Lecture 7 - Collaborative Filtering

Thursday, February 3, 2022 7:18 AM

Plan for today:
- submitty stuff ✓
- Derive matrix factorization gradient descent
- Talk regularization
- Code mode
- Collaborative filtering
- Netflix challenge
- Adapting MF to CF
- More code mode
- Considerations

Matrix factorization gradient

Gradient descent

$A = \underbrace{UVU^T}{} = P$

$a_{n+1} = a_n - \alpha \nabla f(a_n)$

α = learning rate

a_n = ~ features at current step

a_{n+1} = features for next step

$\nabla f(a_n)$ = how our error changes basedon current

Really: we consider how our error changes with changing input

→ then we move in the direction of greates decrease of greatest decrease

$$\min_{U,V} (A - UVU^T)^2$$

$$\hookrightarrow \min_{U,V} \forall_{i,j} \underbrace{(a_{ij} - u_i V u_j^T)}_{\text{our error } e_{ij}}{}^2$$

$$e_{ij}^2 = (a_{ij} - \underbrace{u_i V u_j^T}_{})^2$$

$$\frac{d e_{ij}^2}{d u_i} = (a_{ij} - u_i V u_j^T)(-V u_j^T)$$

$$= -\underbrace{e_{ij} V u_j^T}_{}$$

$$\frac{d e_{ij}^2}{d u_j} = -e_{ij} u_i V$$

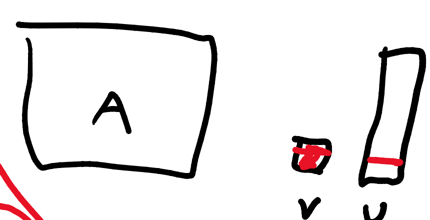
$$\frac{d e_{ij}^2}{d V} = -e_{ij} u_i u_j^T$$

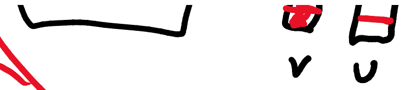
How our error is changing w.r.t. U, V

Our update equations:

$$u_i = u_i + \alpha e_{ij} V u_j^T$$

$$u_j = u_j + \alpha e_{ij} u_i V$$





$U_j = U_j + \alpha e_{ij} U_i V$ ← $O(k^2)$

$V = V + \alpha e_{ij} U_i U_j^T$ ← $O(k^2)$

$A \in \mathbb{R}^{n \times n}$ generally: $k << n$

$U \in \mathbb{R}^{n \times k}$ k: # of latent features

$V \in \mathbb{R}^{k \times k}$

Note: when naively training on A

→ if we train on all $a_{ij} \in A$ including zeros

⇒ our predictor will just output 0 for everything

$O(n^2)$

→ if we train only on nonzeros

⇒ we'll just output 1 for all

$O(n)$

Workaround 1: we weight zeros and nonzeros separately

$$\min_{U,V} \sum_{\substack{\text{nonzeros} \\ \text{in } A}} (a_{ij} - U_i V U_j^T)^2 + w \sum_{\substack{\text{zeros} \\ \text{in } A}} (a_{ij} - U_i V U_j^T)^2$$

weighting to minimize training impact of zeros

Workaround 2: only train on a sample of zeros

Issue: we aren't constraining values of U, V
=> they blow up

Regularization

→ we'll also include values of U, V within our optimization function

$$\min_{U,V} \sum_{nnz} (a_{ij} - U_i V U_j^T)^2 + w \sum_{zeros} (a_{ij} - U_i V U_j^T)^2$$

$$+ B_1 \|U\|_{fro} + B_2 \|V\|_{fro}$$

sum of squared values

parameter to control relative impact of this on overall optimization

additional "loss terms"

- $B U_i$
- $B U_j$
- $B V$

$$- B V$$

Our new update equation:

$$U_i = U_i + \alpha(e_{ij} V U_j^T - B_1 U_i)$$
$$U_j = U_j + \alpha(e_{ij} U_i V - B_1 U_j)$$
$$V = V + \alpha(e_{ij} U_i U_j^T - B_2 V)$$

Collaborative filtering

One common approach for

<u>recommender systems</u>

Definition: Making a selection from available options (filtering) for a specific user given user preferences in general (collaborative)

Similar to predictive stuff in HW 1
- → filtering what products for a user
- → Based off purchases of 'similar' users defined via Jaccard and other

and other

General approaches for C.F.:

→ like HW: define explicit similarities

→ here today: CF via M.F.

→ machine learning:
- define explicit features
- can better capture non-linear behavior that MF can't
- Issue for us: lack of quality data
  Reason: privacy concern, easy to de-anonymize

Netflix Challenge

~2007 or so

Netflix: we have a predictor for user-movie preferences

→ predicts (1-5) rating for given user

Challenge:

→ improve predictor by 10%

→ released 100s of millions of ratings

→ improve p...

→ Released 100s of millions of ratings
→ Had an internal test set for ratings

→ If improve by 10% → you get $1,000,000

→ Took a couple years for teams to do so
- Winners used aggregation of ML approaches
- MF itself got up to 7%

Interesting notes:
Temporal effects of ratings
- some movies if rated immediately vs. later got higher (Patch Adams)
- some had opposite effect (Memento)

Naive approach:

$$a_{ij} = 0.5\left(\underset{j}{\text{avg}}(a_{ij}) + \underset{i}{\text{avg}}(a_{ij})\right)$$

avg for user i   avg for movie j

Adapting MF for CF (MF4CF)

consider user-movie matrix

Consider user-movie matrix
users per row
movies per row

To represent as a graph:
- bipartite $B_1 = \{\text{users}\}$, $B_2 = \{\text{movies}\}$
$W = \{\text{ratings}\}$

We want to predict new edges
e.g.: what a user will rate a movie

Let's setup our MF
- A is a bipartite adjacency matrix
- we want to solve $A = UV^T$
$A: \mathbb{R}^{n \times m}$ $U: \mathbb{R}^{n \times k}$ $V: \mathbb{R}^{m \times k}$ movie features
user features

- Prediction for $a_{ij} = u_i v_j$
- Our minimization problem:

$$\min_{U,V} \sum_{\text{nonzeros}} (a_{ij} - u_i v_j)^2 + \beta_1 \|U\|_F + \beta_2 \|V\|_F$$

our update equations:

↳ our update equations:

$$U_i = U_i + \alpha(e_{ij}V_j - \beta_1 U_i)$$

$$V_j = V_j + \alpha(e_{ij}U_i - \beta_2 V_j)$$

Considerations and Challenges

Sparsity of data
→ much fewer ratings than non-ratings

Scale of data
→ how to adaptive

Generalizations:
→ how do we work on novel data
→ be careful not to overfitting

"Cold-start":
→ How to predict for a new user?

Black sheep:
→ some users you can't predict
→ all they are is noise