## How does this work?

- You are an example  $\mathbf{x^{(i)}}$ , expressed as a vector of features
  - $\mathbf{x}_{j}^{(i)}$  is your "rating" for product j
- ullet The number n of products is large
  - lacktriangle You have rated only a small fraction  $n_i < n$
- ullet You have *not* rated product j'
  - How can the system recommend j' to you?

## PCA to the rescue!

- ullet Perform PCA on  ${f X}$  (m is number of users; n is number of products)
- The Principal Components are
  - "Concepts" that identify groups of products

## We will

- Re-express your ratings of concrete product  $\mathbf{x}^{(i)}$
- Into strength of concepts  $\tilde{\mathbf{x}}^{(i)}$
- ullet Find other users  $i^\prime$  with similar strength of concepts

$$ilde{\mathbf{x}}^{(i')} pprox ilde{\mathbf{x}}^{(\mathbf{i})}$$

- Deduce that you (user i) have similar tastes to user  $i^\prime$
- Recommend to you (user i) any product j
  - where  $\mathbf{x}_{j}^{i')}$  is high

This is roughly how Netflix recommendations work.

- Products are Movies
- Principal Components ("concepts") turn out to be Movie genres
  - Action, Comedy, Romance, Gender-specific

So if your movie preferences lean to Comedy, Netflix will recommend to you Comedytype movies Although this seems like a simple application of PCA

- There is a giant catch!
- $\mathbf{X}$  is sparse (lots of empty entries)
  - How many of the thousands of movies in Netflix have you rated?

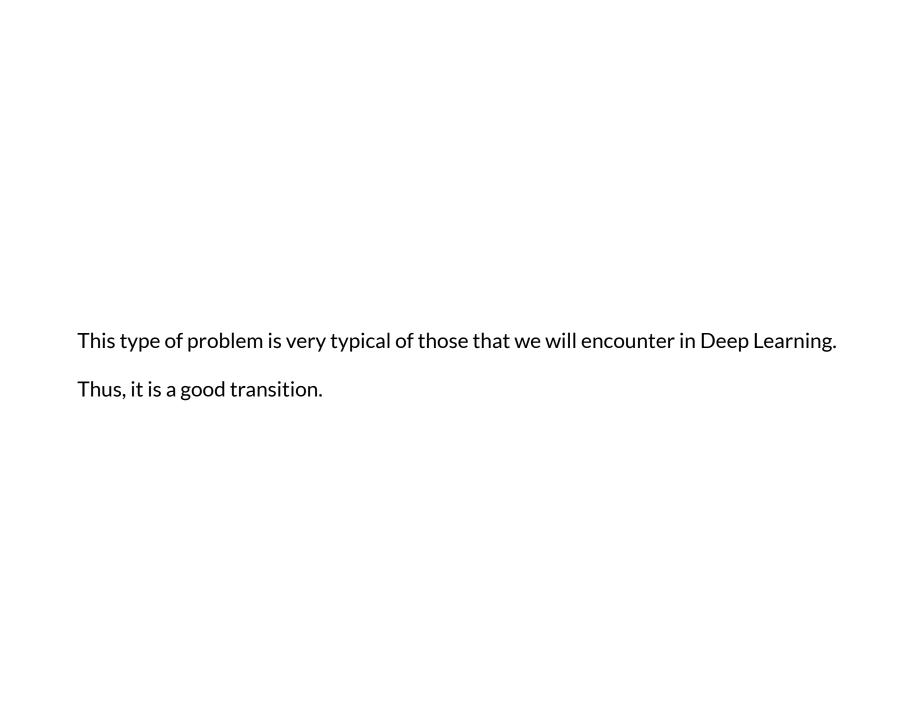
How do we factor a matrix with undefined entries?

Let's go the the <u>notebook section on Pseudo Matrix Factorization</u> (<u>Unsupervised.ipynb#Recommender-Systems:-Pseudo-SVD</u>)

## Pseudo Matrix factorization wrap-up

The techniques in Pseudo factorization are a nice bridge between Classical ML and Deep Learning

- An interesting Loss function
- Not amenable to closed form solution (because of missing entries)
- But approximated using our generic optimization tool
  - Gradient Descent



```
In [4]: print("Done")
```

Done