CS 229: Predict Employee's Computer Access Needs in Company

Wenqi Xiang & Xin Zhou

Department of Electrical Engineering, Department of Computer Science, Stanford university

Introduction

When an employee starts work at a company, he or she first needs to get access to computer resources, like applications and web portals, to fulfill their role. To minimize the human involvement required to grant or revoke employee access, it's important to build a model on historical data to automatically determine an employee's access needs. The input to our algorithm is information about the employee's role at the time of approval. Due to the unique characters of our data set, with only 6% training examples labelling as 0, we use the one-hot Encoder to preprocess the dataset. Furthermore, use feature selection and cross validation to select good feature combinations. Lastly, we use Logistic Regression, SVM, and Naive Bayes as training models to output a predicted ACTION for a specific resource. We get a relatively high accuracy (around 87%) in labelling computer access needs.

Dataset & Data Preprocessing

Unbalanced Dataset

Our data from Kaggle consists of real historical data collected from 2010 & 2011. There are around 30000 samples in our data set, which is unbalanced with 96% labelled to be 1, and only 4% labelled to be 0. For each employee, there are 9 non-negative numeric features, which are:

1. # resource 2. # manager_ID 3. #role_rollup_1
4. # role_rollup_2 5. # department_name 6. #role_title
7. # role_family_description 8. # role_family 9. #role_code

Employees are manually allowed or denied access to resources over time. So, Label Action (y) is binary, to be 1 if the resource was approved or 0 if the resource was not.

One-hot Encoder

It generates a sparse matrix for each feature, for example, all values for a certain feature is 3, 2, 3, 1, 1, 10, 6, 6… and the largest value is 10. The feature vector is [3 2 3 1 1 10 6 6...]T, and the number of columns for its sparse matrix would be 10. The sparse matrix is as follows:

Method

Logistic Regression

We choose sigmoid function as the hypothesis.

$$y = h_\theta(x) = g(\theta^T x), \theta, x \epsilon R^9$$

Given the logistic regression model, we use Newton's method to fix θ. $\nabla_\theta l(\theta)$ is the the vector of partial derivatives of $l(\theta)$

$$\theta := \theta - H^{-1} \nabla_\theta l(\theta)$$

H is Hessian matrix, an n-by-n matrix (if we include the intercept term, H will be an n + 1-by-n + 1 matrix.), where

$$\mathrm{H}_{ij} = \frac{\partial^2 l(\theta)}{\partial\theta_i \partial\theta_j}$$

Support Vector Machines

We use SVM (L2-regularization) to train the dataset, and combine it with cross validation and feature selection to improve prediction accuracy.

Choose different kernel and regularization parameter C to test the g-means and accuracy.

Naive Bayes

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with Naive Bayes (NB) assumption of independence between every pair of features. Given a class variable y and a dependent feature vector Xi through Xn. Naive Bayes theorem can be stated as follows:

$$P(y|x_1, x_2 \cdots, x_n) = \frac{P(y)\,P(x_1, x_2 \cdots, x_n|y)}{P(x_1, x_2 \cdots, x_n)} = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, \cdots, x_n)}$$

Bernoulli Naive Bayes implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions. The decision rule for Bernoulli naive Bayes is based on:

$$P(x_i|y) = P(i|y)\,x_i + (1 - P(i|y))\,(1 - x_i)$$

then, we can get $\hat{y} = \mathrm{argmax}_y P(y) \prod_{i=1}^{n} P(x_i|y)$

Experiments & Results

All the below tables show seven metrics: true negative (TN), true positive(TP), false negative(FN), false positive(FP), the specificity, the sensitivity, and the g-means. The g-means metric is defined as:

$$g = \sqrt{specificity + sensitivity}$$

Table Results for **logistic regression**:

Logistic	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Training Data	1206	18779	54	1960	0.997	0.381	1.174
	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Testing Data	435	8970	202	1163	0.978	0.272	1.118

Table Results for **logistic regression using combined good features**. Use cross validation to test mean auc for each combination of features. It generates slightly better performance than using all the features:

Logistic_Select Features	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Training Data	1205	18768	55	1971	0.997	0.379	1.173
	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Testing Data	434	8979	203	1154	0.978	0.273	1.119

Result for the logistic regression using combined good features. Use cross validation to test mean auc for each combination of features. It generates slightly better performance than using all the features.

Accuracy results for small dataset(label0), large dataset(label1), and all dataset of training data and testing data. Left side is for logistic regression and right side is for logistic regression using combined good features.

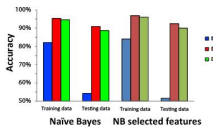
Table Results for **Naive Bayes**:

Naive Bayes	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Training Data	1035	19781	225	958	0.989	0.519	1.228
	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Testing Data	346	9208	291	925	0.969	0.272	1.114

Table Results for **Naive Bayes using combined good features**:

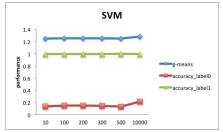
NB_Select Features	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Training Data	1060	20086	200	653	0.990	0.619	1.269
	TN_label0	TP_label0	FN_label1	FP_label1	Specificity	Sensitivity	g-means
Testing Data	329	9392	308	761	0.968	0.302	1.127

Accuracy Results for Naive Bayes and Naive Bayes using combined good features:



Accuracy for large dataset(label1) improved, but accuracy for small dataset(label0) is decreased when using Naive Bayes.

Accuracy and g-means results for **SVM:**



Accuracy for large dataset(label1) improved, but for small dataset(label0) is further decreased when using SVM.

Using different C regularization parameter didn't improve g-means and accuracy significantly.

References

[1] Beck J E, Woolf B P. High-level student modeling with machine learning[C]//Intelligent tutoring systems. Springer Berlin Heidelberg, 2000: 584-593.
[2] H. Zhang (2004). The optimality of Naive Bayes. Proc. FLAIRS.
[3] Kubat, M. Matwin, S. (1997). Addressing the Curse of Imbalanced Training Sets: One- Sided Selection. Proceedings of the 14th International Conference on Machine Learning.
[4] Kubat, M., Holte, R. Matwin, S. (1997). Learning when Negative Examples Abound. In Proceedings of ECML-97, 9th European Conference on Machine Learning.