Automatic Resource Access Control

Xinghuang Xu

Wichita State University

Wichita, Kansas

xxxu3@wichita.edu

*Abstract*— In each organization, there are resources that help its employees to fulfill their job responsibilities. Maybe it is permission to enter to a lab, permission to open a file, or permission to checkout a workstation. To prevent people from abusing resources, resources are usually managed by a resource keeper. Whoever want to gain access need to request for the appropriate permission. The resource management process usually involves a nontrivial human involvement and such human involvement in resource management can be costly. This paper presents an approach to build a machine learning model which would automatically grant/deny a resource request given the requester’s role information.

Keywords—supervised learning; machine learning;

# Introduction

Employees in companies need to access resources like read/manipulate various applications or web portals to fulfill their job responsibilities. Usually there is a knowledgeable supervisor who would manually verify the request and grant the access in order to overcome access obstacles. When employees leave their current roles or move on to new roles, their resources they have accessed to need to be cleaned up or updated accordingly. This access discovery/recovery process could waste a nontrivial amount of time and money especially in a company.

In big organization, there is considerable amount of data regarding historical resource requests. Given the data related to employee roles, their resource requests and the requests outcome, either grant or deny access, we attempt to build a model to automatically cleanup/update access privileges as employees enter/leave roles within a company? The goal of the model is to minimize the human factor in resource access management.

# Problem Definition

## Term Definition

|  |  |
| --- | --- |
| Term | Definition |
| Resource | A supply of money, materials, permissions, staff, and other assets that can be granted to an entity in order to function effectively. An entity can be a person, an organization or an application. |
| Role | The function assumed or part played by a person or thing in a particular situation. For example, manager is a role in a company. |
| Access Control | The process of deciding whether a role has access to a resource. |
| Resource Request | A resource request contains the action (grant or deny), the resource id and a list of requester’s role features. |

## Problem Definition

To automate the access control process through building a machine learning model trained with historical resource requests.

## Evaluation

Based on experience, employees are granted the access they requested most of the time because if they don’t need those resources they would not have asked for it at the first place. So a stupid algorithm that simply guesses “yes” every time will have a high accuracy. Given the skewness of the data, algorithms are measured based on the area under the ROC curve or AUC. The AUC is the probability that an algorithm will rank a randomly chosen positive data point higher than a randomly chosen negative one.

# Solution technique

The solution is implemented in Python, using scikit-learn(sklearn). Sklearn is a set of simple and efficient data mining and data analysis tools built on NumPy, SciPy and matplotlib. This section is a walk through of the solution code. Noted that the solution can be easily implemented in other languages such as R, Matlab. The solution is based on code by Miroslaw Horbal[1] and Paul Duan’s sklearn starter code[3]. The solution is decomposed into multiple steps; each step is explained in detail below.

## Relabel Catagorical Data

The first step is to transform the data. Since all the values are categorical information, we relabel the encoding such that the possible values range from 0 to the number of categories -1. This relabeling preprocessing is to enhance the hashing efficiency and reduce hashing collisions in the later steps.

../../../../Desktop/Screen%20Shot%202015-11-29%20at%203.30.58%20PM.

Fig 1. Relabeling each column with the label encoder provided from sklearn.

## Combine features to build new ones

We want to combine features to create new features, features that would provide more useful information. For example, if we want to predict if a price offered is a good deal using historical house selling data. The price and the size are two features of a house. But if we divide the size by the price, we end up with a new feature called cost per meter square which can be more useful in predicting if the house is a good deal.

In our case, a function named feature creation (see fig.2) is used to group features together into all possible combinations of n degrees. For example, for data entry [1, 2, 3] which has 3 features, all possible combinations of 2 degrees are (0,1), (0,2) and (1,2). For combination (0,1) we get value from column 0 and column 1 and have tuple (1,2) then we create a hash code for tuple (1,2) and treat it as a new data value. To create a degree 2 column groupings of data entry [1,2,3] we would obtain something like [hv1, hv2, hv3] where hv1 is hash value of tuple (1,2) the combination of the first column and second column, hv2, the hash value of (1,3) and hv3 the hash value of (2,3).

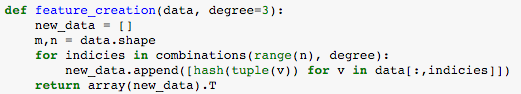


Fig 2. Function definition of feature\_creation

All the 2nd and 3rd order combination of the input features are created. If the original dimension of a data entry has k features. By combining two of the original columns we can create k\*k/2 number of features and k\*(k-1)\*(k-2)/(3\*2) features if we pick any three of the original features to combine. If we combine the first degree, 2nd degree and the 3rd degree features together we would end up with k + k\*k/2 + k\*(k-1)\*(k-2)/(3\*2) features but some of these features can be useless and can even produce noise prediction model. We will filter out bad features in our later steps.

## Merge Rare Categories

This step is to help reduce the number of unique values in a column by replacing all the rare columns with a constant value. For example, assume that the number of unique values in a column is K, we will replace all the categorical value which only appear once in that column with K and replace all the categorical values which appear twice with K+1. Using this approach, a column with more than 60,000 unique values can be reduce to have only 7000 unique values. The reason to reduce the number of unique values in a column is to reduce the size of the sparse matrix generated by encoding a categorical feature. After the encoding, a column with 60,000 unique values would be replaced with 60,000 binary columns. So doing this greatly improve the efficiency of the encoding step and enhance the performance of the whole algorithm. See fig. 3 for the function that is being used to combin rare values.

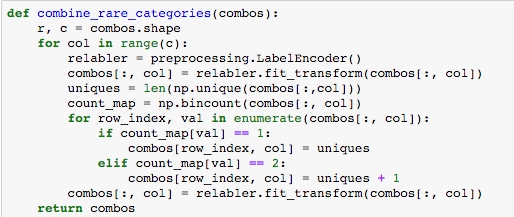


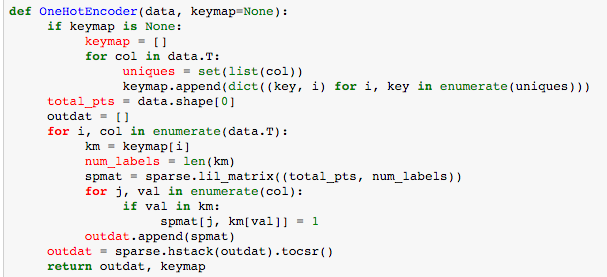
Fig 3. Function to merge rare values in a feature

## Create the Machine Leanring Model

There are many machine learning models that can solve a classification problem. Models like Logistic Regression, Naïve Bayes, Support Vector Machine (SVM) and Decision Trees are the ones we have experimented with in solving our particular resource access control problem. The best AUC score we can come up with is by using the Logistic Regression model.

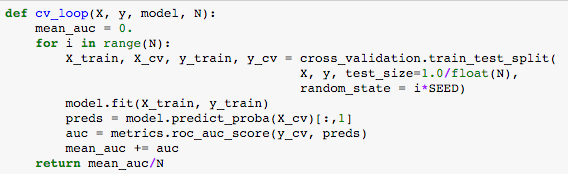
## Encode Categorical Data

Many estimators we are applying such as linear models, Naïve Bayes and SVMs require the categorical data input to be encoded. Categorical data describe some aspects of an object/phenomenon in a way that can’t directly be related to other values in a useful mathematical way. Most of the role information in the data are qualitative variables such as RoleId, the deparemtne the role belongs to, the manager of the role etc. We use a simple way to encode, for each column, we replace it with n different binary columns and each represent a value in the original column. For example, to encode a column gender with two possible values “male” and “female”, we can replace this column with two binary columns “is\_male” and “is\_female”. If in the original column the value is “male”, the value for the “is\_male” column would be set to 1 and the “is\_female” column would be set to 0.

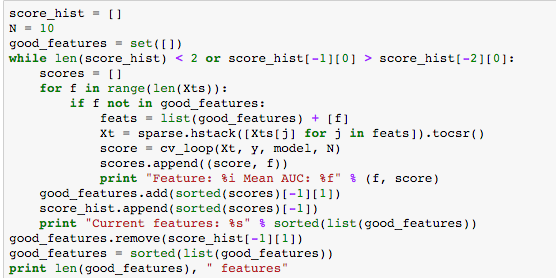


## Greedy Feature Selection

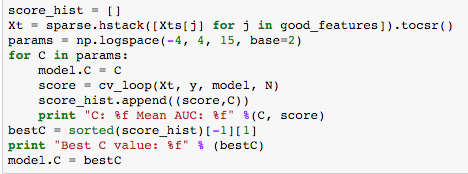
Before performing the feature select, we transform all the traning data into sparse matrix using the OneHotEncoder function. The resulting array Xts contains an array of sparse matrix, each represent an encoding of a feature. We are going to feed these encoded features into our model and perform cross validation to figure out which feature is more useful then the other.



The greedy feature selection process is a while loop. In each iteration, features that have not been selected will be combined with the current good features. For each feature not in the current good feature list, we create a new data set which contains all the current good features and the newly added feature. We feed the newly created data set into a function named “cv\_loop” which does cross validation 10 times on the input data set and return a mean AUC score. By doing this, we know the effect of adding a feature into the current good feature list. We store this feature score mapping in a dictionary, so that by the end we know which feature has the best AUC score. At the end of each while loop iteration, the feature with the highest AUC score is added to the good feature list. The while loop will continue until there are no features that we can add to the good feature list to increase the AUC score.



## Hyperparameter Optimixation

Since we are using logistic regression here, we would want to find out the best C value to use. C parameter is the inverse of regularization strength. Smaller C values specify stronger regularization. To find out the best C values we perform cross validation with function “cv\_loop” on a range of preselected C values and pick the one with the best cross validation AUC score. The feature set we use in this step is only the selected good features not the entire feature set.

## Prediction

With the selected good features and C parameter, we are ready to build our logistic regression model. We first select all the good features from both the training set and the test and encode them using the OneHotEncoder() function. We then create the model with the best C value and train the model with the encoded training features, and ground truth. With the built model at hand, we go through all the encoded test data set and predict an Action for each data entry.

# dataset

## Data Set

The data set we use comes from the kaggle “Amazon.com – Employee Access Challenge” [4]. The data consists of real historical data collected from 2010 & 2011. Amazon Employees are manually allowed or denied access to resources over time. Two files are provided one named *test.csv* and the other *train.csv*. The train.csv file contains more than 30,000 rows. Each row has the ACTION(ground truth), RESOURCE, and information about the employee’s role at the time of approval. The test.csv file has about 60,000 rows and each row contains the request Resource Id and information regarding the requester’s role.

## Column Descriptions

|  |  |
| --- | --- |
| Column Name | Description |
| ACTION | ACTION is 1 if the resource was approved, 0 if the resources was not |
| RESOURCE | An ID for each resource |
| MGR\_ID | The EMPLOYEE ID of the manager of the current EMPLOYEE ID record; and employee may have only one manager at a time |
| ROLE\_ROLLUP\_1 | Company role grouping category id 1 (e.g. US Engineering) |
| ROLE\_ROLLUP\_2 | Company role grouping category id 2 (e.g. US Retail) |
| ROLE\_DEPTNAME | Company role department description (e.g. Retail) |
| ROLE\_TITLE | Company role business title description (e.g. Senior Engineering Retail Manager) |
| ROLE\_FAMILY\_DESC | Company role family extended description (e.g. Retail Manager, Software Engineering) |
| ROLE\_FAMILY | Company role family description (e.g. Retail Manager) |
| ROLE\_CODE | Company role code; this code is unique to each role (e.g. Manager) |

# Results

## AUC Score

After submitting our solution to the kaggle competition, we have obtained a AUC score of 0.90961 which is close to the best solution 0.92964 in the public leadership board on kaggle. Among all the steps we performed, the feature creation and greedy feature selection process greatly help in boosting the AUC score, it was 0.8745 without these two process.

We have also test our solution with different prediction models, such as Naïve Bayes, SVMs and RandomForestClassifier. Each model, we perform hyperparameter optimization. For Naïve Bayes, we use Multinomial Naïve Bayes and vary its smoothing parameter alpha. For SVM, we only use the linear kernel function, with different C, regularization parameters. They all have similar prediction outputs between 0.83 and 0.87. Among them, logistic regression has the best prediction output.

## Time

The following table categorized each step and the time it takes. The whole model building and prediction process takes about 7 minutes.

|  |  |
| --- | --- |
| Name | Time in Second |
| Read in training and testing data | 0.120898962021 |
| Relabeling data | 0.104212999344 |
| Feature creation | 7.87622594833 |
| Merge rare categories | 3.9881739616 |
| Encode all training features | 171.031939983 |
| Greedy feature selection | 186.534155846 |
| Hyperparameter Optimization | 4.56166815758 |
| Encode all the good features in both training and testing data sets | 57.8674 |
| Total: | 432.08467585788 |

# Discussion

I performed a search on google and found no papers regarding applying machine learning technique on automating resource access control. This report would be the first to formally define the problem of resource access control automation.

We can tell that the proposed approach works well with the specific data set from Amazon. In order to figure out if this is a generalized approach that can be apply to the general resource access control problem we need to apply our model on more data from different domains. Unfortunately, it’s very difficult to obtain more such data due to the fact that information regarding a company’s employees’ role and their provisioned access are considered top secret of the company and can post severe security threads to the company once known to the public.

Another question we would like to discuss is how useful this approach would be? Our algorithm has a AUC score of 0.909611. Intuitively, the AUC is the probability that an algorithm ranks a randomly chosen positive data point higher than a randomly chosen negative one. I don’t think this is good enough to fully automate access control because the cost of a false positive, granting access when it shouldn’t be granted, can be very high. Depending on the situation, if the cost of false positive is low we can reduce all the human involvement which would in term save a lot of time and money. If the cost of false positive is high, we can use the model to obtain a set of rules to figure out which roles should get which access, then someone will have to review the list to make sure the list is correct. From time to time, the list will need to be update and human factors are still required to review the list. The approach proposed in this paper can also be used in automating operating system access control. If we can define a role for each application, using our model we can automatically know what permission we can grant to a specific application.

There is future work that can be done to improve this model. First of all, the proposed approach is good with batch prediction, but it wouldn’t work well with incremental model update and online prediction. How can we build an online learning model? To further improve the AUC score, we can also experiment with ensemble learning techniques which is being used in the winning solution presented by Paul Duan [2] to the Kaggle Challenge.

# reference

1. Horbal, Miroslaw. 'Python Code To Achieve 0.90 AUC With Logistic Regression'. 2013. Web. 29 Nov. 2015.
2. Duan, Paul. 'Winning Solution Code And Methodology'. 2013. Web. 29 Nov. 2015.
3. Duan, Paul. 'Starter Code In Python With Scikit-Learn (AUC .885)'. 2013. Web. 29 Nov. 2015.
4. Kaggle.com,. 'Description - Amazon.Com - Employee Access Challenge | Kaggle'. N.p., 2015. Web. 29 Nov. 2015.

All these models have similar AUC scores ranging from 0.83 to 0.88. Among them, logistic regression has the best outcome.

We experiment with multiple machine learning models implemented in the sklearn library. Models like are

We need to find out what machine learning model to use that would give us the best result. The purpose of this project is not to implement any of those models because there are already excellent implementations in the scikit-learn library. Since this is a supervised machine learning problem, we have experimented with Logistic Regression, Naïve Bayes, SVMs and Decision Trees. The final result indicates that logistic regression has the best prediction result with our selected data set and that is the model we will be using in this solution. First, we will use this model to select features that truly help in improving our AUC score and then to make the prediction on the test data set.