Multilayer Perceptron based Recommender System

Jongjin Lee

Presented by Jongjin Lee

2017.11.22

Overview

Introduction: MLP

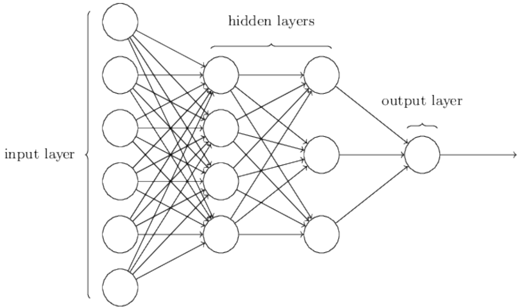

Figure: Multilayer Perceptron

Introduction: MLP

- Concise but effective model.
- Widely used in many area
- Approximate any measurable function to any desired degree of accuracy.

Introduction : Recommender System

Recommendation models are mainly categorized into...

- Collaborative Filtering Recommender System

  $\rightarrow$ By learning from user - item historical interactions
- Content-based Recommender System

  $\rightarrow$ Using item's and user's auxiliary information.
- Hybrid Recommender System

Capture the non-linear relationships between users and item by MLP

Neural Collaborative Filtering

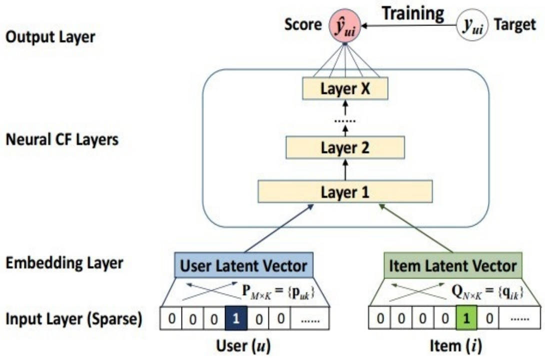

Figure: Neural Collaborative Filtering

Neural Collaborative Filtering

- $Y = \{y_{ui}\}_{(u=1,\ldots,M,i=1,\ldots,N)}$, $(R = \{r_{ui}\}_{(u=1,\ldots,M,i=1,\ldots,N)})$
  ,User-Item Rating Matrix
- s_u^{user}, One-hot identifier of user u, $_{(u=1,\ldots,M)}$
- s_i^{item}, One-hot identifier of item i, $_{(i=1,\ldots,N)}$
- $P \in R^{M \times K}$, $Q \in R^{N \times K}$
  ,the latent factor matrix for users and items, respectively

Learning from implicit feedback

- User-Item interaction : explicit feedback vs implicit feedback
- Define interaction(rating) matrix $Y = \{y_{ui}\}_{(u=1,\dots,M,i=1,\dots,N)}$

$$y_{ui} = \begin{cases} 1, & \text{if interaction (user u, item i) is observed;} \\ 0, & \text{otherwise.} \end{cases}$$

- Implicit data provides only noisy signals
- Scarcity of negative feedback
- The problem of estimating the scores of unobserved entries Y

Matrix Factorization

- Predict rating matrix by introducing the latent vector for user and item.
- Let $p_u = P^T s_u^{user} \in R^K$, and $q_i = Q^T s_i^{item} \in R^K$ denote the latent vector for user u and item i
- MF estimates an interaction y_{ui} as the inner product of u_u, and v_i

$$\hat{y}_{ui} = f(u, i \mid p_u, q_i) = p_u^T q_i$$

- Matrix Factorization have limitations

Neural Collaborative Filtering

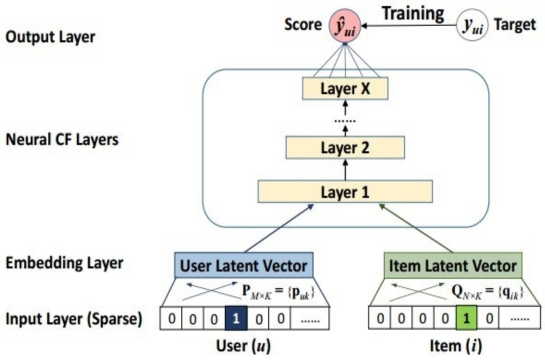

Figure: Neural Collaborative Filtering

Neural Collaborative Filtering

- The prediction rule of NCF is formulated as follows:

$$\hat{y_{ui}} = f(U^T \cdot s_u^{user}, V^T \cdot s_i^{item} \mid U, V, \theta)$$

f($\cdot$): multilayer perceptron, θ: parameters of this network.

- The (general) loss function is

$$L = \sum_{(u,i) \in O \cup O^-} w_{ui}(y_{ui} - \hat{y}_{ui})^2$$

O: observed interaction,
O^-: all(or sampled from) unobserved interactions
w_{ui}: weight(a hyperparameter)

Neural Collaborative Filtering

- For implicit feedback, Use Logistic or Probit function
- The loss function is

$$L = \sum_{(u,i)\in O\cup O^-} y_{ui} log\, \hat{y}_{ui} + (1 - y_{ui}) log(1 - \hat{y}_{ui})$$

  O: observed interaction,
  O^-: all(or sampled from) unobserved interactions
- O^- : uniformly sample them from unobserved interactions.
- Minimize L by Stochastic gradient desent(SGD)

Fusion of GMF and NCF

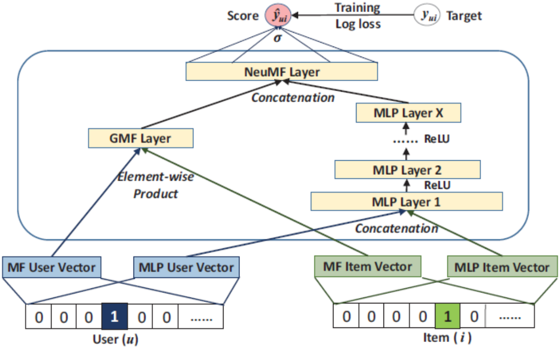

Figure: Neural matrix facorization model

- Generalized Matrix Factorization(GMF):
  $\hat{y}_{GMF} = f_{out}(w^T(\hat{y}_{MF})) = f_{out}(w^T(p_u^T q_i))$

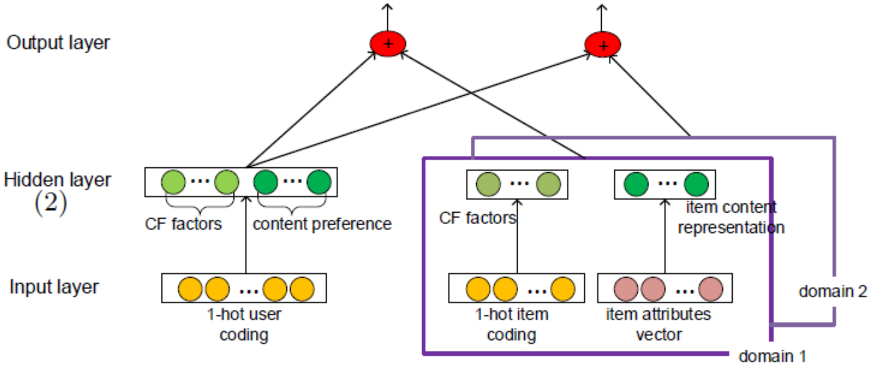

Figure: CCCFNet

CCCFNet

- Cross-domain Content-boosted Collaborative Filtering neural Network.
- Combine collaborative filtering and content filtering.
- Introduce multiple-domain, use knowledge from relatively dense auxiliary domains to overcome sparsity problem.
- Consistently, outperforms several baseline method.

CCCFNet

- Suppose we have two domains: Target (1) & Auxiliary(2)
- $Y^{(1)}_{N \times M^{(1)}}, Y^{(2)}_{N \times M^{(2)}}$, Rating matrix
- $A^{(1)}_{M^{(1)} \times T^{(1)}}, A^{(2)}_{M^{(2)} \times T^{(2)}}$, Content matrix (with $T^{(1)}$, $T^{(2)}$ attribute)
- $P_{N \times K}, {Q^{(1)}}_{M^{(1)} \times K}, {Q^{(2)}}_{M^{(2)} \times K}$, Latent factor matrix for users and items
- $U_{N \times L}, {V^{(1)}}_{T^{(1)} \times L}, {V^{(2)}}_{T^{(2)} \times L}$, Latent factor matrix for content preference, item content
- $\theta = (P, U, Q^{(1)}, Q^{(2)}, V^{(1)}, V^{(2)})$

CCCFNet

- Minimize the followin equation:

$$L = \frac{1}{2}\sum_{u,i}\left({y^{(1)}}_{ui} - p_u \centerdot {q_i}^{(1)} - u_u \centerdot {v_i}^{(1)}\right)^2 + \frac{\lambda_1}{2}\sum_{u,i}\left({y^{(2)}}_{ui} - p_u \centerdot {q_i}^{(2)} - u_u \centerdot {v_i}^{(2)}\right)^2 + \frac{\lambda_*}{2}\sum_{k}(\|\theta\|)^2$$

- $p_u = P^T s_u^{user}, q_i = Q^T s_u^{item}, u_u = U^T s_u^{user}, v_i = row(A, i)V^T$

CCCFNet

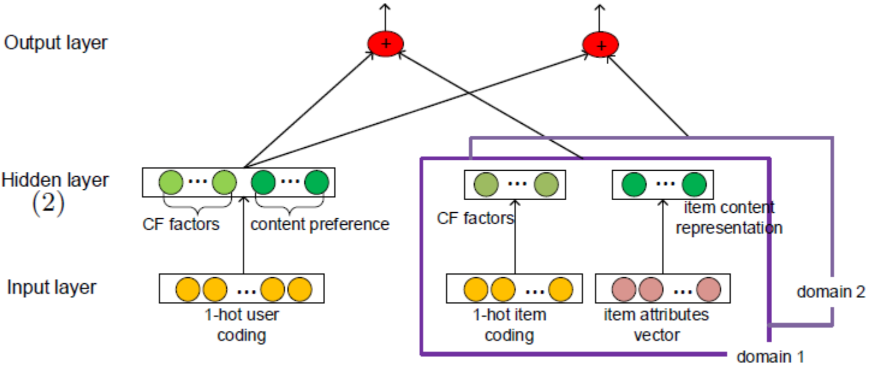

Figure: CCCFNet

Wide & Deep Learning

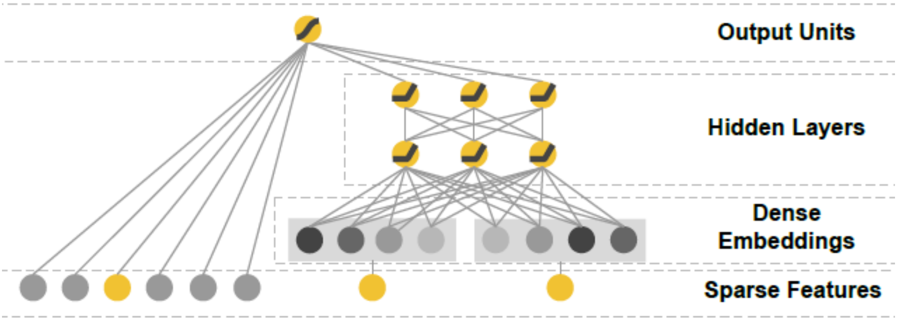

Figure: Wide & Deep Learning

Wide & Deep Learning

- Wide Learning Components
  - Single layer Perceptron
  - Memorization
    $\rightarrow$ Catching the direct features from historical data
    $\rightarrow$ Loosely defined: Learning the frequent co-occurrence of items or features, and exploiting the correlation in the historical data
- Deep Learning Components
  - Multilayer Perceptron
  - Generalization
    $\rightarrow$ Catching the generalized by producing more general and abstract representations
    $\rightarrow$ Loosely defined: Learning the transitivity of correlation and explores new feature combinations that have never or rarely occurred in the past.

Wide & Deep Learning

- The wide learning part

$$y = W_{wide}^{T}(x, \phi(x)) + b$$

- $\phi(x)$ is cross-product transformation which is defined as:

$$\phi_k(x) = \prod_{i=1}^{d} x_i^{c_{ki}} \;, c_{ki} \in \{0, 1\}$$

$$c_{ki} = \begin{cases} 1, & \text{if i-th feature is part of the k-th transformation } \phi(x) \\ 0, & \text{otherwise.} \end{cases}$$

- Cross-product transformation is manually designed

Wide & Deep Learning

- The deep learning part

$$\alpha^{(l+1)} = f(W_{deep}^{(l)}\alpha^{(l)} + b)$$

- The wide & deep learning model is attained by fusing these two models:

$$P(\hat{y_{ui}} = 1 \mid x) = \sigma(W_{wide}^{T}\{x, \phi(x)\} + W_{deep}^{(T)}\alpha^{(lTf)} + bias)$$

Wide & Deep Learning

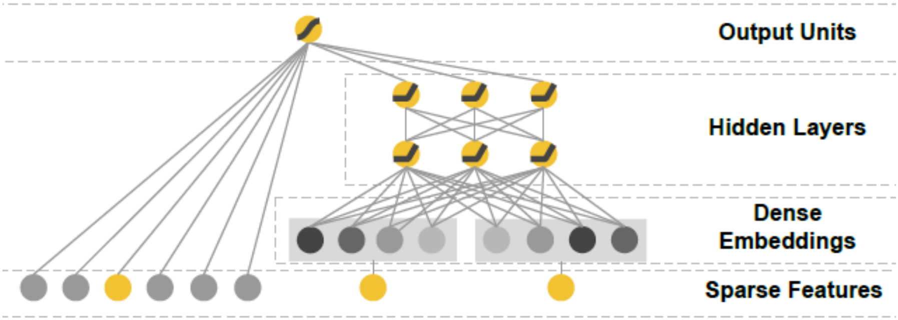

Figure: Wide & Deep Learning

DeepFM

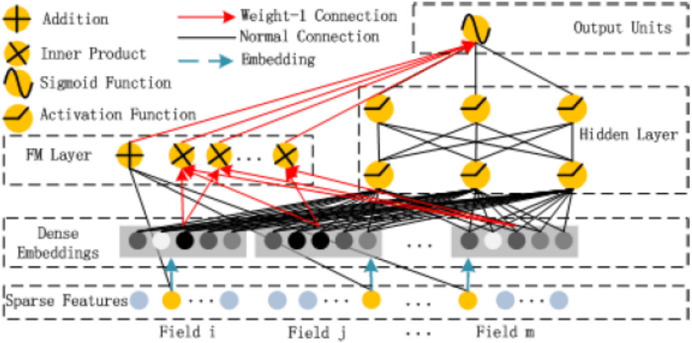

Figure: DeepFM

DeepFM

- Deep Factorization Machine
- Enlightend by Wide & Deep Learning
- Integrates Factorization Machine and MLP
- Capture the high-order feature interactions via deep neural network.
- Capture the low-order feature interactions via factorization machine.
- Does not require tedious feature engineering.

DeepFM

- x is an m-fields data consisting of pairs(u,i), and d-dimensional vector
- x may include categorical fields(gender, location...) and continuous fields (age...)
- x $= [x_{field_1}, x_{field_2}, \cdots, x_{field_m}]$ is d-dimensional vector.
- $\hat{y} = \sigma(y_{FM}(x), y_{MLP}(x))$
- FM component and Deep component, that share the same input.

DeepFM

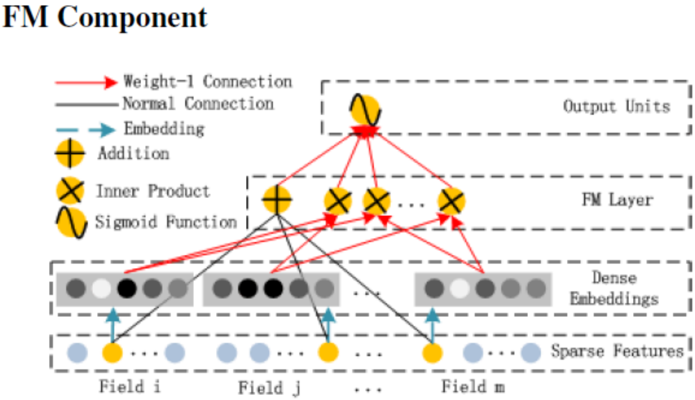

Figure: FM Component

DeepFM

- It can capture order-2 feature interarctions much more effectively than previous approach.(especially the dataset is sparse)

$$\hat{y}_{FM} = < w, x > + \sum_{i=1}^{m} \sum_{j=i+1}^{m} < V_i, V_j > x_i \cdot x_j$$

$$, w \in R^d \textit{and } V_i \in R^k$$

- $< w, x >$ reflects order-1 feature interactions, and summation term reflects order-2 feature interactions.

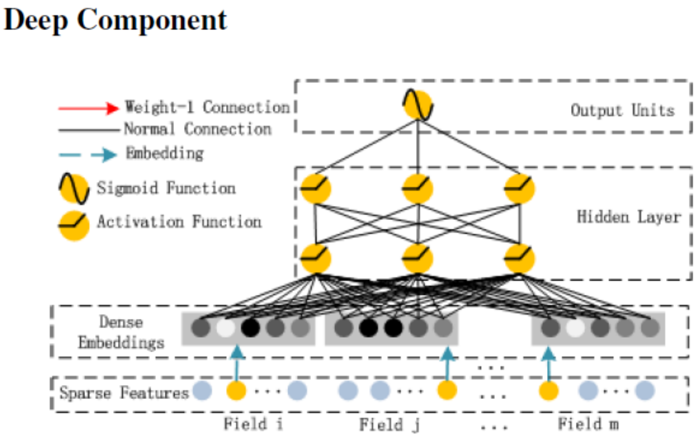

Figure: Deep Component

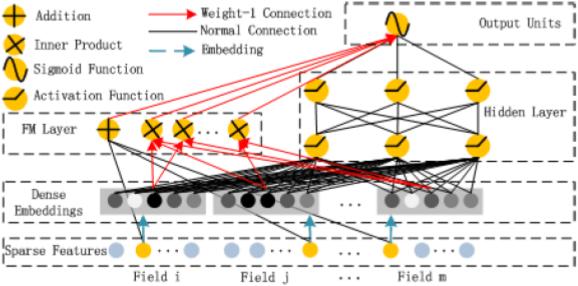

Figure: DeepFM

$$\hat{y} = \sigma(< w, x > + \sum_{i=1}^{m} \sum_{j=i+1}^{m} < V_i, V_j > x_i \cdot x_j + W_{deep}^{(T)} \alpha^{(l_f)} + bias)$$

The End