Matrix Factorization

Matrix Factorization

Ratings Matrix $(N \times M)$
N = users M = items

- Factors:
  $10 = 2.5$
  $15 = 3.5$

Split matrix into product of 2 matrices

$\hat{R} = W U^T$ $\hat{R}$ = approx.

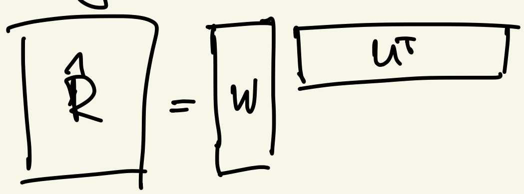
$R_{N \times M}$ is sparse $\Rightarrow$ stored as dict

$N = 130k$ $M = 26k$
$N \times M = 3{,}38B$
ratings $= 20M$

W/U should be very skinny

$W_{N \times k}$ $U_{M \times k}$
users matrix movies matrix

$k \sim 10-15$



If $k = 10$, $N = 130k$, $M = 26k$
$Nk + Mk = 1{,}56M \rightarrow$ even less space than ratings

One rating

$\hat{r}_{ij} = w_i^T u_j$ $w_i = W[i]$

$\hat{r}_{ij} = \hat{R}[i,j]$ $u_j = U[j]$

Interpretation

k elements in w_i and u_j
$k = 5$ and action, comedy, romance, horror, animation
$w_i(1)$ = how much user i likes action
$w_i(2)$ = how much user i likes comedy
$u_j(1)$ = how much movie j contains action
$u_j(2)$ = how much movie j contains comedy

$w_i^T u_j = \|w_i\| \|u_j\| \cos\theta \propto \text{sim}(i,j)$
user i correlate with attributes of movie j

Example

$$w_i = \begin{pmatrix} 1 \\ 0{,}8 \\ -1 \\ 0{,}1 \\ 1 \end{pmatrix} \quad u_j = \begin{pmatrix} 1 \\ 1{,}5 \\ -1{,}3 \\ 0 \\ 1{,}2 \end{pmatrix} \quad w_i^T \cdot u_j = 4{,}7$$

+ve . +ve → +ve
−ve . −ve → +ve
⊖ means feature is not present or not liked

$$w_i = \begin{pmatrix} 1 \\ 0{,}8 \\ -1 \\ 0{,}1 \\ 1 \end{pmatrix} \quad u_j = \begin{pmatrix} -1 \\ -1 \\ 1 \\ 0 \\ -1 \end{pmatrix} \quad = -3{,}8$$

- Features
  each feature is latent, k is the latent dimensionality
  what features mean is not known before examining them (hidden cause)

Dimensionality reduction

if $k = M$ and $N > M$ and $\text{rank}(X) = M$
$U/S/V$ are approximations (shrinked)
Truncated SVD yields best rank k approx of X
MF reduces the dimensionality of R

Training

$$R \approx \hat{R} = W U^T$$

- Squared Error (for regression)

$$J = \sum_{i,j \in \Omega} (r_{ij} - \hat{r}_{ij})^2 = \sum_{i,j \in \Omega} (r_{ij} - w_i^T u_j)^2$$

Gradient:

$$\frac{\partial J}{\partial w_i} = 2 \sum_{j \in \psi_i} (r_{ij} - w_i^T u_j)(-u_j) = 0 \rightarrow \text{min}$$

$$\sum_{j \in \psi_i} (w_i^T u_j) u_j = \sum_{j \in \psi_i} r_{ij} u_j$$

- Solving for W

$$w_i = \left(\sum_{j \in \psi_i} u_j u_j^T \right)^{-1} \sum_{j \in \psi_i} r_{ij} u_j$$

- Solving for U

$$\frac{dJ}{du_j} = 2 \sum_{j \in \Omega_j} (r_{ij} - w_i^T u_j)(-w_i) = 0 \rightsquigarrow u_j = \left(\sum_{i \in \Omega_j} w_i w_i^T \right)^{-1} \sum_{i \in \Omega_j} r_{ij} w_i$$

- 2-way dependency
  solution for w depends on u and reverse
  start with random u/w and apply equations (alternating least squares)
- FAQ
  Loss is symmetric, so w loss = u loss

Expanding Model

- Regression
  $\hat{y} = mx + b$
- Bias terms
  user vector
  user bias
  $$\hat{r}_{ij} = w_i^T u_j + b_i + c_j + \mu$$
  - global avg.
  predicted rating
  movie vector
  movie bias

  μ = centering the dataset with mean 0
  b_i / c_j = biases
- Movie Bias
  great movie → everyone likes it; attributes are not enough
  Avatar vs SciFi shit
- Training
  objective: $J = \sum_{i,j \in \Omega} (r_{ij} - \hat{r}_{ij})^2$

  $$\hat{r}_{ij} = w_i^T u_j + b_i + c_j + \mu$$

$$\frac{\partial J}{\partial w_i} \overset{!}{=} 0 \rightsquigarrow w_i = \left(\sum_{j \in \psi_i} u_j u_j^T \right)^{-1} \sum_{j \in \psi_i} (r_{ij} - b_i - c_j - \mu) u_j$$

symmetric

$$\frac{\partial J}{\partial u_j} \overset{!}{=} 0 \rightsquigarrow u_j = \left(\sum_{i \in \Omega_j} w_i w_i^T \right)^{-1} \sum_{i \in \Omega_j} (r_{ij} - b_i - c_j - \mu) w_i$$

$$\frac{\partial J}{\partial b_i} \overset{!}{=} 0 \rightsquigarrow b_i = \frac{1}{|\psi_i|} \sum_{j \in \psi_i} (r_{ij} - w_i^T u_j - c_j - \mu)$$

$$\frac{\partial J}{\partial c_j} \overset{!}{=} 0 \rightsquigarrow c_j = \frac{1}{|\Omega_j|} \cdot \sum_{i \in \Omega_j} (r_{ij} - w_i^T u_j - b_i - \mu)$$

Regularization

prevents overfitting

– For linear regression

Model: $\hat{y} = w^T x$

MSE — square magnitude large weights are penalised

Objective: $J = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \|w\|_2^2$

Solution: $w = (\lambda I + X^T X)^{-1} X^T y$

$\| * \|_F$ = Frobenius norm $\qquad \|W\|_F^2 = \sum_{i=1}^{N} \sum_{k=1}^{K} |w_{ik}|^2 = \sum_{i=1}^{N} \|w_i\|_2^2 = \sum_{i=1}^{N} w_i^T w_i$

$$J = \sum_{i,j \in \Omega} (r_{ij} - \hat{r}_{ij})^2 + \lambda \left(\|W\|_F^2 + \|U\|_F^2 + \|b\|_2^2 + \|c\|_2^2 \right)$$

$$\frac{\partial J}{\partial w_i} \overset{!}{=} 0 \rightsquigarrow w_i = \left(\sum_{j \in \Psi_i} u_j u_j^T + \lambda I \right)^{-1} \sum_{j \in \Psi_i} (r_{ij} - b_i - c_j - \mu) u_j$$

$$u_j = \left(\sum_{i \in \Omega_j} w_i w_i^T + \lambda I \right)^{-1} \sum_{i \in \Omega_j} (r_{ij} - b_i - c_j - \mu) w_i$$

$$b_i = \frac{1}{|\Psi_i| (1+\lambda)} \sum_{j \in \Psi_i} (r_{ij} - w_i^T u_j - c_j - \mu)$$

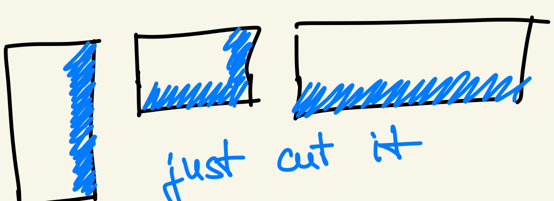
$$c_j = \frac{1}{|\Omega_j| (1+\lambda)} \sum_{i \in \Omega_j} (r_{ij} - w_i^T u_j - b_i - \mu)$$

SVD

X can be decomposed into U, S, V

$X_{N \times M} = U_{N \times M} \cdot S_{M \times M} \cdot V^T_{M \times M}$

truncated SVD → U, S, V ordered by importance



$U_{N \times K} \cdot S_{K \times K} \cdot V^T_{M \times K} \approx X$ (approximation)

$$\hat{x}_{ij} = \sum_k u_{ik} s_{kk} v_{kj} = \sum (u_{ik} \cdot s_{kk}) v_{kj} = \sum_k u'_{ik} v_{kj} = u_i'^T v_j$$

Probabilistic Matrix Factorization

$$\underline{R} \sim N(\underbrace{W U^T}_{\text{mean}}, \sigma^2)$$

$$r_{ij} \sim N(w_i^T u_j, \sigma^2)$$

- Maximum likelihood estimation

$$L = \prod_{ij \in \Omega} \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{1}{2\sigma^2}(r_{ij} - w_i^T u_j)^2\right)$$

$$W, U = \arg\max_{W,U} L$$

$$\log(L) = C - \sum_{i,j \in \Omega} \frac{1}{2\sigma^2}(r_{ij} - w_i^T u_j)^2$$

- MAP Estimation

MLE : $p(R \mid W, U)$

MLE : $p(W, U \mid R)$ a-posteriori

$$p(W, U \mid R) = \frac{p(R \mid W, U)\, p(W)\, p(U)}{p(R)}$$

$$p(W) = N(0, \lambda^{-1}) \qquad p(U) = N(0, \lambda^{-1})$$

precision parameter

normal, centered around zero

- MAP objective

$$L = \prod_{i,j \in \Omega} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(r_{ij} - w_i^T u_j)^2\right) \times \frac{\lambda}{\sqrt{2\pi}} e^{-\frac{\lambda}{2}\|W\|_F^2} \cdot \frac{\lambda}{\sqrt{2\pi}} e^{-\frac{\lambda}{2}\|U\|_F^2}$$

Bayesian Matrix Factorization

everything is random variable

- Embeddings

$$\text{cat} \rightarrow \begin{pmatrix} 0,5 \\ -0,2 \\ 1,3 \end{pmatrix}$$

king - men ≈ queen - woman

N word vocabulary $\underset{N \times K}{W[i]}$ = single word

K = vector size

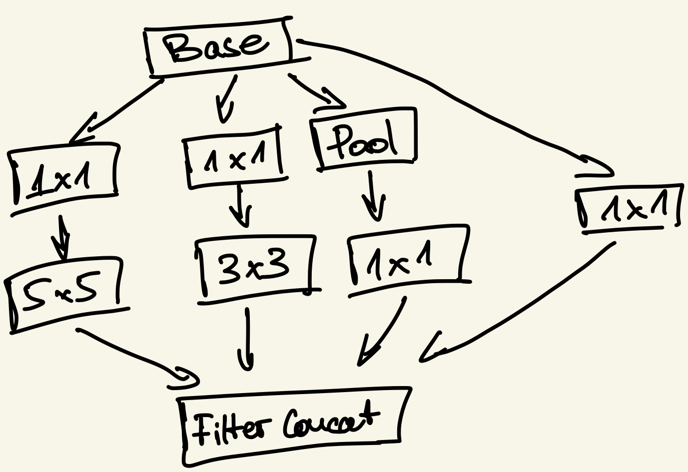
- In keras
  2 Embedding Layers
  emb_u = Embedding (N, K)
  emb_m = Embedding (M, K)

  2 inputs → user + movie

dot (Embed user, Embed movie)

- Bias Terms
  keras is not low-level library
  (layer level lib)
  Embedding (N, 1) → is scalar (user Bias)
  (M, 1) → movie bias
  actual rating − μ (full dataset mean)

Residual Learning (Vision Inspired)

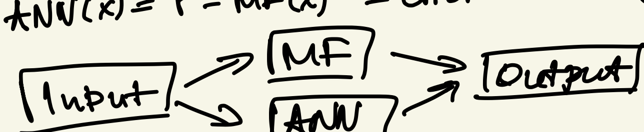


Different Branches, summed together at the end to detect different patterns

- MF performs well, but is linear
- NN can find nonlinear patterns

$\hat{r} = MF(x) + ANN(x)$ Residual (can be non-linear)

$ANN(x) = \hat{r} - MF(x)$ = error made by MF model



NN will try to predict the error made by MF

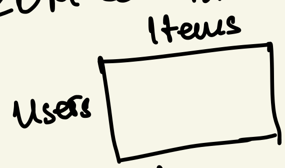
/non linear pattern may be absent → not learnable from the residuals

Autoencoders (AutoRec)

- Feed forward NN that predicts it's own input
  Error = (output − input)^2
  general NN: ann.fit(X, Y)
  autoencoder: ann.fit(X, X)
- Denoising Autoencoders
  reconstructing image with blackened parts
- Recommenders → fill missing ratings
- Structure of data in non-recomenders
  N x D (# samples, # features)

N = images
D = pixel values in each images

- Ratings Matrix: 20M cells in the matrix are filled in



N = number of samples (users)
M = number of features (movies)
Complexity → loops over 130k users (not 20M individual ratings)

- New Addition
  add noise to input (Dropout layer to the front of NN)
  not simply reconstruct the matrix
- Sparse Matrix (N x M too big)
  lil / csr / coo matrices
  Keras does not recognise sparse matrix → convert to dense arrays
  Missing values (0) have to be skipped
- Test MSE
  model.predict(x_test) / y_test → normal
  ignore keras val-loss, because it can copy input to output
  train ratings should predict test ratings
  where test values exist

predictions are entries in N x M, where test values exist

$$J_{test} = \frac{1}{|\Omega_{test}|} \sum_{i=1}^{N} \sum_{j=1}^{M} m_{ij}^{(test)} (r_{ij} - \hat{r}_{ij})^2$$

$m_{ij}^{(test)} = 1$ if $(i,j) \in \Omega_{test}$ else 0
$\hat{r}_{ij}$ = reconstruction from training data