Yichen Lin, Yahya Abdul Aziz, and Franck Benichou

Mila
HEC MONTRĒAL

A Comparative Analysis of Recommendation Systems between Matrix Factorization & Deep Learning Techniques

Introduction

- **Goal**: Comparing the strengths and limitations of Matrix Factorization Techniques with AutoRec & GNN
- **MF Techniques state of the art Collaborative Filtering techniques since Netflix Context (2006)**
  - Addresses **sparsity** issue
  - Better **scalability** than content based
  - Better **predictions** than content based
- MF faces **cold start problem, poor interpretability, loss of information and scalability issues** with very large datasets
- **Deep Learning** Recommender Systems **Pros:**
  - **Non linear** transformation
  - **Representation learning** (underlying factors)
- **Deep Learning** Recommender Systems **Cons:**
  - **Interpretability**
  - **Large Data Requirement**
  - **Extensive Hyperparameter Tuning**

Matrix Factorization Recommender Systems

Predicting ratings of user *u* for item *i*

$$\hat{r}_{ui} = x_u^T \cdot y_i = \sum_k x_{uk} y_{ki}$$

Goal : Loss Function Minimization

$$L = \sum_{u,i \in s} (r_{u,i} - x_u^T \cdot y_i)^2 + \lambda_x \sum_u \|x_u\|^2 + \lambda_y \sum_u \|y_i\|^2$$

Alternating Least Square (ALS)

$$\frac{\partial L}{\partial x_U} = -2 \sum_i (r_{ui} - x_u^T \cdot y_i)\, y_i^T + 2\lambda_x x_u^T$$

$$x_u^T = r_u Y (Y^T Y + \lambda_X I)^{-1}$$

u vector derived

$$\frac{\partial L}{\partial y_i} = -2 \sum_i (r_{iu} - y_i^T \cdot x_U)\, x_u^T + 2\lambda_y y_i^T$$

$$y_i^T = r_i X (X^T X + \lambda_y I)^{-1}$$

i vector derived

Stochastic Gradient Descent (SGD)

$$\hat{r}_{ui} = \mu + b_u + b_i + x_u^T \cdot y_i$$

$$L = \sum_{u,i} (r_{u,i} - \hat{r}_{ui})^2 + \lambda_{xb} \sum_u \|b_u\|^2 + \lambda_{y_b} \sum_i \|b_i\|^2 + \lambda_{xf} \sum_u \|x_u\|^2 + \lambda_{yf} \sum_u \|y_i\|^2$$

$$\frac{\partial L}{\partial b_U} = -e_{ui} + \lambda_{xb} b_u$$

Derivative for user bias

$$b_u = b_u + \alpha(e_{ui} - \lambda_{xb} b_u) \qquad x_u = x_u + \alpha(e_{ui}\, y_i - \lambda_{xf} x_u)$$

$$b_i = b_i + \alpha(e_{ui} - \lambda_{yb} b_i) \qquad y_i = y_i + \alpha(e_{ui}\, x_u - \lambda_{yf} y_i)$$

Hyperparameters ALS

k: latent factors & Number of iterations
$\lambda : L2\ Regularization$

Hyperparameters SGD

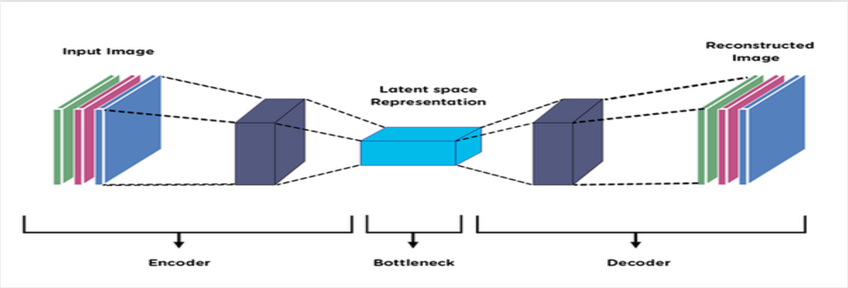
k: latent factors & α: learning rate
$\lambda : L2\ Regularization$ & Number of iterations

Experiments & Details

	Movie Lens 100k	Movie Lens 1M	Movie Lens 2.5 M	Amazon 89k
MF1 RMSE	ALS:2.95	ALS:2.99	NA	KNN: 1.34
MF2 RMSE	SGD:0.96	SGD:0.95	NA	NA
AutoRec RMSE	**1.02**	**0.90**	**0.78**	**0.0847**
GNN RMSE	**0.9041**	NA	NA	NA
MF1 Hyperparameters	**Regularization = 100, Latent Factors = 80, 25 iterations**	**Regularization = 100, Latent Factors = 80, 50 iterations**	NA	**K=5 Metric: Pearson_baseline Type: Item_based**
MF2 Hyperparameters	**α = 0.001, Latent Factors = 40, 200 iterations, regularization = 0.01**	**α = 0.01, Latent Factors = 40, 200 iterations, regularization = 0.01**	NA	NA
AutoRec HyperParameters	**Adam Optimizer, Layer = 1, α = 0.001, batch size = 512, λ= 1, hidden neurons = 500, epochs = 500, activation: sigmoid**			**Adam Optimizer, α = 0.0001, Activation: SELU, Dropout=0.8, Layers = 4**
GNN Hyperparameters	**4 R-GCN (32,32, 32,32) 1 MLP (128) 1-hop enclosing subgraph**			

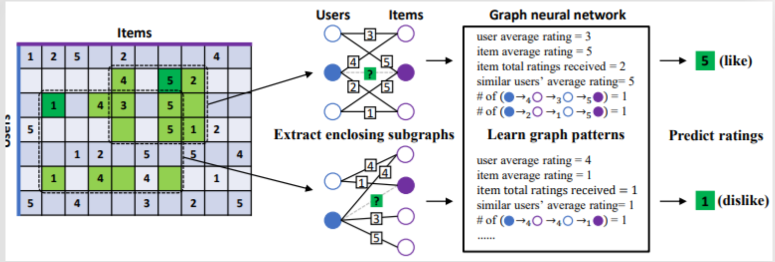
Deep Learning Techniques

Autoencoder Recommender Systems



1. Reduce dimensionality of sparse data into a latent space representation
2. High level features are learned
3. Encode the matrix as a probability distribution
4. Reconstruction based on sampling from the distribution

GNN Recommender Systems



1. Instead of learning transductive latent features, IGMC learns local graph patterns related to ratings inductively based on graph neural networks.
2. It does not rely on using side information of users/items.
3. The model learns mixed local graph patterns (such as average ratings, paths, etc.), adding new idea to matrix completion and recommendation systems.

Takeaways

- **Stochastic Gradient Descent Matrix Factorization provides the best performance amongst MF** but becomes computationally intensive the larger the dataset
- **MF methods are O(M + N)** given users do not have a lot of ratings**. Choosing the number of latent factors k is key to avoid overfitting**.
- **Autorec** generates better performance **than MF in larger datasets**
- In larger datasets, **the autoencoder allows for the implicit learning of latent features**
- AutoRec is more computationally demanding than MF but the gap closes with larger datasets
- Autorec and GNN both capture **implicit features without using extra side information**
- **Most recommendation data have graph properties**. GNN (IGMC) utilizes these properties by enabling inductive matrix completion and increases its generalization