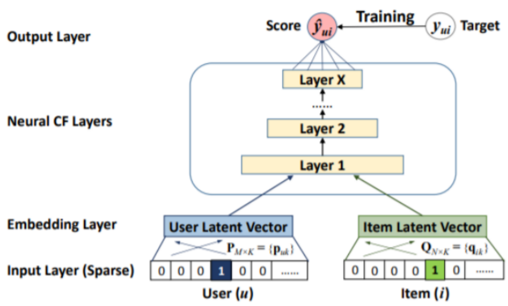
🕶️

NeuMF 논문 정리

Neural Collaborative Filtering

- **General Framework**



  - Input layer : user와 item의 one-hot encoding vector
  - Embedding layer
    - input 단계의 sparse 벡터를 dense vector로 mapping
    - Fully connected neural netwokr를 이용 → 가중치 행렬 **P**
    - **P** 행렬의 각 row는 각 사용자를 표현하는 저차원의 dense vector → `user latent vector`
  - Neural CF layer
    - `user latent vector` 와 `item latent vector` 을 연결한 벡터를 인풋으로 DNN 통과함
    - 각 벡터

      user latent vector = $P^T v_u^U$ (이때, v_u^U는 user u를 나타내는 one-hot 벡터)

      item latent vector = $Q^T v_i^I$ (이때, v_i^I는 item i를 나타내는 one-hot 벡터)

      deep neural network = $\phi_X(\ldots\phi_2(\phi_1(P^T v_u^U, Q^T v_i^I))\ldots)$, $where\ \ \phi_x(.) = x$번째 neural network

  - Output layer

        - 유저 u와 아이템 i가 얼마나 관련되어 있는지 출력 (0과 1 사이)
        - logistic 함수나 probit 함수 사용

$$\hat{y}_{u,i} = f(P^T v_u^U, Q^T v_i^I | P, Q, \Theta_f) = \phi_{out}(\phi_X(\ldots\phi_2(\phi_1(P^T v_u^U, Q^T v_i^I))\ldots)), \quad 0 \leqq \hat{y}_{u,i} \leqq 1$$

    - Learning NCF
        - loss function을 정의해야 함
        - NCF는 0과 1 사이의 예측값을 갖고, 데이터는 0 또는 1의 이진 데이터로 이루어져 있음

          → 이런 분포를 모델링 할 때는 bernoulli distribution 이용
        - likelihood function

$$p(\mathcal{Y}, \mathcal{Y}^- | P, Q, \Theta_f) = \prod_{(u,i)\in\mathcal{Y}} \hat{y}_{u,i}{}^{y_{u,i}} \prod_{(u,j)\in\mathcal{Y}^-} (1 - \hat{y}_{u,j})^{1-y_{u,i}}$$

        - loss function

$$\begin{aligned} L &= -log\, p(\mathcal{Y}, \mathcal{Y}^- | P, Q, \Theta_f) \\ &= -\sum_{(u,i)\in\mathcal{Y}} y_{u,i}\, log\, \hat{y}_{u,i} - (\sum_{(u,j)\ in \mathcal{Y}^-} (1 - y_{u,i})\, log\, (1 - \hat{y}_{u,j})) \\ &= -(\sum_{(u,i)\in\mathcal{Y}\cup\mathcal{Y}^-} (y_{u,i}\, log\, \hat{y}_{u,i} + (1 - y_{u,i})\, log\, (1 - \hat{y}_{u,i}))) \end{aligned}$$

Generalized Matrix Factorization (GMF)

- 예측값을 다음과 같이 파라미터라이징한다면 MF와 같음

$$\begin{aligned} \hat{y}_{u,i} &= a_{out}(h^T \phi_1(P^T v_u^U, Q^T v_i^I)) \\ where\ \ a_{out} &= \text{identity function} \\ h^T &= [1, \ldots, 1]_{\text{1 x k}} \\ \phi_1 &= P^T v_u^U \odot Q^T v_i^I \end{aligned}$$

- 이때 aout과 hT를 아래와 같이 두어 MF를 일반화함

$$a_{out} = 1/(1 + e^{-x}),\ \ h^T = [h_1, \ldots, h_k]$$

→ hT에 non uniform 값을 주어 내적할 때, 각 term에 다른 가중치를 둘 수 있도록 함

→ latent vector의 중요도를 조절

Multi-layer Perceptron (MLP)

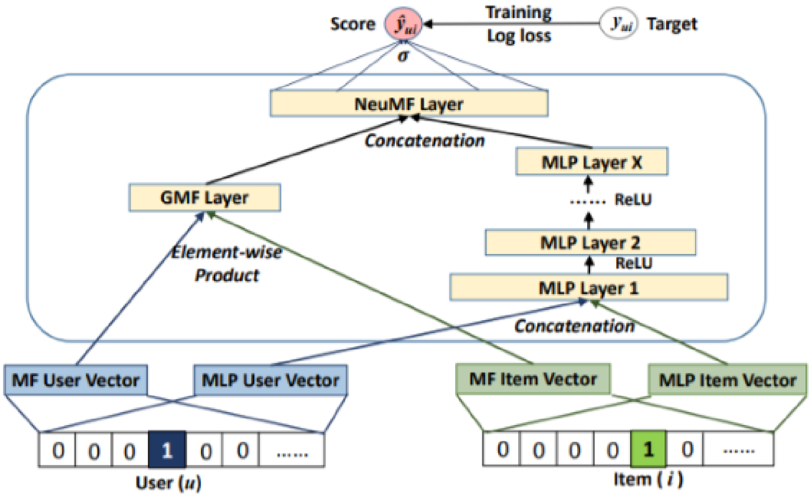
- 유저와 아이템 간 복잡한 관계를 표현하기 위해 MLP의 non-linearity 특성을 이용
- 식

$$\begin{aligned}
z_1 &= \phi_1(P^T v_u^U, Q^T v_i^I) = \begin{bmatrix} P^T v_u^U \\ Q^T v_i^I \end{bmatrix} \\
z_2 &= \phi_2(z_1) = a_2(W_2^T z_1 + b_2) \\
&\cdots \\
z_L &= \phi_L(z_{L-1}) = a_L(W_L^T z_{L-1} + b_L) \\
\hat{y}_{u,i} &= \sigma(h^T \phi_L(z_{L-1}))
\end{aligned}$$

  - `psi 1` 유저와 아이템의 latent vector을 연결하는 함수
  - `psi l` neural net 함수 (W, b는 각각 가중치 행렬과 편향 벡터)
  - 마지막 식은 GMF 구조와 동일

Fusion of GMF and MLP

- 모델 구조



- 특징
  - 모델별로 서로 다른 embedding layer을 사용함
  - 식

$$\phi^{GMF} = p_u^G \odot q_i^G$$

$$\phi^{MLP} = a_L(W_L^T(a_{L-1}(\dots a_2(W_2^T \begin{bmatrix} p_u^M \\ q_i^M \end{bmatrix} + b_2)\dots)) + b_L)$$

$$\hat{y}_{u,i} = \sigma(h^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix})$$

  - user-item 간 상호 관계를 표현하기 위해 MF의 linearity와 MLP의 non-linearity 결합

    $\Rightarrow$ `Neural matrix factorization`