Transduction Classification with Matrix Completion

By. **Miao Fan**
Ph.D. Candidate
Dept. of C.S.
Tsinghua University
fanmiao.cslt.thu@gmail.com

1. Introduction (Matrix completion)

- Netflix Problem(KDD Cup)

Ratings	Movie_1	Movie_2	…	Movie_n
User_1	4.5	??	??	5.2
User_2	2.1	??	??(prediction)	4.3
User_3	2.2	??	2.3	?
…	??	?	??	?
User_m	??	3.2	4.3	?

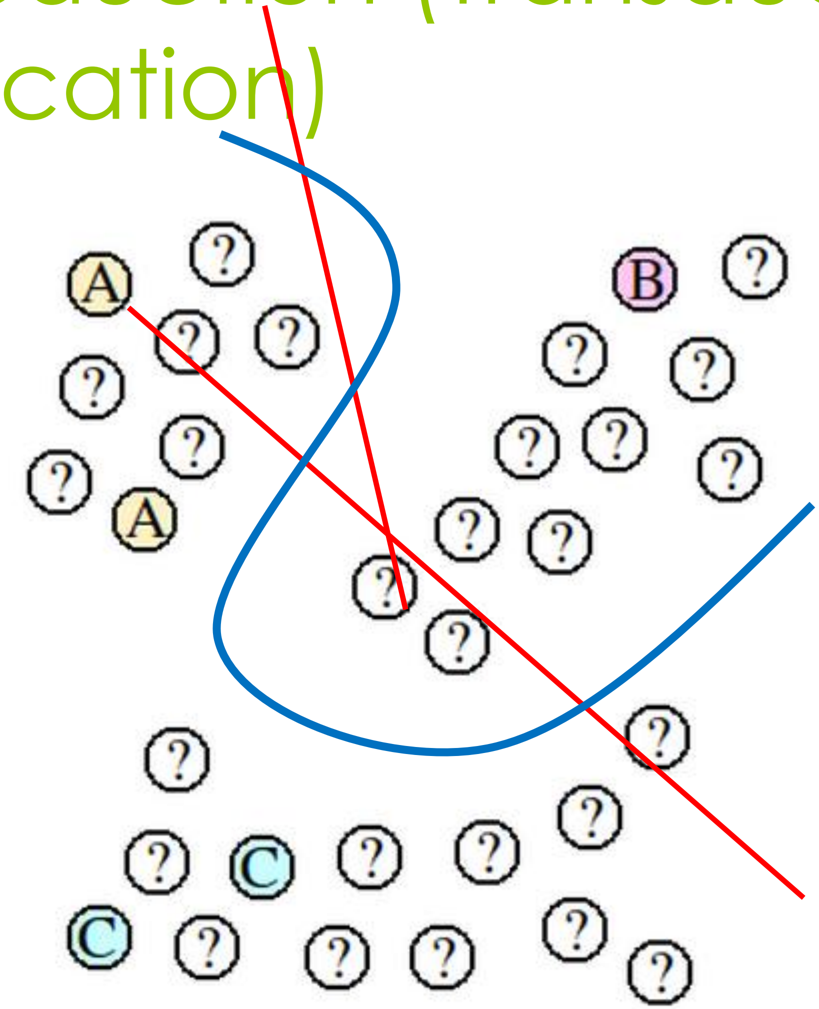
The key of recommender system: How to complete the matrix!!!!

1. Introduction(Transduction classification)

- **Transduction inference**
  - Reasoning from **observed, specific cases** to **specific cases**.

- **Induction inference**
  - Reasoning from **observed training cases** to **general rules (Then we may use these rules to predict test cases)**.
  - For example, the Netflix problem, we can also use Logistic regression model, however, for the sparsity, LR won't perform well.

1. Introduction (Transduction classification)



1. Introduction (Transduction classification)

- An example of learning which is not inductive would be in the case of **binary classification**, where the inputs tend to **cluster** in two groups. **A large set of test inputs may help in finding the clusters, thus providing useful information about the classification labels.**
- Let us think about the Netflix problem again, the number of **underlying factors** may be quiet less than the **observed feature dimension**. (Like PCA, low rank!!)

2. Transduction Classification with Matrix Completion

- The key assumption:
  - suppose that we have **m** items, for each item, it potentially have features with **d** dimensions and labels with **t** dimensions. We assume that the item-by-feature matrix **(X)** and **item-by-label matrix (Y)** are **jointly low rank (Z) $Z = AV^T$**. A and V are quiet low rank, compared with (LSA, PLSA, LDA) svd(Z) = [U, S, V].

	Feature_1	**…**	**Feature_d**
Item_1	0/1 or R		
Item_2			
Item_3			
…			
Item_n			

X

Label_1	**…**	**Label_t**
+1/-1 or 0/1		

Y

2. Transduction Classification with Matrix Completion

- Z = [X, Y];
  - $\underset{Z\, \in\, R^{n\,*(t+d)}}{argmin}\; Rank(Z)$
  - $s.t. \quad sign(z_{i+d,j}) = y_{i,j}\ ,\ \forall (i,j) \in \Omega_Y\ ;$
    $z_{i,j} = x_{i,j},\ \forall (i,j) \in \Omega_X$

- This formula is so hard, as rank() is a none-convex function! We use nuclear norm $||z||_*$ instead.

2. Transduction Classification with Matrix Completion

- We assume that **X** and **Y** are jointly produced by an **underlying low rank matrix**. We then take advantage of the **sparsity** to fill in the **missing labels and features** using a modified method of matrix completion.

2. Transduction Classification with Matrix Completion

- It starts from a n * d low rank “pre”-feature matrix X0, rank(X0) << min(d, n).
- The actual feature matrix X ($x\_ij \in R$)is obtained by adding **iid. Gaussian noise to the entries of X0**
- Y0 = WX0 + b,
- P(y_ij | y0_ij) = 1 / (1 + exp(-y_ij * y0_ij))

2. Transduction Classification with Matrix Completion

- $\underset{Z \in R^{n*(t+d)}}{argmin} Rank(Z)$
- $s.t. \quad sign(z_{i+d,j}) = y_{i,j}, \forall (i,j) \in \Omega_Y;$

$$z_{i,j} = x_{i,j}, \forall (i,j) \in \Omega_X$$

$$\underset{Z,b}{argmin} \quad \mu \|Z\|_* + \frac{\lambda}{\Omega_Y} \sum_{(i,j)\in\Omega_Y} c_y(z_{i+d,j} + b_i, y_{i,j}) + \frac{1}{\Omega_X} \sum_{(i,j)\in\Omega_X} c_x(z_{i,j}, x_{i,j})$$

2. Transduction Classification with Matrix Completion

Input: Initial matrix $\mathbf{Z}_0$, bias b_0,
    parameters μ, λ, Step sizes $\tau_{\mathrm{b}}, \tau_{\mathbf{Z}}$
Determine $\mu_1 > \mu_2 > \cdots > \mu_L = \mu > 0$.
Set $\mathbf{Z} = \mathbf{Z}_0, \mathrm{b} = \mathrm{b}_0$.
foreach $\mu = \mu_1, \mu_2, \ldots, \mu_L$ **do**
  while *Not converged* **do**
    Compute $\mathrm{b} = \mathrm{b} - \tau_{\mathrm{b}} g(\mathrm{b})$, $\mathbf{A} = \mathbf{Z} - \tau_{\mathbf{Z}} g(\mathbf{Z})$
    Compute SVD of $\mathbf{A} = \mathbf{U} \Lambda \mathbf{V}^\top$
    Compute $\mathbf{Z} = \mathbf{U} \max(\Lambda - \tau_{\mathbf{Z}} \mu, 0) \mathbf{V}^\top$
  end
end
Output: Recovered matrix $\mathbf{Z}$, bias b

Algorithm 1: FPC algorithm for MC-b.

Input: Initial matrix $\mathbf{Z}_0$,
    parameters μ, λ, Step sizes $\tau_{\mathbf{Z}}$
Determine $\mu_1 > \mu_2 > \cdots > \mu_L = \mu > 0$.
Set $\mathbf{Z} = \mathbf{Z}_0$.
foreach $\mu = \mu_1, \mu_2, \ldots, \mu_L$ **do**
  while *Not converged* **do**
    Compute $\mathbf{A} = \mathbf{Z} - \tau_{\mathbf{Z}} g(\mathbf{Z})$
    Compute SVD of $\mathbf{A} = \mathbf{U} \Lambda \mathbf{V}^\top$
    Compute $\mathbf{Z} = \mathbf{U} \max(\Lambda - \tau_{\mathbf{Z}} \mu, 0) \mathbf{V}^\top$
    Project $\mathbf{Z}$ to feasible region $z_{(t+d+1)\cdot} = \mathbf{1}^\top$
  end
end
Output: Recovered matrix $\mathbf{Z}$

Algorithm 2: FPC algorithm for MC-1.