Machine Discovery Homework 3

網媒所 R05944014 何文琦
化工系 B01504044 宋易霖
資工系 B04902072 陳威嘉

I、Abstract

In this assignment, given two rating matrices with some overlapped users/items whose mappings are unknown, we want to identify the unknown mapping between the users and items, further utilize the mappings to improve the recommendation. There are three different tasks in this assignment; the first one is corresponding in same domain, user and item. The second is matching in same domain but different user and item. The last one is mapping in different domain, user and item. We apply Latent space matching [1] to solve task 1, task 2, and use codebook [2] to solve task 3.

II、Algorithm

Method 1. Latent Space Matching (for task1, 2, 3)

According to [1]，given two observed rating matrices R_1 and R_2, we want to link the common users/items (rows/columns) between R_1 and R_2, then we can merge them into one single matrix. To find out the correspondence between R_1 and R_2, we solve the following equation, where G_{user} and G_{item} are 0-1 matrices that represent user and item correspondences.

$$\mathbf{R}_1 \approx \mathbf{G}_{\text{user}} \mathbf{R}_2 \mathbf{G}_{\text{item}}^T$$

First, we need to find singular value decomposition (SVD) of R_1 and R_2, however, both matrices are sparse matrix which have many values unknown, we can't use conventional methods to obtain the decomposition. Thus, we perform Matrix Factorization on R to obtain P and Q, and then transform (P_{MxK}, Q_{KxN}) to (U_{MxK}, D_{KxK}, V_{KxN}) by applying conventional SVD on P and Q, while K is the parameter that estimates the rank.

Matrix Factoriztion (by SGD):

We make $R = P * Q^T$, while all entries in P and Q is initialize within 0 and 1, and our goal is to minimize cost function:

$$L = \frac{1}{2}\left(\sum_{i=1}^{M}\sum_{j=1}^{N}\left(P \times Q^T - R_{ij}\right)^2 + \lambda_P |P|^2 + \lambda_Q |Q|^2\right) \quad [while \;\; Rij \;\; is \;\; observed]$$

And the gradient:

$$\frac{\partial L}{\partial P_i} = \left(PQ^T - R_{ij}\right)^2 Q_j \ + \ \lambda_P P_i$$

$$\frac{\partial L}{\partial Q_j} = \left(PQ^T - R_{ij}\right)^2 P_i \ + \ \lambda_Q Q_j$$

So we can update P and Q based on the rule showed below:

$$P_i = \ P_i - LR \ \times \ \frac{\partial L}{\partial P_i}$$

$$Q_j = \ Q_j - LR \ \times \ \frac{\partial L}{\partial Q_j}$$

where: LR is learning rate.

Now, we have dense matrices P and Q, so we can do SVD to each of them. There is a trick here, we do SVD to P and Q separately, then combine the matrices generated by SVD. Here are detailed descriptions:

$$PQ^T = \ U_P D_P {V_P}^T (U_Q D_Q {V_Q}^T)^T$$

And then do SVD to ${D_P}{V_P}^T V_Q {D_Q}^T$, so we can get:

$$U_P D_P {V_P}^T (U_Q D_Q {V_Q}^T)^T = \ U_P U_X D_X {V_X}^T {U_Q}^T$$

If $U_P U_X = U$, $D_X = \ D$, ${V_X}^T {U_Q}^T = V^T$, then we can get (U, D, V) of original sparse matrix R.

So that we finally can start the real work, our goal is to find the relationship between R1 and R2, according to SVD, R1, R2 become $U_1D_1V_1^T$, $U_2D_2V_2^T$. D is a diagonal matrix, and all diagonal values(eigenvalues) are sorted (big to small), so we can think of that they mean the importance of each feature. We assume R1 and R2 is highly correlated, so they we treat same feature in same behavior, so we only to find the relation of U_1 and U_2, so as V_1 and V_2.
So we have equation:

$$\mathbf{U}_1\mathbf{D}_1\mathbf{V}_1^T \approx \mathbf{G}_{\text{user}}\mathbf{U}_2\mathbf{D}_2\mathbf{V}_2^T\mathbf{G}_{\text{item}}^T$$

Where G_{user} /G_{item} mean the mapping of user/item from R2 to R1.

Then, we can decouple the user/item matching problem because of the uniqueness of SVD, the above equation can be decouples into two matching problems:

$$\min_{\mathbf{G}_{\text{user}},\mathbf{S}_{\text{user}}} \|\mathbf{U}_1\mathbf{D}_1^{0.5} - \mathbf{G}_{\text{user}}\mathbf{U}_2\mathbf{D}_2^{0.5}\mathbf{S}_{\text{user}}\|_{\text{Fro}}^2$$

$$\min_{\mathbf{G}_{\text{item}},\mathbf{S}_{\text{item}}} \|\mathbf{V}_1\mathbf{D}_1^{0.5} - \mathbf{G}_{\text{item}}\mathbf{V}_2\mathbf{D}_2^{0.5}\mathbf{S}_{\text{item}}\|_{\text{Fro}}^2$$

Given:

$$\mathbf{Z}_1 = \mathbf{U}_1\mathbf{D}_1^{0.5}, \mathbf{Z}_2 = \mathbf{U}_2\mathbf{D}_2^{0.5}\mathbf{S}$$

For each row in Z_1, the goal is to select the most similar row in Z_2.

Also, we have mentioned that SVD is unique except the sign of column vectors. Thus, we apply a sign matrix S where each diagonal element is either 1 or -1.To solve S, we solve G twice and find smaller objective value in nearest neighbor.

Finally, we pick the most suitable G_{user} and G_{item} from G(1:K), where the G(1:K) is the output in every iteration while solving S. W_1 is the matrix which record whether R_1 have value or not, We choose the G_{user} and G_{item} that minimize:

$$\|\mathbf{W}_1 \odot \left(\mathbf{R}_1 - \mathbf{G}_{\text{user}}\mathbf{U}_2\mathbf{D}_2\mathbf{V}_2\mathbf{G}_{\text{item}}^T\right)\|_{\text{Fro}}^2.$$

Moreover, in previous latent space matching, for each user/item we identify its nearest neighbor. Now, we propose a method to further refine them. We record the top 5 nearest candidates for each user and item, and reevaluate their quality based on the above equation to get the final G_{user} and G_{item}.

There following picture shows the whole algorithm in Latent Space Matching.

Algorithm 1 Latent Space Matching

Require: $\mathbf{U}_1, \mathbf{D}_1, \mathbf{V}_1, \mathbf{U}_2, \mathbf{D}_2, \mathbf{V}_2$ and $\mathbf{R}_1$
Ensure: $\mathbf{G}_{\text{user}}^*$ and $\mathbf{G}_{\text{item}}^*$
  function NN($\mathbf{Z}_1, \mathbf{Z}_2, T$)
    ▷ perform nearest neighbor search
    ▷ T is the distance cache for speed-up
    $\alpha = \min_{\mathbf{G}} \|\mathbf{Z}_1 - \mathbf{G}\mathbf{Z}_2\|_{\text{Fro}}^2$
    $\mathbf{G} = \underset{\mathbf{G}}{\text{argmin}} \|\mathbf{Z}_1 - \mathbf{G}\mathbf{Z}_2\|_{\text{Fro}}^2$
    return $(\alpha, \mathbf{G})$
  end function
  function MATCHING($\mathbf{Z}_1, \mathbf{Z}_2$)
    ▷ T is a cache for speed-up
    initialize all elements of the distance table T to 0
    for k = 1 to K **do**
      let $s_k = +1$
      $(\alpha_+, \mathbf{G}_+) = \text{NN}(\mathbf{Z}_{1\,(:,1:k)}, \mathbf{Z}_{2\,(:,1:k)}\mathbf{S}_{(1:k,1:k)}, \text{T})$
      let $s_k = -1$
      $(\alpha_-, \mathbf{G}_-) = \text{NN}(\mathbf{Z}_{1\,(:,1:k)}, \mathbf{Z}_{2\,(:,1:k)}\mathbf{S}_{(1:k,1:k)}, \text{T})$
      if $\alpha_+ \leq \alpha_-$ **then**
        $\mathbf{G}_{(k)} = \mathbf{G}_+, s_k = +1$
      else
        $\mathbf{G}_{(k)} = \mathbf{G}_-, s_k = -1$
      end if
      update T according to $\mathbf{G}_{(k)}$
    end for
    return $\mathbf{G}_{(1)}, \cdots, \mathbf{G}_{(K)}$
  end function
  $\mathbf{G}_{\text{user}\,(1:K)}$ = MATCHING($\mathbf{U}_1\mathbf{D}_1^{0.5}, \mathbf{U}_2\mathbf{D}_2^{0.5}$)
  $\mathbf{G}_{\text{item}\,(1:K)}$ = MATCHING($\mathbf{V}_1\mathbf{D}_1^{0.5}, \mathbf{V}_2\mathbf{D}_2^{0.5}$)
  for k = 1 to K **do** ▷ dimension select
    $\text{error}_k = \|\mathbf{W}_1 \odot \left(\mathbf{R}_1 - \mathbf{G}_{\text{user}\,(k)}\mathbf{U}_2\mathbf{D}_2\mathbf{V}_2\mathbf{G}_{\text{item}\,(k)}^T\right)\|_{\text{Fr}}^2$
  end for
  $(\text{error}_{\text{min}}, k_{\text{min}}) = \min_k \text{error}_{1:K}$
  $\mathbf{G}_{\text{user}}^* = \mathbf{G}_{\text{user}\,(k_{\text{min}})}$
  $\mathbf{G}_{\text{item}}^* = \mathbf{G}_{\text{item}\,(k_{\text{min}})}$

After we get G_{user} and G_{item}, we use them to convert the source's information and then fill them in train matrix. Finally, we can get a matrix merged of source and train data, and we can use libfm[2] to predict the answer of queries in test set.

Method 2: Biased MF.(for task 2)

Let R be a $m \times n$ rating matrix, k be the number of latent factor. Define the objective function as:

$$f(U,V) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(U_i V_j^T + \boldsymbol{b}_i + \boldsymbol{c}_j + \mu - R_{ij}\right)^2 + \frac{\lambda_U}{2}\|U\|^2 + \frac{\lambda_V}{2}\|V\|^2$$
$$+ \frac{\lambda_b}{2}\|\boldsymbol{b}\|^2 + \frac{\lambda_c}{2}\|\boldsymbol{c}\|^2 + \frac{\lambda_\mu}{2}\mu^2, if \;\; R_{ij} \;\; is \;\; given.$$
$$f(U,V) = 0, otherwise.$$

where $U \in \mathbb{R}^{m\times k}, V \in \mathbb{R}^{n\times k}, \boldsymbol{b} \in \mathbb{R}^m$ is the user bias vector, $\boldsymbol{c} \in \mathbb{R}^n$ is the item bias vector , μ is global bias, and λ are hyper parameters. The goal is to find $U, V, \boldsymbol{b}, \boldsymbol{c}, \mu$ such that $f(U,V)$ is minimized. Define the objective function of an instance as

$$f_{ij}(U,V) = \frac{1}{2}\left(U_i V_j^T + \boldsymbol{b}_i + \boldsymbol{c}_j + \mu - R_{ij}\right)^2 + \frac{\lambda'_U}{2}\|U_i\|^2 + \frac{\lambda'_V}{2}\left\|V_j\right\|^2$$
$$+ \frac{\lambda'_b}{2}\|\boldsymbol{b_i}\|^2 + \frac{\lambda'_c}{2}\left\|\boldsymbol{c_j}\right\|^2 + \frac{\lambda'_\mu}{2}\mu^2, if \;\; R_{ij} \;\; is \;\; given.$$
$$f_{ij} \;\; (U,V) = 0, otherwise.$$

We have

$$f(U,V) = \sum_{i,j} f_{ij}(U,V).$$

In order to minimize $f(U,V)$, we can minimize $f_{ij}(U,V)$ for every i,j by stochastic gradient descent with the following formula and update rules (if R_{ij} is given):

$$\frac{\partial f_{ij}}{\partial U_i} = \left(U_i V_j^T + b_i + c_j + \mu - R_{ij}\right)V_j + \lambda'_U U_i$$

$$\frac{\partial f_{ij}}{\partial V_j} = \left(U_i V_j^T + b_i + c_j + \mu - R_{ij}\right)U_i + \lambda_V' V_j$$

$$\frac{\partial f_{ij}}{\partial b_i} = \left(U_i V_j^T + b_i + c_j + \mu - R_{ij}\right) + \lambda_b' b_i$$

$$\frac{\partial f_{ij}}{\partial c_j} = \left(U_i V_j^T + b_i + c_j + \mu - R_{ij}\right) + \lambda_c' c_j$$

$$\frac{\partial f_{ij}}{\partial \mu} = \left(U_i V_j^T + b_i + c_j + \mu - R_{ij}\right) + \lambda_U' \mu$$

$$U_i := U_i - \eta_U \frac{\partial f_{ij}}{\partial U_i}$$

$$V_j := V_j - \eta_V \frac{\partial f_{ij}}{\partial V_j}$$

$$b_i := b_i - \eta_b \frac{\partial f_{ij}}{\partial b_i}$$

$$c_j := c_j - \eta_c \frac{\partial f_{ij}}{\partial c_j}$$

$$\mu := \mu - \eta_\mu \frac{\partial f_{ij}}{\partial \mu}$$

where η are learning rates. Once we learn $U, V, \boldsymbol{b}, \boldsymbol{c}, \mu$, we can predict the rating of item i from user u as $U_i V_j^T + b_i + c_j + \mu$. With $k = 5$, our best average root mean square error of 5-fold cross validation is 1.324950.

Reflection

LibMF and LibFM with 5-fold cross validation are 1.4416 and 1.2644, respectively (refer to the TA's slide.) Our method has better rmse than LibMF but still have a gap to LibFM. Here are some differences. First, in our biased MF method, we use stochastic gradient descent to learn the parameters, while LibFM learned the parameters by Markov Chain Monte Carlo method (in this case.) Second, LibMF doesn't consider user bias, item bias, and global bias. Third, the stopping condition and initialization might matter. Fourth, our splitting method

might be different.

Implementation details

Since computing f is expensive in other tasks, we came up with the following algorithm to approximate it.

1. Let $user\ \ window = {}^{m}/_{p},\ item\ \ window = n/q$,where p, q are integers. (Denoted as uw, iw respectively.)
2. Randomly choose $user\ \ lower$ from interval $[\ 1, m - uw]$ and $item\ \ lower$ from interval $[\ 1, n - iw]$. (Denoted as ul, il respectively.)
3. Approximate f as

$$f \approx \sum_{\substack{i \in [ul,\ ul+uw], \\ j \in [il,\ il+iw]}} f_{ij}$$

4. Repeat 1. to 3. r time, where r is an (small) integer. Compute their average as final approximation.

Reflection: After implementing the algorithm, we found that it's hard to determine when to stop. The problem is that different batch instances have different losses, hence the approximation of f won't be monotonically decreasing.

Method 3: Codebook Transfer(for task3)

We implemented the following algorithms in [3].

Algorithm 1 Codebook Construction

Input: An $n \times m$ auxiliary rating matrix $\mathbf{X}_{aux}$; the numbers of user and item clusters k and l.

Output: A $k \times l$ codebook $\mathbf{B}$ learned from $\mathbf{X}_{aux}$.

1: Randomly initialize $\mathbf{U}^{(0)}$, $\mathbf{V}^{(0)}$, and $\mathbf{S}^{(0)}$ in Eq. (1).
2: **for** $t \leftarrow 1, \ldots, T$ **do**
3: Update $\mathbf{U}^{(t-1)}$, $\mathbf{V}^{(t-1)}$, $\mathbf{S}^{(t-1)}$ using Eqs. (28–30) in [Ding *et al.*, 2006], and obtain $\mathbf{U}^{(t)}$, $\mathbf{V}^{(t)}$, $\mathbf{S}^{(t)}$.
4: **end for**
5: Allocate spaces for $\mathbf{U}_{aux}$ and $\mathbf{V}_{aux}$.
6: **for** $i \leftarrow 1, \ldots, n$ **do**
7: $\hat{j} = \arg\max_{j \in \{1,\ldots,k\}} \{\mathbf{U}_{ij}\}$.
8: $[\mathbf{U}_{aux}]_{i\hat{j}} \leftarrow 1$; $[\mathbf{U}_{aux}]_{ij} \leftarrow 0$ for $j \in \{1, \ldots, k\}/\hat{j}$.
9: **end for**
10: **for** $i \leftarrow 1, \ldots, m$ **do**
11: $\hat{j} = \arg\max_{j \in \{1,\ldots,l\}} \{\mathbf{V}_{ij}\}$.
12: $[\mathbf{V}_{aux}]_{i\hat{j}} \leftarrow 1$; $[\mathbf{V}_{aux}]_{ij} \leftarrow 0$ for $j \in \{1, \ldots, l\}/\hat{j}$.
13: **end for**
14: Calculate the codebook $\mathbf{B}$ using Eq. (2).

In algorithm 1, eq. (1) is

$$\min_{\mathbf{U}\geq 0, \mathbf{V}\geq 0, \mathbf{S}\geq 0} \left\| \mathbf{X}_{aux} - \mathbf{U}\mathbf{S}\mathbf{V}^\top \right\|_F^2$$
$$\text{s.t.} \quad \mathbf{U}^\top\mathbf{U} = \mathbf{I}, \mathbf{V}^\top\mathbf{V} = \mathbf{I},$$

eqs. (28-30) are

$$U_{ik} := U_{ik} \frac{(X_{aux}VS^T)_{ik}}{(UU^TX_{aux}VS^T)_{ik}}$$

$$V_{jk} := V_{jk} \frac{(X_{aux}^TUS\ )_{jk}}{(VV^TX_{aux}^TUS\ )_{jk}}$$

$$S_{ik} := S_{ik} \frac{(U^TX_{aux}V)_{ik}}{(U^TUS\ V^TV)_{ik}},$$

eq.(2) is

$$\mathbf{B} = [\mathbf{U}_{aux}^\top \mathbf{X}_{aux} \mathbf{V}_{aux}] \oslash [\mathbf{U}_{aux}^\top \mathbf{1}\mathbf{1}^\top \mathbf{V}_{aux}],$$

$U_{(aux)} \in \mathbb{R}^{n\times k}$, $S \in \mathbb{R}^{k\times l}$, $V_{(aux)} \in \mathbb{R}^{m\times l}$, and $\oslash$ means entry-wise division. Once we have the codebook B, we take it as input to algorithm 2:

Algorithm 2 Codebook Transfer

Input: The $p \times q$ target rating matrix $\mathbf{X}_{tgt}$; the $p \times q$ weighting matrix $\mathbf{W}$; the $k \times l$ codebook $\mathbf{B}$.

Output: The filled-in $p \times q$ target rating matrix $\tilde{\mathbf{X}}_{tgt}$.

1: Allocate spaces for $\mathbf{U}_{tgt}$ and $\mathbf{V}_{tgt}$.
2: **for** $i \leftarrow 1, \ldots, m$ **do**
3: Randomly select $\hat{j}$ from $\{1, \ldots, l\}$.
4: $[\mathbf{V}_{tgt}^{(0)}]_{i\hat{j}} \leftarrow 1$; $[\mathbf{V}_{tgt}^{(0)}]_{ij} \leftarrow 0$ for $j \in \{1, \ldots, l\}/\hat{j}$.
5: **end for**
6: **for** $t \leftarrow 1, \ldots, T$ **do**
7: **for** $i \leftarrow 1, \ldots, p$ **do**
8: $\hat{j} = \arg\min_j \left\| [\mathbf{X}_{tgt}]_{i*} - [\mathbf{B}[\mathbf{V}_{tgt}^{(t-1)}]^\top]_{j*} \right\|^2_{\mathbf{W}_{i*}}$.
9: $[\mathbf{U}_{tgt}^{(t)}]_{i\hat{j}} \leftarrow 1$; $[\mathbf{U}_{tgt}^{(t)}]_{ij} \leftarrow 0$ for $j \in \{1, \ldots, k\}/\hat{j}$.
10: **end for**
11: **for** $i \leftarrow 1, \ldots, q$ **do**
12: $\hat{j} = \arg\min_j \left\| [\mathbf{X}_{tgt}]_{*i} - [\mathbf{U}_{tgt}^{(t)}\mathbf{B}]_{*j} \right\|^2_{\mathbf{W}_{*i}}$.
13: $[\mathbf{V}_{tgt}^{(t)}]_{i\hat{j}} \leftarrow 1$; $[\mathbf{V}_{tgt}^{(t)}]_{ij} \leftarrow 0$ for $j \in \{1, \ldots, l\}/\hat{j}$.
14: **end for**
15: **end for**
16: Calculate the filled-in rating matrix $\tilde{\mathbf{X}}_{tgt}$ using Eq. (8).

In algorithm 2, eq.(8) is

$$\tilde{\mathbf{X}}_{tgt} = \mathbf{W} \circ \mathbf{X}_{tgt} + [\mathbf{1} - \mathbf{W}] \circ [\mathbf{U}_{tgt}\mathbf{B}\mathbf{V}_{tgt}^\top],$$

and $U_{tgt} \in \mathbb{R}^{p \times k}$, $V_{tgt} \in \mathbb{R}^{q \times l}$. After we get the approximation of X_{tgt}, we perform memory based predicting methods.

Implementation details

1. In line 1 of algorithm 1, instead of initializing U, V, S randomly, we initialized them as follows. Do k-means clustering on users of X_{aux}, then initialize U as the cluster indicator of each user. Similarly, do k-means clustering on items of X_{aux}, then initialize V as the cluster indicator of each item. Then, let $U := U + 0.02$, $V := V + 0.02$, $S = U^T X_{aux} V$.
2. We found that eq.(2) of algorithm 1 needs X_{aux} to be fully observed, but we don't want to fill in the missing value in X_{aux}. Hence, we modify eq.(2) such that the algorithm only needs to consider the observed ratings.

III、Result

We have used 5-fold cross validation average to test whether our result is above the baseline.

In task 1 using Latent Space Matching, Considered the mass time consuming problem, we only choose K = 50 for each training. However, it still wasted so much time calculating the difference (error) between R1. Therefore, we tried not to run all elements in R1, which means only used 1/1000 elements to test error, then found out the result is about 0.1758, which will be similar to using all elements, but this method really saved lots of time. Also, we have tested whether the performance will become better or not if we only find the most nearest row while matching user and item pairs and it turns out similar result again, 0.1756.

In task 3 using Latent Space Matching, we got the most improvement, which the result is about 1.22.

Following are some results of task 3 in method3:

(5-fold cross validation's average root mean square error,$k = 18, l = 8$)

K/method	RCK	RPK	CCK	CPK	GCK	GPK	CBT.CK	CBT.PK
10	1.4587	1.4058	1.4110	1.4055	1.4163	**1.4028**	1.4401	1.4261
20	1.4152	1.3964	1.4004	1.3949	1.4019	**1.3940**	1.4207	1.4145
30	1.4009	1.3951	1.3942	1.3926	1.3941	**1.3925**	1.4105	1.4065

Notations

RCK/ RPK/ CCK/ CPK: Given a query $< u, i >$, first filled in missing rating of target matrix by the mean of each '**r'ow/'c'olumn(1st character)**, then do user-based collaborative filtering with '**c'osine similarity/'P'earson correlation(2nd character)**. Predict $r_{u,i}$ as $\frac{1}{N}\sum_{u' \in U} r_{u',i}$, where U is the set of top $\boldsymbol{K^{th}}$ users that are most similar

to u who rated i.

GCK/ GPK: Filled in missing value by the **global** mean of target matrix, then do top K^{th} user-based collaborative filtering with '**c'osine similarity/'P'earson correlation**.

CBT.CK/ CBT.PK: Cookbook transfer. (Our method.) Then do top K^{th}user-based collaborative filtering with '**c'osine similarity/'P'earson correlation**.

Remarks

1. As K increases, all methods' rmse decreases.
2. We filled in the target matrix by the mean of each column, then directly use the (u, i) entry as prediction (i.e., no user based collaborative filtering.) We get rmse 1.3910, which outperforms all the rmse in the above table, but still have a gap to LibFM's 1.2644.
3. It seems that codebook transfer still need to be improved to outperform other baseline methods. For example, we can relax the constraint such that each row of U, V can have multiple $ones$.

IV、Contribution

何文琦 ： Implement method1 (Latent matching) + experiment on task1, 2, 3
宋易霖 ： Implement method1 (MF + SVD) + experiment on task 1, 2, 3
陳威嘉 ： Implement method2, 3 + experiment on task 2, 3

V、Reference

[1] Chung-Yi Li, Shou-De Lin,Matching Users and Items Across Domains to Improve the Recommendation Quality. In KDD'14
[2] http://www.libfm.org/
[3]:Can Movies and Books Collaborate? Cross-Domain Collaborative Filtering for Sparsity Reduction