Week 2: RECOMMENDER SYSTEMS

Example: predicting movie ratings
- users rated each of 5 movies (0-5 ★)
- some users didn't rate some movies
- n_u = # users, n_m = # movies, $r(i,j) = \begin{cases} 1 & \text{if } j \text{ rated } i \\ 0 & \text{else} \end{cases}$
  $y^{(i,j)}$ = rating of i by j (def. only if $r(i,j)=1$)

What if we have features for the movies?
- e.g. x_1 = romance score, x_2 = action score
- for user j, predict rating for i as $w^{(j)} \cdot x^{(i)} + b^{(j)}$
  where $w^{(j)}$ and $b^{(j)}$ are fit per-user

Cost function to learn $w^{(j)}, b^{(j)}$

$$L(w^{(j)}, b^{(j)}) = \frac{1}{2} \sum_{i:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)} \right)^2$$

i.e. try to select $w^{(j)}, b^{(j)}$ to minimize error in predicting labeled data.

to find all: $J(w^{(1)}, \dots, w^{(n_u)}, b^{(1)}, \dots, b^{(n_u)}) = \sum_{j=1}^{n_u} L^{(j)}$

$$+ \underbrace{\frac{\lambda}{2m^{(i)}} \sum_{k=1}^{n} (w_k^{(j)})^2}_{\text{reg. term}}$$

What if we don't have per-item features?
See next section ☺

Collaborative Filtering

Can we learn per-item features from the user rating data?

Suppose we already have $(w^{(j)}, b^{(j)})$ for the users. Can we learn the features $x^{(i)}$ from this?

$$J(x^{(i)}) = \frac{1}{2} \sum_{j:r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2$$

So this is nice but where do we get the $(w^{(j)}, b^{(j)})$ from?

Combine the above two approaches — ranging over all $j:r(i,j)=1$ in one and $j:r(i,j)=1$ in the other, then summed over all samples, gives

$$J(\vec{w}, \vec{b}, \vec{x}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left(w^{(j)} \cdot x^{(i)} + b^{(j)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(w_k^{(j)}\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2$$

this is crazy, we gradient descent the weights/biases AND THE FEATURES SIMULTANEOUSLY

This works because we have enough structure in the model itself and enough information in the labels — rest is just learned! Features just params!

Binary labels

Predicting labels $\{1, 0\}$
- chose to purchase after viewing
- liked after viewing
- at least 30 sec interaction/viewing
- clicks

Predicting $y^{(i,j)}$ with $w^{(j)} \cdot x^{(i)} + b^{(j)}$ is regression. Instead predict $P(y^{(i,j)} = 1)$ given model $g(w^{(j)} \cdot x^{(i)} + b^{(j)})$, $g(z) = \frac{1}{1+e^{-z}}$

and $L(f_{w,b,x}) = -y^{(i,j)} \log(f_{w,b,x}(x)) - (1-y^{(i,j)}) \log(1-f)$

(binary cross-entropy cost function)

$$J(w,b,x) = \sum_{(i,j):r(i,j)=1} L(f_{(w,b,x)}(x), y^{(i,j)})$$

Implementation: Mean Normalization

- Normalize ratings first (mean normalization)
- this helps avoid a case where you learn all-zero weights + biases for users who have rated nothing
- This basically makes new users receive predictions that correspond to the mean
- Also training will be faster

Building collaborative filtering with Tensorflow

- Set up an optimizer
- write a training loop
- do autograd to get derivatives
- apply derivatives to optimizer

loop N times until convergence

Note this isn't a neural network, it's just using the lib.

Related Items

- Using collaborative filtering, the learned feature vectors $x^{(i)}$ are hard to interpret
- But they do encode data about the entities into a vector space! Just do nearest neighbor search from a given feature vector to find "similar" items

Limitations of Collaborative Filtering

- Cold start problem: how to rank new items, what to show new users?
- How to use side information? i.e. additional data, attributes etc. of entities or users?

Content-Based Filtering

Compare:
- Collaborative filtering recommends based on ratings of users who gave similar ratings as you
- Content-based filtering recommends items based on features of users and items

Example features
- Users
  - age
  - gender (1-hot encoded)
  - country (1-hot encoded)
  - movies watched (e.g. from top-1000)
  - avg. rating per genre
  - etc.

$\vec{x}_u^{(j)}$ for user j

- Items (e.g. movies)
  - Year
  - Genres
  - Reviews
  - Avg. rating

$\vec{x}_m^{(i)}$ for movie i (can be diff size from $\vec{x}_u^{(j)}$)

Prediction: $\vec{v}_u^{(j)} \cdot \vec{v}_m^{(i)}$ = rating of user j for movie i

where $\vec{v}_u^{(j)}, \vec{v}_m^{(i)}$ computed from the above features $\vec{x}_u^{(j)}, \vec{x}_m^{(i)}$

For classification can use $g(\vec{v}_u^{(j)} \cdot \vec{v}_m^{(i)})$ where g is sigmoid

How to compute the $\vec{v}$? Need to be the same size...

Use deep learning.

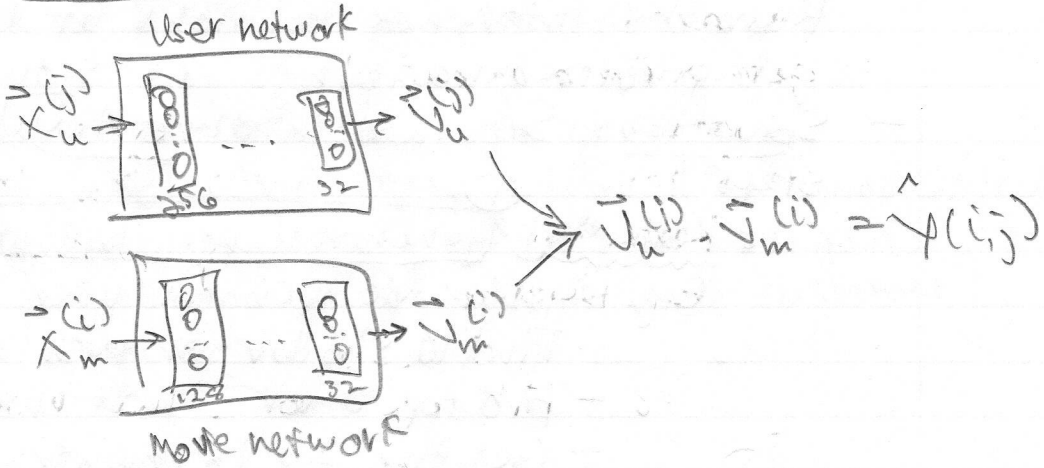
User network for $\vec{x}_u^{(j)} \in \mathbb{R}^{128}$, want $\vec{v}_u^{(j)} \in \mathbb{R}^{32}$

$\mathbb{R}^{128} \ni \vec{x}_u^{(j)} \rightarrow$ [128] $\rightarrow$ [64] $\rightarrow$ [32] $\rightarrow \vec{v}_u^{(j)} \in \mathbb{R}^{32}$

Movie network

$\mathbb{R}^{256} \ni \vec{x}_m^{(i)} \rightarrow$ [256] $\rightarrow$ [128] $\rightarrow$ [32] $\rightarrow \vec{v}_m^{(i)} \in \mathbb{R}^{32}$

But how to build and train these networks?



Cost function $J = \sum_{(i,j):r(i,j)=1} \left(\vec{v}_u^{(j)} \cdot v_m^{(i)} - y^{(i,j)}\right)^2 +$ reg. term

Now do gradient descent and backpropagation to train the entire network — user & movie networks are learned!

Finding similar items

Once you train these networks, you can use the user and movie networks to embed the (users and) movies and then find similar movies by finding the N nearest neighbors to a given movie embedding.

> **Tip:** Feature engineering is still important!

Large catalogs

- Catalog can be millions of items. Running inference millions of times to get predicted rankings is slow!
- Solution: Separate into retrieval and ranking steps:
  1. Retrieval: ① generate large list of candidates.
     E.g. - get movie recs for user X
     - Find 10 similar for each of 10 last watched
     - Find top 10 for 3 most-watched genres
     - Get top 20 in country
     ② Combine into one list (removing dupes / already watched / etc.)
  2. Ranking: Take list and rank using learned model (as above: just gen. predictions for items in this list)

★ How to decide how many candidates to fetch during retrieval? (N = candidates)
→ Offline experiments to see if higher N produces better predictions (as measured by cost fn.)

Ethics

- What is the goal?
  - movies most likely to have high ratings
  - products most likely to be purchased
  - ads most likely to be clicked
  - products generating highest profit
  - videos leading to max watch time

! Some of these can be problematic!

E.g. ads most likely to be clicked
- Travel industry: this is a good objective!
- Payday loan industry: more profit ⇒ more money for ads ⇒ more customers. This is bad!

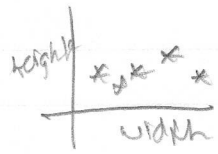
Some solutions:
- Refuse some customers
- Carefully choose objective

Other examples: maximizing engagement ⇒ more recs for toxic content (soln: content contr.), ads as organic content (soln: transparency)

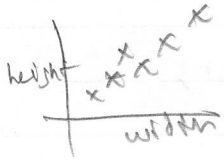
Principal Component Analysis

Goal: feature reduction, often for dev. visualization or computer vision

Example: cars, width and height.
But considering height doesn't vary much, PCA could just drop height.



Example 2: suppose length and height are correlated (e.g. linearly). PCA could give us one feature that combines them



Algorithm

1. Preprocess features (normalization to zero mean, scaling)
2. Choose an axis and project examples onto it
   → You actually want to maximize variance as this captures more information
   → Choose the axes of projection accordingly
3. Produce new coordinates on the new axes for each example (i.e. $\vec{x} \cdot \vec{u}$ where $\vec{x}$ is the example and $\vec{u}$ is the coord. vector for a new axis)
4. For additional axes, these should be orthogonal to prev. ones