Latent Factor Model I

STAT3009 Recommender Systems

by **Ben Dai** (CUHK-STAT)
on **August 13, 2022**

» Recall: RS

* **Training dataset:** [**userID**, **itemID**, **rating**]
* **Testing dataset:** [**userID**, **itemID**, **?**]
* **Evaluation:** Given a testing index set Ω^{te} (set of user-item pairs we want to predict),

$$RMSE = \Big(\frac{1}{|\Omega^{\text{te}}|}\sum_{(u,i)\in\Omega^{\text{te}}}(\hat{r}_{ui}-r_{ui})^2\Big)^{1/2}.$$

* **Goal:** Find predicted ratings $(\hat{r}_{ui})_{(u,i)\in\Omega^{\text{te}}}$ such that **minimizes RMSE**
* **Baseline methods:** Global-average, user-average, item-average, user-item average
* **Correlation-based RS:** User-correlation-based RS, item-correlation-based RS, and correlation-based + baseline

» Machine learning (ML): RS

Using ML methods to build RS:

- Step 1. Introduce a model with some **parameters**
- Step 2. Estimate the **parameters** by minimizing (maximizing) the **Evaluation Loss** in **Training Set**
- Step 3. Use the estimated model to **predict**
- Idea Learning from Data: A model works well in **Training Set**, tend to work well in **Testing Set**

» Machine learning (ML): RS

⚠ MATH

* Training dataset $(\text{feat}_i, \text{label}_i)_{i=1}^n$
* Testing dataset $(\text{feat}_j)_{j=1}^m$:

Step 1. Introduce a model with some **parameters**: f_θ

Step 2. Estimate the **parameters** by minimizing (maximizing) the **Evaluation Loss** in **Training Set**

$$\widehat{f}_\theta = \underset{f_\theta}{\text{argmin}} \ \frac{1}{n}\sum_{i=1}^n L(\text{label}_i, f_\theta(\text{feat}_i)) + \lambda \text{Reg}(f_\theta).$$

Step 3. Use the estimated model to **predict**

$$\widehat{\text{label}}_j = \widehat{f}_\theta(\text{feat}_j).$$

» Components in ML

Data (feat, label) is a pair of input features and its outcome

Model f_θ: a parameterized function to map features to label

Loss $L(\cdot, \cdot)$: The measure of how good the predicted outcome compared with the true outcome

Reg $\mathrm{Reg}(f_\theta)$: regularization term in ERM to prevent overfitting

Opt The algorithm for solving the problem

» Rethink baseline methods: RS

Can we use this learning paradigm to find the best parameters for baseline models?

* **Step 1.** Introduce *a* method with some **parameters**

method	math formula	**parameters**
Global Average	$\hat{r}_{ui} = \mu_0$	μ_0
User Average	$\hat{r}_{ui} = a_u$	$\boldsymbol{a} = (a_1, \cdots, a_n)^\intercal$
Item Average	$\hat{r}_{ui} = b_i$	$\boldsymbol{b} = (b_1, \cdots, b_m)^\intercal$
User-Item Average	$\hat{r}_{ui} = \mu_0 + a_u + b_i$	$\mu_0, \boldsymbol{a}, \boldsymbol{b}$

» Rethink baseline methods: RS

- **Step 2.** Estimate the **parameters** by minimizing **RMSE**
  - Global Average.

$$\min_{\mu_0 \in \mathbb{R}} \left(\frac{1}{|\Omega|} \sum_{(u,i) \in \Omega} (\mu_0 - r_{ui})^2 \right)^{1/2} \iff \min_{\mu_0 \in \mathbb{R}} \sum_{(u,i) \in \Omega} (\mu_0 - r_{ui})^2$$

» Rethink baseline methods: RS

- **Step 2.** Estimate the **parameters** by minimizing **RMSE**
  - Global Average.

$$\min_{\mu_0 \in \mathbb{R}} \Big(\frac{1}{|\Omega|} \sum_{(u,i)\in\Omega} (\mu_0 - r_{ui})^2 \Big)^{1/2} \iff \min_{\mu_0 \in \mathbb{R}} \sum_{(u,i)\in\Omega} (\mu_0 - r_{ui})^2$$

Taking the derivative to μ_0:

$$2 \sum_{(u,i)\in\Omega} (\mu_0 - r_{ui}) = 0, \implies \mu_0 = \frac{1}{|\Omega|} \sum_{(u,i)\in\Omega} r_{ui} = \bar{r}$$

» Rethink baseline methods: RS

* **Step 2.** Estimate the **parameters** by minimizing **RMSE**
  * User Average.

$$\min_{\boldsymbol{a}\in\mathbb{R}^n} \sum_{(u,i)\in\Omega} (a_u - r_{ui})^2 \iff \min_{\boldsymbol{a}\in\mathbb{R}^n} \sum_{u=1}^{n} \sum_{i\in\mathcal{I}_u} (a_u - r_{ui})^2$$

  * The loss function is **separable**, thus it suffices to consider user-wise minimization:

$$\begin{aligned} a_u &= \operatorname*{argmin}_{a_u\in\mathbb{R}} \sum_{i\in\mathcal{I}_u} (a_u - r_{ui})^2 \\ &= \frac{1}{|\mathcal{I}_u|} \sum_{i\in\mathcal{I}_u} r_{ui} = \bar{r}_u, \quad \text{for } u = 1, \cdots, n \end{aligned}$$

» Rethink baseline methods: RS

* **Step 2.** Estimate the **parameters** by minimizing **RMSE**
  * Item Average.

$$\min_{\boldsymbol{b}\in\mathbb{R}^m} \sum_{(u,i)\in\Omega} (b_i - r_{ui})^2 \iff \min_{\boldsymbol{b}\in\mathbb{R}^m} \sum_{i=1}^{m} \sum_{u\in\mathcal{U}_i} (b_i - r_{ui})^2$$

  * The loss function is **separable**, thus it suffices to consider item-wise minimization:

$$\begin{aligned} b_i &= \operatorname*{argmin}_{b_i\in\mathbb{R}} \sum_{u\in\mathcal{U}_i} (b_i - r_{ui})^2 \\ &= \frac{1}{|\mathcal{U}_i|} \sum_{u\in\mathcal{U}_i} r_{ui} = \bar{r}_i, \quad \text{for } i = 1, \cdots, m \end{aligned}$$

» Rethink baseline methods: RS

- **Step 2.** Estimate the **parameters** by minimizing **RMSE**
  - User-Item Average.

$$\min_{\boldsymbol{a}\in\mathbb{R}^n, \boldsymbol{b}\in\mathbb{R}^m} \sum_{(u,i)\in\Omega} (\mu_0 + a_u + b_i - r_{ui})^2$$

  - The loss is **non-separable**, taking the gradient w.r.t. $(\mu_0, \boldsymbol{a}, \boldsymbol{b})$ and eqaul to zeros

$$\sum_{(u,i)\in\Omega} (\mu_0 + a_u + b_i - r_{ui}) = 0$$

  - Specify μ_0, $\boldsymbol{a}_u$, and $\boldsymbol{b}_i$ sequentially.

» Prediction: Baseline Methods

- **Step 3.** Use the **estimated model** to do prediction
  - Global Average: $\hat{r}_{ui} = \hat{\mu}_0$
  - User Average: $\hat{r}_{ui} = \hat{a}_u$
  - Item Average: $\hat{r}_{ui} = \hat{b}_i$
  - User-Item Average: $\hat{r}_{ui} = \hat{\mu}_0 + \hat{a}_u + \hat{b}_i$

» Discussion: baseline methods

"All models are wrong, but some are useful." — George E. P. Box

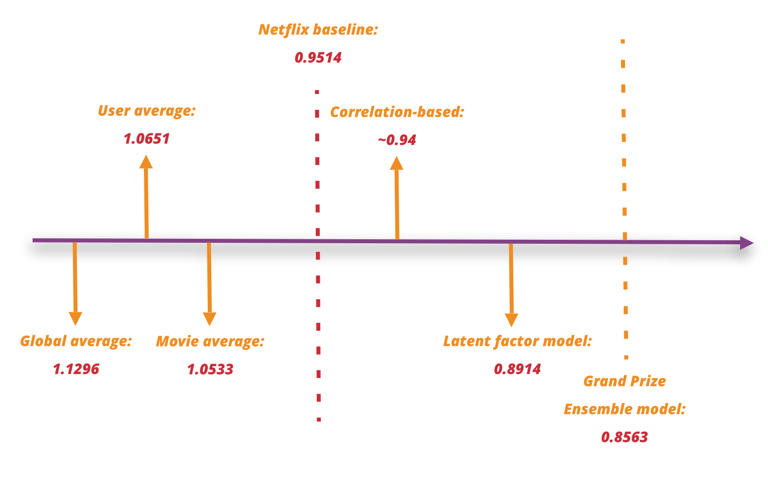
We need to figure out the **assumptions** for each method!

* **Global average** assumes that all users and items are essentially same
* **User average** assumes that a user has equal preference to all items
* **Item average** assumes that all users like "good" items
* **User-item average** assume that additive effects from users and items, **no interaction**

Example: I just don't like Action movies, even their ratings is quit high.

We need to model the user-item **interaction**.

» Motivation: Latent Factor Model



» Motivation: Latent Factor Model

* **Step 1.** Introduce *a* method with some **parameters** (latent factors)
    * Introduce K-length latent factors $\boldsymbol{p}_u$ for the user u:

$$\boldsymbol{p}_u = (p_{u1}, \cdots, p_{uK})^\top$$

    * Introduce K-length latent factors $\boldsymbol{q}_i$ for the item i:

$$\boldsymbol{q}_i = (q_{i1}, \cdots, q_{iK})^\top$$

    * Model the user-item **interaction** as **inner production**

$$\hat{r}_{ui} = \boldsymbol{p}_u^\top \boldsymbol{q}_i \to r_{ui}$$

* K is pre-specified number: **#Latent Factors**
* We tend to learn the **latent factors** from the data

» Loss: Latent Factor Model

- **Step 2.** Estimate the **parameters** by minimizing **RMSE**

$$\min_{\boldsymbol{P} \in \mathbb{R}^{n \times K}, \boldsymbol{Q} \in \mathbb{R}^{m \times K}} \frac{1}{|\Omega|} \sum_{(u,i) \in \Omega} (r_{ui} - \boldsymbol{p}_u^\intercal \boldsymbol{q}_i)^2 \tag{1}$$

- K is a pre-specified #Latent Factor, can **NOT** be solved by (3). In ML, we call it **tuning parameter** or **hyperparameter**.
- K increases $\Longrightarrow$ more **parameters** $\Longrightarrow$ lower **training loss**
- Lower training loss is always better? **NO!**

» Overfitting in ML: Latent Factor Model

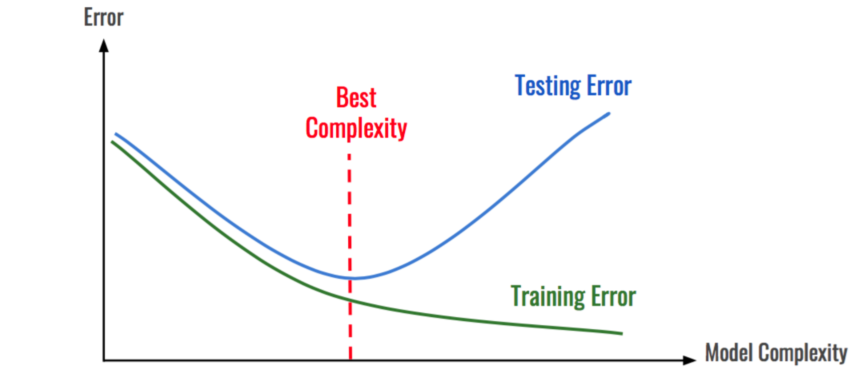
Source[1]

- **Overfitting**: fit the noise
- Too many **parameters** (**model complexity**) leads to **overfitting**

---

1 https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

» Overfitting in ML: Latent Factor Model



Source[2]

- **Complexity** too large $\Longrightarrow$ Low Training loss but high Testing loss

[2] https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

» Overfitting in ML: Latent Factor Model

	Low *Training* Error	High *Training* Error
Low *Testing* Error	The model is learning!	Probably some error in your code. Or you've created a *psychic* AI.
High *Testing* Error	OVERFITTING	The model is not learning.

Source[3]

[3] https://hackernoon.com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

» Tuning: Latent Factor Model

- **Q1:** How to **quantify** the Model **Complexity**:
  - **#Parameters**: $(n+m)K$
  - **Magnitude** of Parameters: $\sum_{u=1}^{n} \|\boldsymbol{p}_u\|_2^2, \sum_{i=1}^{m} \|\boldsymbol{q}_i\|_2^2$
- **A1:** Control (#Parameters by K, **Magnitude** by l_2-norm).
- **Regularized** Latent Factor Model:

$$\min_{\boldsymbol{P},\boldsymbol{Q}} \underbrace{\frac{1}{|\Omega|} \sum_{(u,i)\in\Omega} (r_{ui} - \boldsymbol{p}_u^\intercal \boldsymbol{q}_i)^2}_{\text{Training loss}} + \lambda \underbrace{\Big(\sum_{u=1}^{n} \|\boldsymbol{p}_u\|_2^2 + \sum_{i=1}^{m} \|\boldsymbol{q}_i\|_2^2 \Big)}_{\text{Params magnitude}} \quad (2)$$

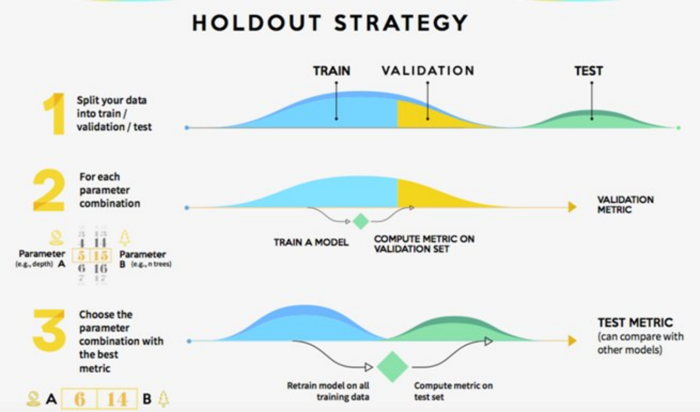
  where K and $\lambda > 0$ are **tuning** parameters to **balance** the model complexity and training loss.
- Why the later term can control the magnitude?

» Tuning: Latent Factor Model

* **Q2:** How to find the **best** tuning parameters (K, λ)
* **A2:** **Cross-validation** (CV) based on **Training/Validation** splitting

» Tuning: Latent Factor Model

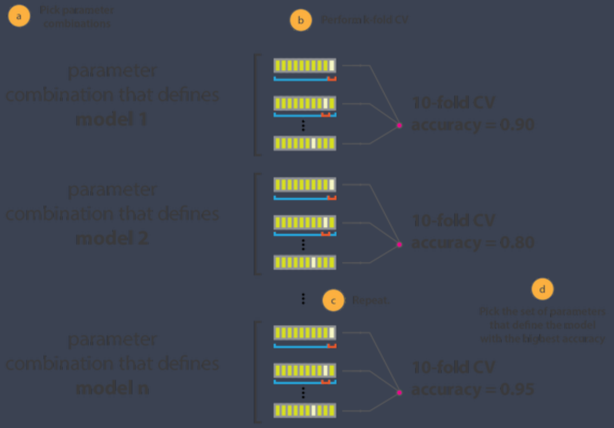
* **Q2:** How to find the **best** tuning parameters (K, λ)
* **A2:** **Cross-validation** (CV) based on **Training/Validation** splitting



Source[4]

» *k*-Fold Cross-Validation

* Typical splitting method: *k*-**fold** CV



Source[5]

[5] https://cambridgecoding.wordpress.com/2016/04/03/scanning-hyperspace-how-to-tune-machine-learning-models/

» Recall: Latent Factor Model

* **Step 2.** Estimate the **parameters** by minimizing **RMSE**

$$\min_{\boldsymbol{P},\boldsymbol{Q}} \frac{1}{|\Omega|} \sum_{(u,i)\in\Omega} (r_{ui} - \boldsymbol{p}_u^\intercal \boldsymbol{q}_i)^2 + \lambda \Big(\sum_{u=1}^{n} \|\boldsymbol{p}_u\|_2^2 + \sum_{i=1}^{m} \|\boldsymbol{q}_i\|_2^2 \Big)$$

* Using **Cross-Validation** to determine the optimal **tuning parameters** (K, λ), denote as K^* and λ^*
* **Refit** the model based on **full training data** with K^* and λ^*.
* The final estimator is denoted as $(\hat{\boldsymbol{p}}_u)_{u=1}^n$ and $(\hat{\boldsymbol{q}}_i)_{i=1}^m$

» Prediction: Latent Factor Model

* **Step 3.** Using the **estimated model** with the best tuning parameters to do prediction

$$\hat{r}_{ui} = \hat{\boldsymbol{p}}_u^\intercal \hat{\boldsymbol{q}}_i, \text{ for } (u, i) \in \Omega^{\text{te}}$$

» Summary: Latent Factor Model

- **Step 1.** Introduce *a* method with some **parameters** (latent factors)
  - Model the user-item **interaction** as **inner production**

$$\hat{r}_{ui} = \boldsymbol{p}_u^\intercal \boldsymbol{q}_i \to r_{ui}$$

- **Step 2.** Estimate the **parameters** by minimizing **RMSE**

$$\min_{\boldsymbol{P},\boldsymbol{Q}} \frac{1}{|\Omega|} \sum_{(u,i)\in\Omega} (r_{ui} - \boldsymbol{p}_u^\intercal \boldsymbol{q}_i)^2 + \lambda \Big(\sum_{u=1}^{n} \|\boldsymbol{p}_u\|_2^2 + \sum_{i=1}^{m} \|\boldsymbol{q}_i\|_2^2 \Big) \quad (3)$$

  - Using **Cross-Validation** to determine the optimal **tuning parameters** (K, λ), denote as K^* and λ^*
  - **Refit** the model based on **full training data** with K^* and λ^*.
- **Step 3.** Using the **estimated model** with the best tuning parameters to do prediction