

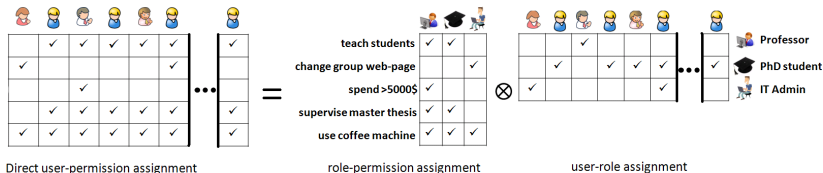
# **Boolean Matrix Decomposition: Problem Formulations and Quality Measures**

**Mario Frank**

April 6, 2011

# Role-Based Access Control (RBAC)

**Main Idea:** Users own their permissions because they fulfill a **role**. Each role defines a set of permissions. Users are assigned to one or several roles and they get all permissions of all roles they belong to.



Note that a user can belong to more than one role.

The **Boolean matrix product**  $\otimes$  is defined such that

$$\mathbf{X} = \mathbf{U} \otimes \mathbf{Z} \iff x_{di} = \bigvee_k [u_{dk} \wedge z_{ki}]$$

# Evaluation Criteria for Boolean Matrix Decomposition

For each of the three problem definitions, there is a (set of) evaluation criteria:



1. **Matrix Reconstruction:** How accurately does your solution reconstruct the input matrix  $\mathbf{X}$ ?
2. **Number of Sources:** How many sources do you need to get an approximation at a given accuracy?
3. **Inference Quality:** How accurately can you infer the underlying structure?

# Evaluating a Matrix Reconstruction

How precisely is the given matrix  $\mathbf{X}$  approximated by the inferred solution? Measures for this discrepancy are:

- ▶ **Deviation:** Compute  $\frac{1}{N \cdot D} \|\mathbf{X} - \hat{\mathbf{U}} \otimes \hat{\mathbf{Z}}\|_1$ . This is the criterion used to define the min-noise approximation.
- ▶ **Coverage:** The ratio of true 1's which are reconstructed as 1's. Formally:  $Cov := \frac{|\{(i,j) | \hat{x}_{i,j} = x_{i,j} = 1\}|}{|\{(i,j) | x_{i,j} = 1\}|}$
- ▶ **Deviating Ones (deviating Zeros):** The number of deviating 1's (0's) is the number of matrix elements which are wrongly reconstructed divided by the total number of 1's (0's) in  $\mathbf{X}$ .

$$d_1 := \frac{|\{(i,j) | \hat{x}_{i,j} = 1, x_{i,j} = 0\}|}{|\{(i,j) | x_{i,j} = 1\}|}$$
$$d_0 := \frac{|\{(i,j) | \hat{x}_{i,j} = 0, x_{i,j} = 1\}|}{|\{(i,j) | x_{i,j} = 0\}|}$$

## Exercise 1a - Coverage

**Coverage:** The ratio of true 1's which are reconstructed as 1's.  
Formally:

$$Cov := \frac{|\{(i, j) | \hat{x}_{i,j} = x_{i,j} = 1\}|}{|\{(i, j) | x_{i,j} = 1\}|}$$

Discuss the coverage measure, Why is it useful? What are its weak points?

## Exercise 1b - Deviation Ones/Zeros

**Deviating Ones (deviating Zeros):**

$$d_1 := \frac{|\{(i, j) | \hat{x}_{i,j} = 1, x_{i,j} = 0\}|}{|\{(i, j) | x_{i,j} = 1\}|}$$

$$d_0 := \frac{|\{(i, j) | \hat{x}_{i,j} = 0, x_{i,j} = 1\}|}{|\{(i, j) | x_{i,j} = 0\}|}$$

Calculate the deviating ones and zeros of the two following matrices (matlab code is okay):

$$\mathbf{X} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \quad \hat{\mathbf{X}} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix} \quad (1)$$

## Exercise 1c - Coverage and Deviating Zeros

**Coverage:**

$$Cov := \frac{|\{(i, j) | \hat{x}_{i,j} = x_{i,j} = 1\}|}{|\{(i, j) | x_{i,j} = 1\}|}$$

**Deviating Zeros:**

$$d_0 := \frac{|\{(i, j) | \hat{x}_{i,j} = 0, x_{i,j} = 1\}|}{|\{(i, j) | x_{i,j} = 0\}|}$$

What is the relation between  $d_0$  and  $Cov$ ? If you were only given the numerator and the denominator of  $d_0$  and the dimensions of  $X$ , how can you compute  $Cov$ ?

# Rounded SVD for Denoising

Use SVD to get rid of some noise in the data matrix  $\mathbf{X}$ :

1. Compute the singular value decomposition

$$\mathbf{X} = \mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^T$$

2. Discard columns  $K + 1, \dots, D$  of  $\mathbf{U}$  to get  $\mathbf{U}_{(K)}$   
Discard rows and columns  $K + 1, \dots, N$  of  $\mathbf{S}$  to get  $\mathbf{S}_{(K)}$   
Discard rows  $K + 1, \dots, N$  of  $\mathbf{V}^T$  to get  $\mathbf{V}_{(K)}^T$
3. Round the reconstruction

$$\mathbf{X}_{(K)} := \mathbf{U}_{(K)} \times \mathbf{S}_{(K)} \times \mathbf{V}_{(K)}^T$$

to get

$$\hat{\mathbf{X}}_{(K)} = (\mathbf{X}_{(K)} > t_X)$$

where  $t_X$  is a threshold.



# SVD in Matlab: Dimension Reduction

**Fact:** The data matrix was generated with  $K = 3$  sources.

```
[ U, S, V ] = svd(X);
```

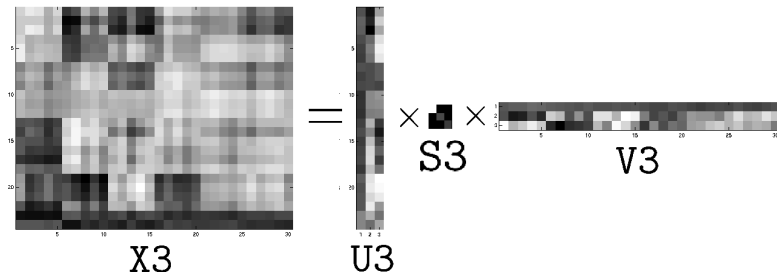
```
K = 3;
```

```
U3 = U( : , 1:K);
```

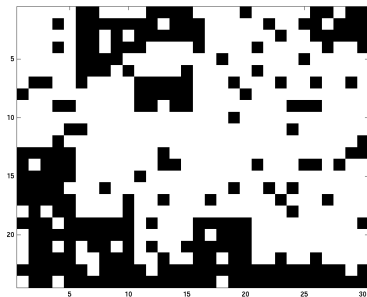
```
S3 = S(1:K, 1:K);
```

```
V3 = V(1:K, : );
```

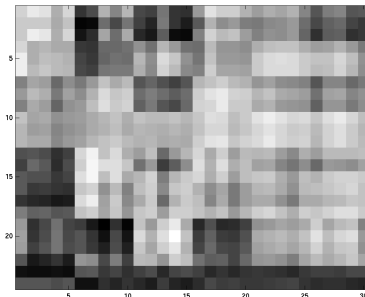
```
X3 = U3 * S3 * V3';
```



# SVD: The Low-Dimensional Reconstruction



Input Data



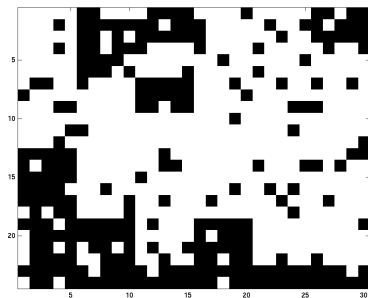
3-dimensional reconstruction

## Observations:

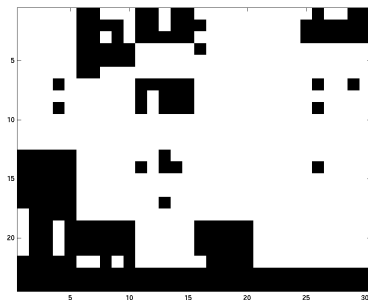
- ▶ Without rounding, the reduction of the number of dimensions clearly deteriorates the reconstruction.

# SVD: Rounding the Reconstruction

Using  $t_X = 0.445$ , we get the best reconstruction of the binary matrix  $\mathbf{X}$  in 3 dimensions.



Input Data



3-dimensional reconstruction  
rounded with  $t_X = 0.4450$

$\frac{1}{N \cdot D} \|\mathbf{X} - \hat{\mathbf{X}}_{1,\dots,K}\|_1 = 11.9\%$ . Some noise in the matrix seems to be removed.

**Soo... all done?**

# SVD: Choosing the Right Parameters

We actually need 3 values:

- ▶ a threshold  $t_X$ , to be chosen depending on the data
- ▶ a **hyper-parameter**  $K$ , the number of dimensions
- ▶ an estimate for the reconstruction error

In the previous example,  $K$  was fixed to  $K = 3$ .

The reconstruction error was 11.9% on the training data.

**BUT:** It is not clear that the same error will be achieved on a new, “untouched” data set!

## SVD: Choosing the Right Parameters (2)

We need three disjoint data sets:







If you have only one data set: Split it into three disjoint parts.

- ▶ Use the **training data** to estimate the best value of the parameters  $\theta$ . In our setting:  $\theta = t_X$  (given a fixed  $K$ )
- ▶ Use the **validation data** to determine the best value of the hyper-parameters  $\theta^H$  (for the best value of  $\theta$ ). In our setting:  $\theta^H = K$
- ▶ Use the **test data** to estimate the reconstruction error

# Choosing Hyper-Parameters: High-Level Algorithm

To find the optimal values of the hyper-parameter  $\theta^H$ :



1. Get three disjoint data (sub)sets: 
2. For all possible values of the hyper-parameter  $\theta^H$ : Fix  $\theta^H$  and
  - 2.1 **train** your method on the **training data** to get  $\theta^*$  
  - 2.2 **validate** your method on the **validation data** 
3. Set

$$\theta^{H*} = \left( \begin{array}{l} \text{Value of } \theta^H \text{ that maximized the performance} \\ \text{on the validation data }  \\ \text{(with the respective best value of } \theta \text{)} \end{array} \right)$$

The **test data** is NOT used to determine the value of  $\theta^{H*}$ !

# Model Evaluation: High-Level Algorithm

Once you have found the optimal hyper-parameter  $\theta^{H*}$ , proceed as follows to evaluate your method:

1. Fix  $\theta^{H*}$
2. Train your method to get the optimal parameter value  $\theta^*$  on the **training data** and the **validation data** 
3. Test your method with  $\theta^{H*}$  and  $\theta^*$  on the **test data** 

**This is the final result for your method.**

# Choosing Hyper-Parameters: Comments



- ▶ If you have several data matrices from the same source, you can use them as training, validation and test data. If not, you have to randomly split your data.
- ▶ It is (implicitly) assumed that the data in the three parts has the **same distribution**. If this is not the case, things become much more complicated.
- ▶ The division into the three parts is random and reduces the number of training data. To avoid too much dependency on this choice, one often uses several splits into training and validation data. This is called **cross-validation**.



# Choosing Hyper-Parameters: Comments

Crossvalidation allows you to estimate how well your method will perform on a new, unseed dataset.

- ▶ This avoids **overfitting**, a too detailed adaptation to the training data which renders the method very specific to the training data
- ▶ Therefore, **the test data is only considered at the very end! Never** use it before, or you get the red card (and misleading results).
- ▶ If you select methods or parameters based on the performance on the test data, you also get misleading results!



## Exercise 3: Cross Validation

Use cross validation to determine the number of dimension  $K$  and the threshold  $t$  for the SVD denoising method you have seen in the lecture.

### Environment setup:

1. The material for this exercise is on the CIL webpage.
2. Download the .zip file containing a template for the cross validation function, the evaluation script, and the data into a local folder.
3. Try running the evaluation script `Evaluate_RBAC_roundedSVD.m`. The result you get is the reconstruction error for an ignorant choice of the parameters, which is hard coded in the cross validation function `cross_validate_roundedSVD.m`.

## Exercise 3: Evaluation script

Look at the `Evaluate_RBAC_roundedSVD.m` script now to see what it does:

1. load data set
2. split data randomly into a training and a test set.
3. call cross validation function on training set.
4. compute test error.
5. repeat multiple times and compute standard deviation and median of the error.