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 8, Apr 14th, 2011 (Probabilistic Role Mining)

Problem 1 (Complete and Fast Miner):

In the lecture, the role mining algorithms *CompleteMiner* and *FastMiner* were introduced. The idea behind these methods is that the permission set of a user is generated by a combination of roles. Thus, all possible intersections of the permission sets of all users should give the underlying atomic components.

Recall the algorithm of *CompleteMiner*:

- Initialize the candidate roles $\mathbf{U}$ and the vector of counts $c = (c_1, .., c_k)$ with the k unique users in $\mathbf{X}$.
- Compute all possible set intersections between the members of $\mathbf{U}$ and add them to $\mathbf{U}$. Update the count of roles whose permission set is a superset of the respective intersection in c.

The output is a prioritized set of roles U, where the c_k gives the priority of the role $\mathbf{u}_{.,k}$.
See the slides of the lecture for a small example of the algorithm.

Exercises:

1. Complexity of *CompleteMiner*: Let t be the number of users with unique permission set. Show that *CompleteMiner* computes $2^t - t - 1$ set intersections.
2. *FastMiner* reduces complexity by computing only the pairwise intersections. Show that *FastMiner* computes $\frac{t(t-1)}{2}$ intersections.

Problem 2 (Probabilistic Clustering):

In probabilistic clustering, one maximizes the probability of $\mathbf{X}$ given the cluster (user-role) assignments $\mathbf{Z}$ and the cluster centroids (role-permission assignments) $\mathbf{U}$ with respect to a likelihood $p(\mathbf{X}|\mathbf{U}, \mathbf{Z})$.

Recall the model for single-assignment clustering without a noise process from the lecture:

- maximize the data likelihood $p(\mathbf{X}|\boldsymbol{\beta}, \mathbf{Z})$
- β_{dk} is the probability that role k does *not* contain permission d: $\boldsymbol{\beta} = (p\{u_{dk} = 0\})^{D \times K}$
- each object (user) i is assigned to exactly one cluster (role) k_i: $\sum_k z_{ki} = 1, \forall i \in \{1, \ldots, N\}$

The data likelihood is defined as:

$$\begin{aligned} p(\mathbf{X}|\boldsymbol{\beta}, \mathbf{Z}) &= \prod_{i=1}^{N} \prod_{d=1}^{D} p(x_{di} = 1|\boldsymbol{\beta}_{\boldsymbol{d}\cdot}, \mathbf{z}_{\cdot\mathbf{i}})^{x_{di}} \cdot p(x_{di} = 0|\boldsymbol{\beta}_{\boldsymbol{d}\cdot}, \mathbf{z}_{\cdot\mathbf{i}})^{1-x_{di}} \\ &= \prod_{i,d} (1 - \beta_{dk_i})^{x_{di}} \beta_{dk_i}^{1-x_{di}} \end{aligned}$$

Exercise:

- Derive the $\beta_{\mu\nu}$ for given $\mathbf{Z}$ and $\mathbf{X}$ that maximizes the data likelihood.
  Hint: Instead of maximizing $p(\mathbf{X}|\boldsymbol{\beta}, \mathbf{Z})$ it is algebraically much simpler to minimize $-\log(p(\mathbf{X}|\boldsymbol{\beta}, \mathbf{Z}))$.
  Hint 2: What would your ad hoc guess be?

To extend the approach from single assignment clustering to multi-assignment clustering, we drop the constraint that the assignments have to sum up to one: $\sum_k^K z_{ki} \leq K, \forall i \in \{1, \ldots, N\}$. For simpler notation, we introduce the assignment set $\mathcal{L}_i$, which contains all clusters an object i is assigned to: $\mathcal{L}_i := \{k|z_{ki} = 1\}$

Exercises:

- What is the maximal cardinality $|\mathcal{L}_i|$ of such an assignment set?
- How many assignment sets exist in total?
- How many assignment sets exist if we restrict the number of multi-assignments to some fixed number $L \leq K$ s.t.

$$\sum_k^K z_{ki} \leq L, \ \forall i \in \{1, \ldots, N\}$$

Problem 3 (Programming: Role Mining):

This week you start to work on the role mining problem. The direct user-permission assignment is represented in the matrix $\mathbf{X}$. The task is to learn permission-role assignments and user-role assignments from $\mathbf{X}$, such that they generalize well to new users.

Submission system environment setup:

1. Download the zipped Matlab files for this assignment from the CIL-webpage. There is a template of the function you will implement, as well as the evaluation script and a training dataset.
2. Unzip everything into the *same* folder.
3. Try running the evaluation script `results = rbac_eval.m` – it should output results.ranking ≈ 0.35. This is the generalization error of the dummy implementation in `estimateRolesAndAssignments.m`.

The file `estimateRolesAndAssignments.m` defines the function

```
[Z,U] = estimateRolesAndAssignments(X).
```

This function takes a data matrix $\mathbf{X}$ as an input and outputs the role-user assignment matrix $\mathbf{Z}$ and the role-permission assignment matrix $\mathbf{U}$. The dummy version just takes the permission sets of the first 2 users as roles. Please substitute this with your own code.

To submit your solution to the online evaluation system, prepare a ".zip" file containing:

1. A function file called `estimateRolesAndAssignments.m` which implements the function `[Z,U] = estimateRolesAndAssignments(X)`. This function takes a data matrix $\mathbf{X}$ as input and outputs $\mathbf{Z}$ and $\mathbf{U}$.
2. All your additional functions needed by your `estimateRolesAndAssignments.m`.

Your submission is evaluated according to the following criteria:

1. Generalization error
2. CPU time of the optimization routine

Representation of data:

- $\mathbf{X}$: a $D \times N$ Boolean matrix assigning D permissions to N users
- K: a scalar value representing the number of roles
- $\mathbf{Z}$: a $K \times N$ matrix assigning one role to each user
- $\mathbf{U}$: a $D \times K$ matrix assigning permissions to roles

Single assignment clustering for role mining: The easiest approach to solve the task is to use K-means with Hamming distance.

The optimization alternates between two steps:

1. Assign each user to the roles that best fits his/her permissions.
2. Update the roles to be prototypical for the users assigned.

As the role mining problem only makes sense for Boolean data, we have to adapt the objective function to Hamming distance (0-norm):

$$R(\mathbf{U}, \mathbf{Z}) = \|\mathbf{X} - \mathbf{U}\mathbf{Z}\|_0 = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ki} \|\mathbf{x}_i - \mathbf{u}_k\|_0.$$

Furthermore we have to make sure that the roles only contain Boolean values. Thus, we need to adapt the update step:

$$u_{kd} = \text{median}\left(\{x_{dn} | z_{kn} = 1\}\right) \ \forall k \in \{1, \ldots, K\}, \forall d \in \{1, \ldots, D\}$$

You should now have all relevant information to implement the function `[Z,U] = estimateRolesAndAssignments(X,K)`.

Points to consider:

- think about an initialization for $\mathbf{U}$
- think about a criterion to terminate the iteration procedure
- start the function by initializing $\mathbf{U}$
- now loop over the assignment and update step, until your convergence criterion is fulfilled
- implement the assignment and the update step as two extra help functions
- for the assignment step, assume that the set of roles $\mathbf{U}$ is known and assign each user his/her role by defining $\mathbf{Z}$
- for the update step assume that $\mathbf{Z}$ is known and compute $\mathbf{U}$
- implement your code for a fixed number of roles first (say 2) and then think about how you could use the evaluation script `generalizationError.m` to automatically select K.

Probabilistic Roles: As seen in the lecture, the roles can be formulated in a probabilistic way by setting $\boldsymbol{\beta} := (p\{u_{dk} = 0\})^{D \times K}$. Extend your code such that it can handle the probabilistic formulation, and use the solution from Problem 2 to find the optimal values of $\boldsymbol{\beta}$ given the assignment matrix $\mathbf{Z}$.
Note that the roles should be binary. You can obtain the matrix $\mathbf{U}$ from the final $\boldsymbol{\beta}$ by either sampling or simply by rounding.

Possible extensions: Feel free to explore other methods as well.