Exercises
Computational Intelligence Lab
SS 2011

Machine Learning Laboratory
Dept. of Computer Science, ETH Zürich
Prof. Dr. Joachim M. Buhmann
Web http://ml2.inf.ethz.ch/courses/cil

Series 7, Apr 7th, 2011
(Introduction to the Role Mining Problem)

Problem 1 (Evaluation Criteria):

In the lecture you have seen different evaluation criteria for Boolean matrix decomposition:

- **Coverage:** The ratio of true 1es which are reconstructed as 1es. Formally:

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

- **Deviating Ones (deviating Zeros):** The ratio of deviating 1es (0es) is the number of matrix elements which are wrongly reconstructed and the total number of 1es (0es) in $\mathbf{x}$.

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

Exercises:

- Discuss the coverage measure. Why is it useful? What are its weak points?
- Calculate the deviating ones and zeros of the two following matrizes:

$$\mathbf{X} = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix} \qquad \hat{\mathbf{X}} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

- What is the relation between d_0 and Cov? Hint: If you were only given the enumerator and the denominator of d_0 and the dimensions of X, how can you compute Cov?

Problem 2 (Cross Validation):

This programming problem should make you familiar with cross validation. You develop and evaluate a denoising algorithm locally on your machine, but don't have to send it to the submission system.

The task is to denoise a Boolean matrix via truncated SVD. As you have seen in the lecture, we have to select two parameters: K, the number of nonzero singular values (the truncation sets the others to zero) and t, the rounding threshold. Our algorithm uses cross-validation to tune these parameters.

We begin with first setting up the environment on your local machine such that you can run your algorithm.

Environment setup:

1. The material for this exercise can be found here:

   http://cil.inf.ethz.ch/material/exercise/material/crossValidation.zip.

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 output is the reconstruction error of the denoised matrix compared against the noise-free matrix (the ground truth). For now the two parameters are hard coded in the cross validation function `cross_validate_roundedSVD.m`.

Evaluation script:

Look at the `Evaluate_RBAC_roundedSVD.m` script and run it once to see what it does:

1. It loads a data set, which consists of the noisy matrix $\mathbf{X}$ and the *ground truth* matrix $\mathbf{X}_{gt}$ which does not contain any noise. As we are working on simulated data, the ground truth is known and we can evaluate the result of rounding against the true matrix $\mathbf{X}_{gt}$.

2. The data is randomly split into a training set and a test set.

3. The cross validation function is called on the training set. This cross validation function is the part you should implement. It returns the estimated number of clusters K for the SVD and a threshold t.

4. The parameters returned by the cross validation function are then applied to the test data set and the test error is computed.

5. This whole process is done multiple times to average over multiple results and to estimate the standard deviation of the error.

The evaluation struct you get contains the following fields:

- results.ranking.medianError: median error over a number of trials
- results.ranking.stdDeviation: standard deviation of the error
- results.errors: the individual error values
- results.K: the K^* estimated by your function for each trial
- results.t: the t_{K^*} estimated by your function for each trial

Cross validation function for rounded SVD:

Your task is to modify the file called `cross_validate_roundedSVD.m`. It implements the function

```
[K, t] = cross_validate_roundedSVD(X, X_gt)
```

which takes a data matrix $\mathbf{X}$ and a ground truth matrix $\mathbf{X}_{gt}$ as input and outputs the parameters K and t. Remove the hard coded parameter values for K and t, and add code which implements the following steps:

1. Split $\mathbf{X}$ and $\mathbf{X}_{gt}$ into training sets and validation sets: $\mathbf{X}^{train}$, $\mathbf{X}^{val}$, $\mathbf{X}_{gt}^{train}$ and $\mathbf{X}_{gt}^{val}$.
2. For every possible value of K, find the best threshold t_K which minimizes the error on the training data:
   (a) Compute SVD of training data $\mathbf{X}^{train}$.
   (b) Loop over all possible values K:
      i. Compute $\mathbf{X}_{(K)}^{train}$ which approximates $\mathbf{X}^{train}$ using K components of the SVD.
      ii. Find the rounding threshold t_K that minimizes the error $\|\tilde{\mathbf{X}}_{(K)}^{train} - \mathbf{X}_{gt}^{train}\|_1$, where $\tilde{\mathbf{X}}_{(K)}$ is the rounded version of $\mathbf{X}_{(K)}$ using threshold t_K.
      iii. Save (K, t_K).

   For every K, you now have a corresponding optimal t_K, stored as a tuple (K, t_K).
3. Find K^* such that the pair (K^*, t_{K^*}) performs best on the validation set:
   - Compute the SVD of validation data $\mathbf{X}^{val}$.
   - Loop over all K to find K^* such that the pair (K, t_K) minimizes the error on the validation data.
4. For K^* (estimated on the validation set), re-estimate t_{K^*} on the training and validation data set together ($\mathbf{X}$ and $\mathbf{X}_{gt}$):
   - Compute the SVD of $\mathbf{X}$.
   - Compute $\mathbf{X}_{(K^*)}$ which approximates $\mathbf{X}$ using K^* components of the SVD.
   - Find the rounding threshold t_{K^*} that minimizes the error $\|\tilde{\mathbf{X}}_{(K^*)} - \mathbf{X}_{gt}\|_1$.