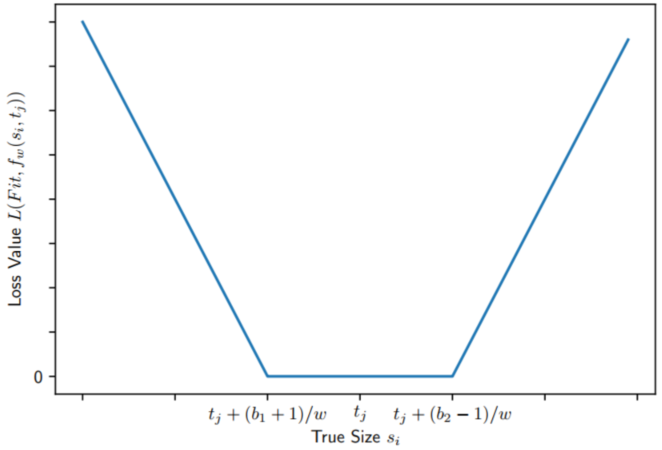


Figure 1: Hinge loss value for a `Fit` transaction vs s_i.

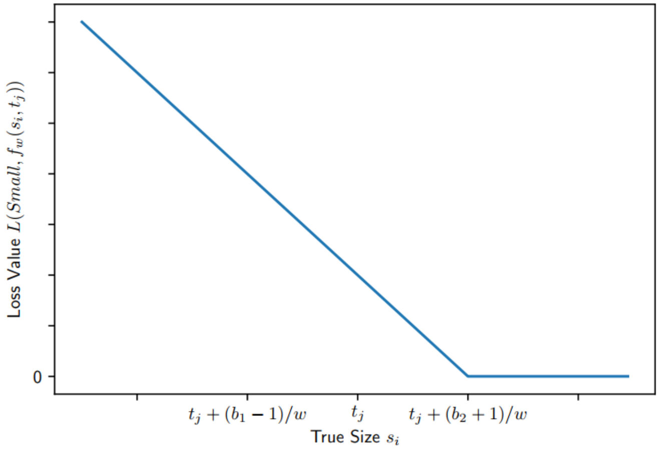


Figure 2: Hinge loss value for a `Small` transaction vs s_i.

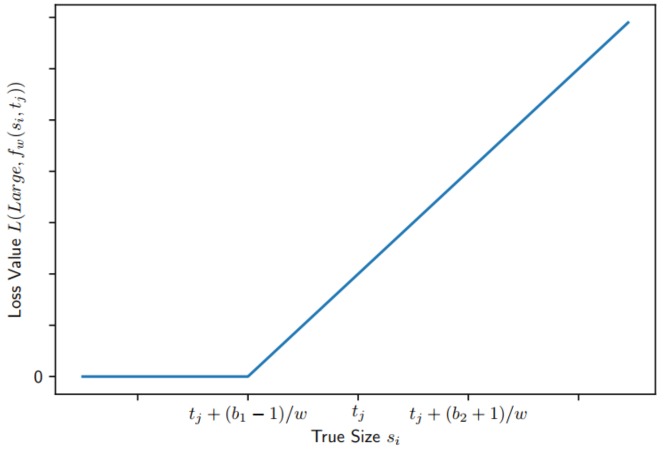


Figure 3: Hinge loss value for a `Large` transaction vs s_i.

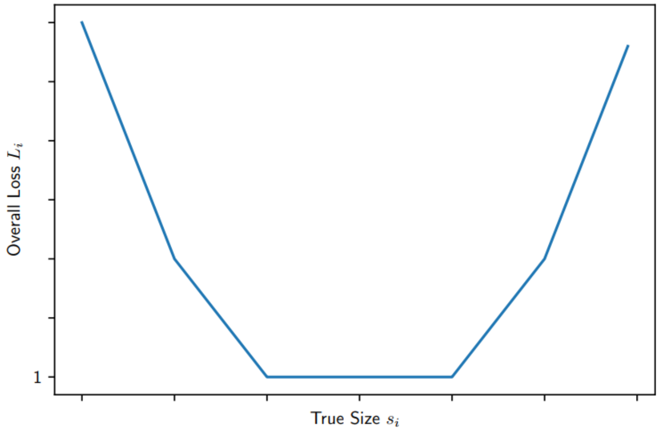
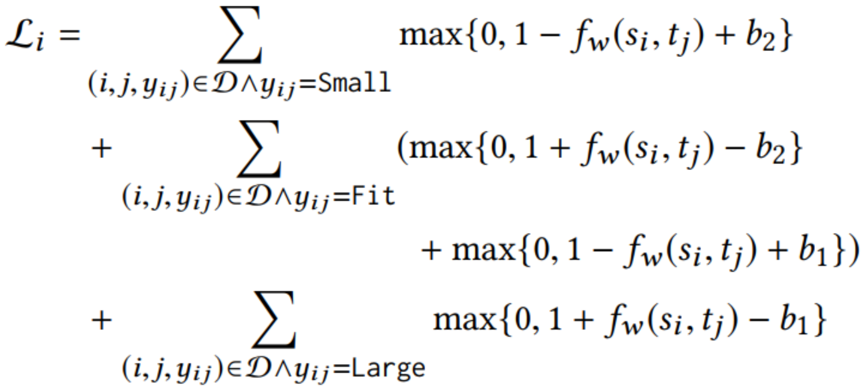


Figure 4: Illustrative overall hinge loss vs s_i.

Recommending product sizes to customers

Loss function:



$$
\begin{aligned}
\mathcal{L}_i = & \sum_{(i,j,y_{ij})\in\mathcal{D}\wedge y_{ij}=\texttt{Small}} \max\{0, 1 - f_w(s_i, t_j) + b_2\} \\
& + \sum_{(i,j,y_{ij})\in\mathcal{D}\wedge y_{ij}=\texttt{Fit}} (\max\{0, 1 + f_w(s_i, t_j) - b_2\} \\
& \qquad\qquad + \max\{0, 1 - f_w(s_i, t_j) + b_1\}) \\
& + \sum_{(i,j,y_{ij})\in\mathcal{D}\wedge y_{ij}=\texttt{Large}} \max\{0, 1 + f_w(s_i, t_j) - b_1\}
\end{aligned}
$$

Recommending product sizes to customers

Extensions:

- *Multi-dimensional sizes*

$$w_1(s_{i_1} - t_{j_1}) + w_2(s_{i_2} - t_{j_2})$$

- *Customer and product features*

$$w(s_i - t_j) + \phi(x, i)w'$$

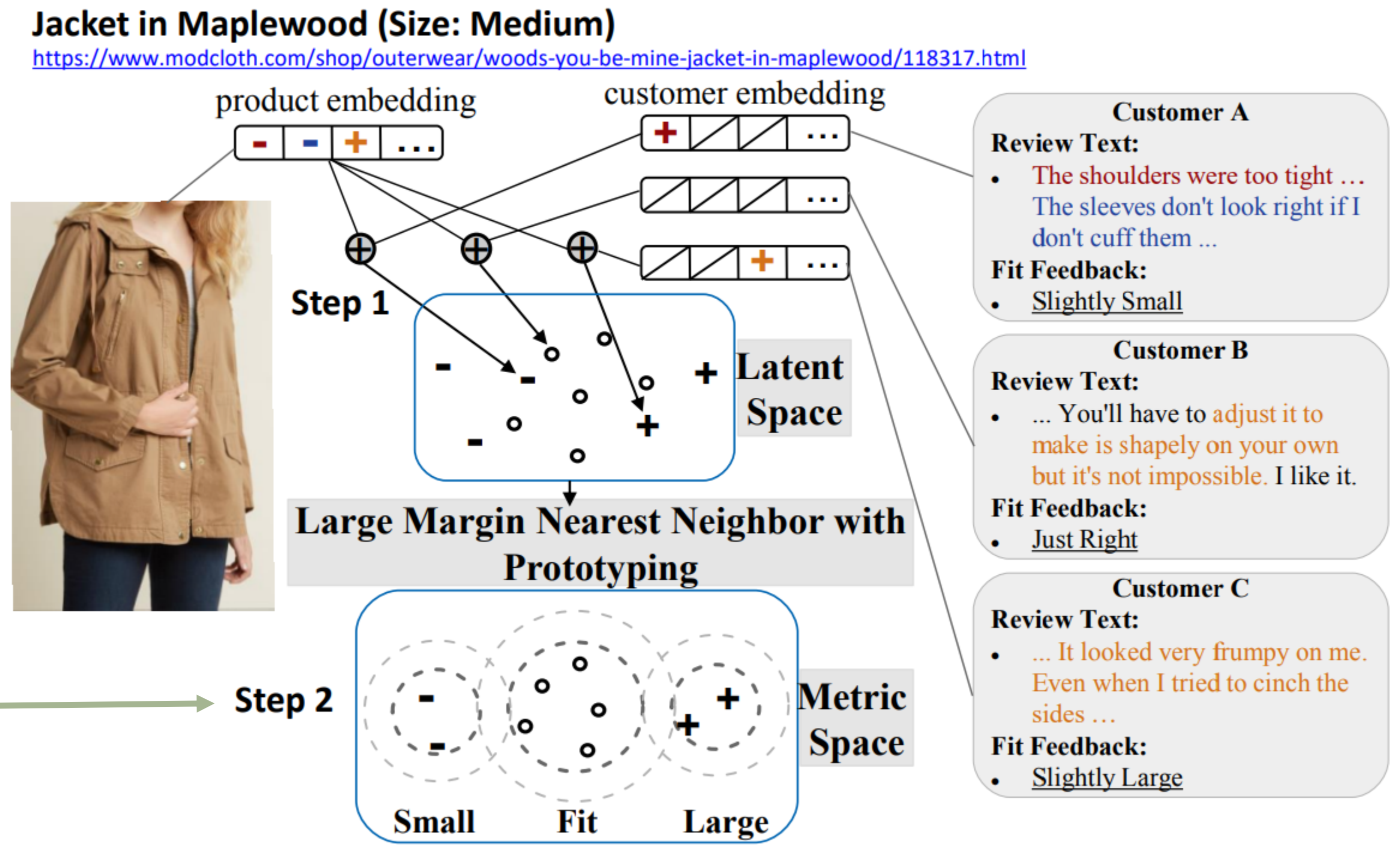
- *User personas*

Recommending product sizes to customers

Morals of the story:

- Very simple model that actually works well in production
- Only a single parameter per user and per item!

Fit prediction – more complex



Jacket in Maplewood (Size: Medium)

https://www.modcloth.com/shop/outerwear/woods-you-be-mine-jacket-in-maplewood/118317.html

Customer A

Review Text:

- The shoulders were too tight … The sleeves don't look right if I don't cuff them ...

Fit Feedback:

- Slightly Small

Customer B

Review Text:

- ... You'll have to adjust it to make is shapely on your own but it's not impossible. I like it.

Fit Feedback:

- Just Right

Customer C

Review Text:

- ... It looked very frumpy on me. Even when I tried to cinch the sides …

Fit Feedback:

- Slightly Large

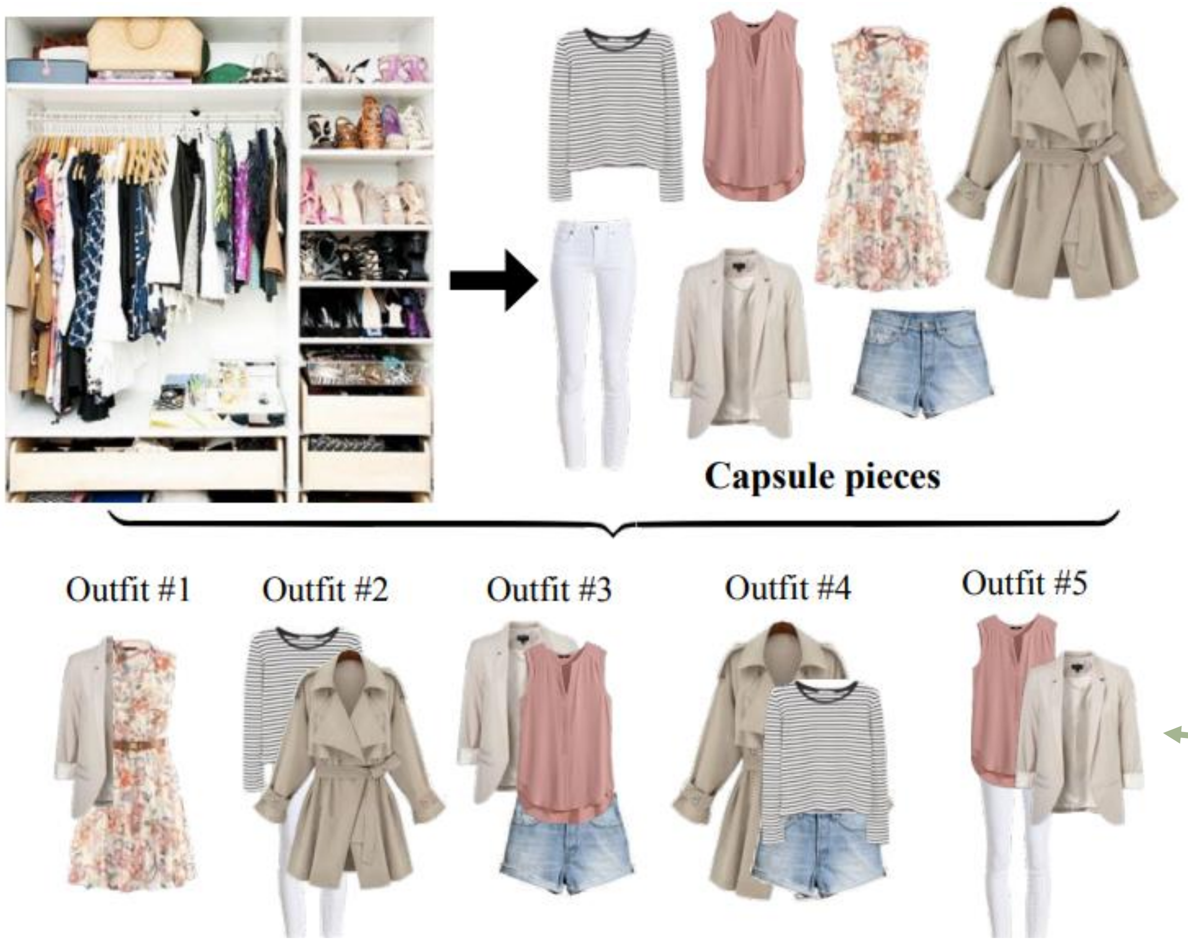
User embeddings should be close to items that fit, and far from items that don't

Decomposing Fit Semantics for Product Size Recommendation in Metric Spaces

Web Mining and Recommender Systems

Other fashion recommendation tasks

Wardrobes and sets



Generate a set of items

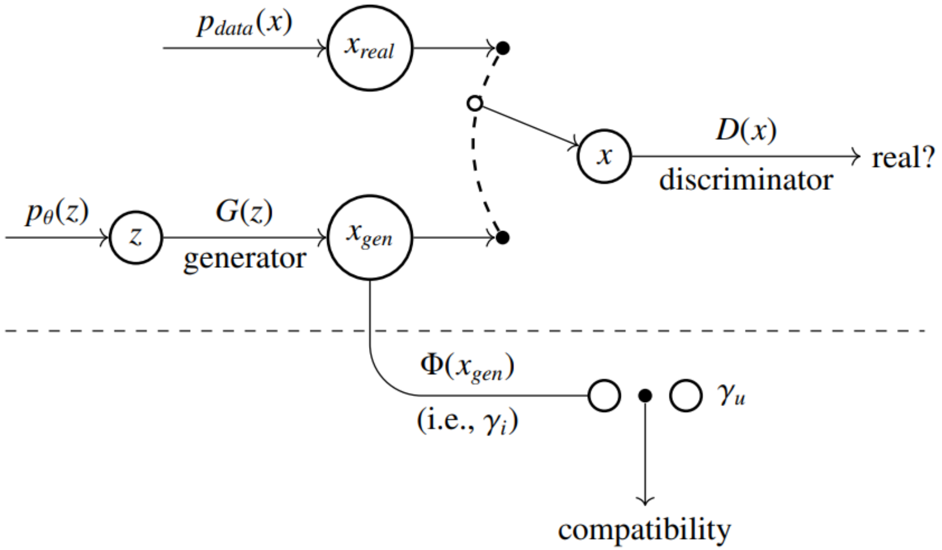
That can be combined in various ways to generate many compatible outfits

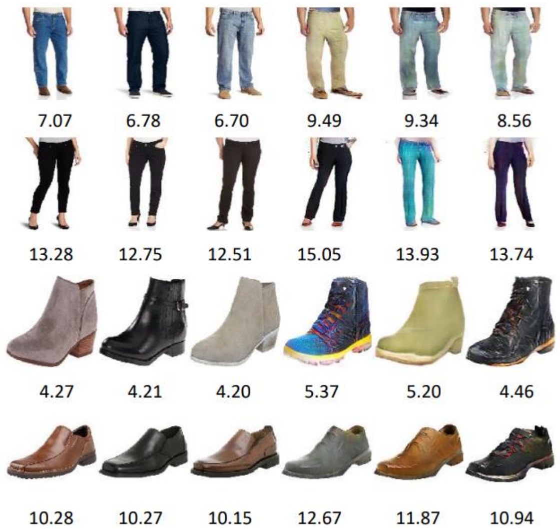
Creating Capsule Wardrobes from Fashion Images

Wardrobes and sets – further thoughts

- Do complex, high-order functions really add anything beyond what pairwise compatibility can do? Is all that complexity really worth it?
- Pairwise compatibility data (in spite of its problems) is easy to harvest. Can enough outfit (or set) data be harvested for this to be worthwhile?

Generation & design





Generated items that are highly compatible with some user

Visually-Aware Fashion Recommendation and Design with Generative Image Models

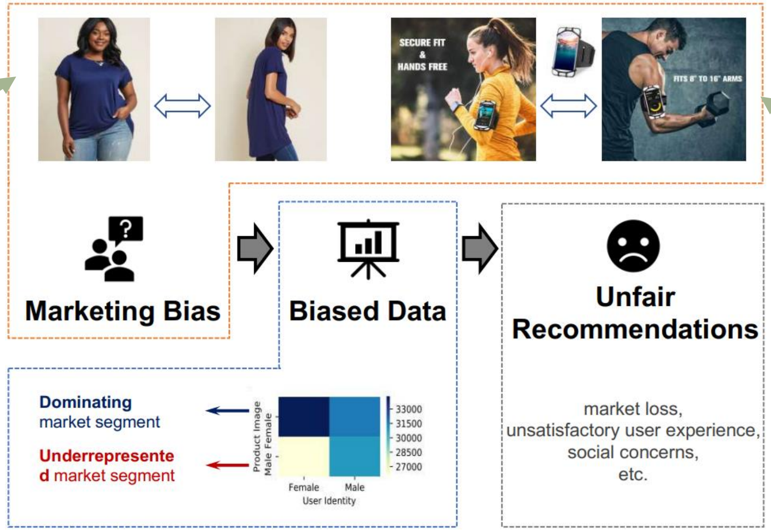
Generation & design – further thoughts

- Recommender systems can be used in the context of image generation, though this is a long way from actually being able to build wearable products
- Ethics of *fast fashion* etc.

Fairness (see next week!)

- Fairness questions in fashion (and recommendation in general) are quite different from traditional problems in fair ML

Body type of model in marketing image



Gender of model in marketing image

- In what other ways may "long-tail" users be underserved by recommendation (size, skin-tone, income, etc.)

Addressing marketing bias in product recommendations

Summary

- Many unique tasks (and plenty of open ones!) in fashion recommendation
- Datasets are a bottleneck (more so than models?) in terms of getting useful results. Extremely difficult to harvest data that corresponds to "real" fashion preferences
- Do visually-aware models really capture "fashion" or just high-level object characteristics?
- Are complex models really necessary?