Learning to Rate the Reliability of the Fire Safety System

Chang-Ming Xu
Northeastern University at Qinhuangdao, China

Ai-Zhong Wang
Gulf Security Technology Co., Ltd., China

Abstract

This paper concentrates on the problem of how to evaluate the reliability of the fire safety systems by machine learning methods. A fault of the fire safety system can be caused by more than dozens of factors. However, machine learning can do it better if enough data are expected to be available. Fortunately, the deployed fire safety system can naturally generate a huge amount of data from sensors.

We propose addressing such a reliability rating problem of the fire safety systems as a problem of supervised learning. Firstly, we redefine the concept of reliability by adding more variables. Secondly, we choose the boosting trees method for regression as an example in terms of the balance between effectiveness and interpretability. Lastly, we check the performance of the rating models with the testing data or validation set. The data we use are somewhat noisy and fairly incomplete on account of avoiding heavy cost. Even so, experiments show the learning methods are applicable and effective.

Keywords: Reliability, rating, machine learning, fire safety system

Introduction

In this paper, we will focus on the reliability of the fire safety system of a building, which is characterized by a real number between 0.0 and 1.0. Many researches concentrate more on the domain knowledge and make a solid foundation of relevant theory on the fire safety. The primary objective of reliability rating is the prediction of failures caused by monitoring equipment condition as a function of time [1]. However, we want to utilize more information to predict the reliability. There are two ways in machine learning: Generative methods and Discriminative methods.

On the one side, generative methods, such as fault tree, graphical model, are quite intuitive. E.g., [2] claims that risk cannot be determined unless the hazard is fully understood and described.

Numerous professional literatures on such a problem probably dive into revealing why failures happen and how they are generated. Then they build a model mimicking the actual scenario as similar as possible. However, only the experts can intensively study from the historical materials and discover some of patterns. Furthermore, to build a model to approximate to the real situation is also tedious and difficult, even for the experts. Unfortunately, such a system cannot escape from going wrong as time goes by and things keeps changing. Worst of all, the root of evil - overfitting, caused by the overly complex model encoded by the less accuracy prior knowledge of the experts. Overfitting means the final model fits the known data very well, but fails to predict any unseen data. Because the model learns not only the general rules, but also too much noise from training data.

Discriminative model of machine learning, as an alternative way, can overcome some defects of generative model, which deals the tasks with a black box so that we needn't concern about whether or not we mimic the things work exactly. For instance, by a vastly different playing style from the world championships of human being, Alpha Go defeats Lee Sedol 4:1 in the Google Deep Mind match [3]. Such a machine learning method does not care whether or not the model we used is analogical to the true agent governing the problem. In this way, we can simplify our model easily by taking out the unnecessary parameters to avoid overfitting. Deep Neural Network is a successful example of discriminative method, which in last decade has been making a range of impressive achievements in speech recognition, computer vision, natural language processing, etc. Gradient Boosting algorithm [5], such as XGBoost [6], is another example, which wins several influential champions in machine learning contests in most recent years.

Later in the article, we will firstly introduce how to define the goal of problem, and what if we use the theory of Gradient Booted Tree. After that we show how to use XGBoost to learn from historical data. Next we show how to check and analyze the result. At last, we summarize our work and look ahead into the future work.

Define the Task of Learning

[1] introduces the concept of reliability $\mathrm{R}(T)$ as follows

$$\mathrm{R}(T) = \int_T^\infty f(t)dt \qquad \text{(Eq. 1)}$$

Complexity and reliability are mutually contradictory. But the complexity isnot represented in Eq. 1. Machine learning is a branch of Computer Science to deal with the complex problem. Therefore, more variables added rather than time T only, machine learning algorithm is then adopted to solve the problem. To enhance the ability of the model and simplify the original problem, the following two assumptions are proposed in subsequent sections:

- First, consider the reliability as a function of both the time and the censored data. The censored data sometimes hints that some bad or good thing may occur. This assumption contributes to increasing more accuracy.
- Second, assume that the system fails, under the context of the fire safety system, only when fault alarms and/or false alarms are triggered. This assumption helps to simplify the analysis of the original problem.

Before proceeding to describe the task, we first introduce notations. We use tuple $(T, Z_{1:k})$ instead of a scaler T to generalize the definition of the reliability. Under the new definition, reliability depends on not only the time T, but also other random variables Z_1, Z_2, …, Z_k, which are denoted as Z, or $Z_{1:k}$ for short. Z_i is a single random variable for each i.

$$\mathrm{R}(T,Z) = \int_T^\infty \int_Z^\infty f(t,z)\, dz dt \qquad \text{(Eq. 2)}$$

As a result, a data set with 31-dimensionality and up to 8,000,000 observations is used by which to predict a real number representing the reliability of a building in half a month.

Gradient Boosting Algorithm and XGBoost library

GBM (Gradient boosting Machine) [5], described as algorithm 1, is an ensemble learning method, which collects each decision of a series of weak basis learning machine and forms a new decision as an output. To learn those parameters, we should set an object function

$$Obj(\Theta) = \sum_{t=1}^{N} L\big(y_t, f_M(x)\big) + \sum_{m=1}^{M} \Omega(f_m) \qquad \text{(Eq. 3)}$$

Consider L is a square error function, Ω is a regularization term. GBM compute an $Obj^{(t)}(\Theta)$, which is defined as

$$Obj^{(t)}(\Theta) = \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t), \text{ Where}$$

$$g_i = \frac{\partial \ell(y_i, \widehat{y_i}^{(t-1)})}{\partial \widehat{y_i}^{(t-1)}},\ h_i = \frac{\partial^2 \ell\big(y_i, \widehat{y_i}^{(t-1)}\big)}{\partial (\widehat{y_i}^{(t-1)})^2},\ \Omega(f_t) = \gamma T + \lambda \frac{1}{2} \sum_{j=1}^{T} w_j^2 \qquad \text{(Eq. 4)}$$

Here, w is the vector of leaves, and T is the number of leaves of regression machine. Then decent gradient method is applied to compute the parameters. For simplicity, we adopt squared error loss function. In each iteration, corresponding to line 3 up to line 7, algorithm firstly calculate the residuals of the current model $f_{m-1}(x)$. and then make the residuals fit furthest to a new basis learning machine $T(\mathrm{x}; \theta_m)$, which will be added to the previous model. There are many parameters that we can adjust in GBM.

```
Algorithm 1 Gradient Boosted Machine
Input: training set $T = \{(x^{(t)}, y^{(t)})\}_{t=1}^{N}$
Output: boosted machine $f_M(x)$
1.    Initialize $f_0(x) = 0$
2.    For m=1 to M do
3.        Compute residual $r_{mi} = y_i - f_{m-1}(x_i), i = 1, 2, \dots N$
4.        Learn a machine $T(\mathrm{x}; \theta_m)$ by fitting the residual $r_{mi}$
5.        Update $f_m(x) = f_{m-1}(x) + T(\mathrm{x}; \theta_m)$
6.        If converged
7.            Break
8.    Return boosted Machine $f_M(x) = \sum_{m=1}^{M} T(x; \theta_m)$
```

XGBoost [6], designed to be highly efficient, flexible and portable, is an implementation of GBM. In the meantime, it’s an optimized distributed gradient boosting library as well. XGBoost is easy to use because it provides interfaces with several script languages, including R and python. XGBoost needs less feature engineering than most of the previous algorithms. As we all know, feature engineering demands a superb skill while the XGBoost do not, which is apparently superior.

In GBM, each weak basis learning machine can be homogeneous, that is, they belong to the same type model, e.g., decision tree. A single decision tree can be constructed by greedy methods according to the criterion of information metrics, such as Entropy, Gini index, etc. The entropy of a random variable X can be calculated by Eq.5.

$$\mathrm{H(X)} = \mathbb{E}_{x\in X}\left[\log\frac{1}{(P(x))}\right] = \sum\nolimits_{x\in X} P(x)\log\frac{1}{(P(x))} \qquad \text{(Eq. 5)}$$

Gini index is another popular measure for purity of a set with nodes in different type, the merit of which is the low complexity for calculation.

Consider a problem of whether a man plays tennis on a specific day. some records are listed in Table 1. The decision tree learning will fit those data so that it can predict whether he plays tennis or not on a new day not listed in the table.

Table 1. The training set for learning a decision tree

Observation	Input			Output
	OUTLOOK	HUMIDITY	WINDY	PLAY
1	sunny	85	FALSE	NO
2	sunny	90	TRUE	NO
3	overcast	78	FALSE	YES
4	rain	96	FALSE	YES
5	rain	80	FALSE	YES
6	rain	70	TRUE	NO
7	overcast	65	TRUE	YES
8	sunny	95	FALSE	NO
9	sunny	70	FALSE	YES
10	rain	80	FALSE	YES
11	sunny	70	FALSE	YES
12	overcast	90	TRUE	YES
13	overcast	75	FALSE	YES
14	rain	80	TRUE	NO

Fig. 1 shows how to slice the collection into pieces according to information collected, in each of which will contain pure observations that belongs to the same classification. The decision tree will be constructed by selecting a best feature from candidate feature set to split the current node into subset with the same classifier.

An unseen new observation can be predicted when a decision tree having learned from the training data. E.g., there is a new observation x = (OUTLOOK = sunny, HUMIDITY = 20, WINDY = TRUE). The decision tree in Fig.1. answers the question by a series of the attribution value checking: OUTLOOK = sunny, and HUMIDITY <= 70. The predicted output falls into the leftmost leaf which is labeled with YES, that says, he will play tennis. See [7] for more detailed info.

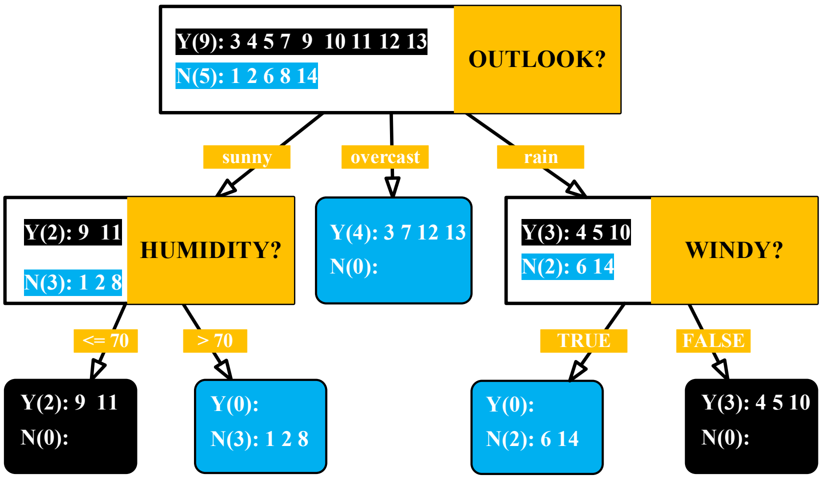


Fig.1. A decision tree learned from the training set in Table 1.

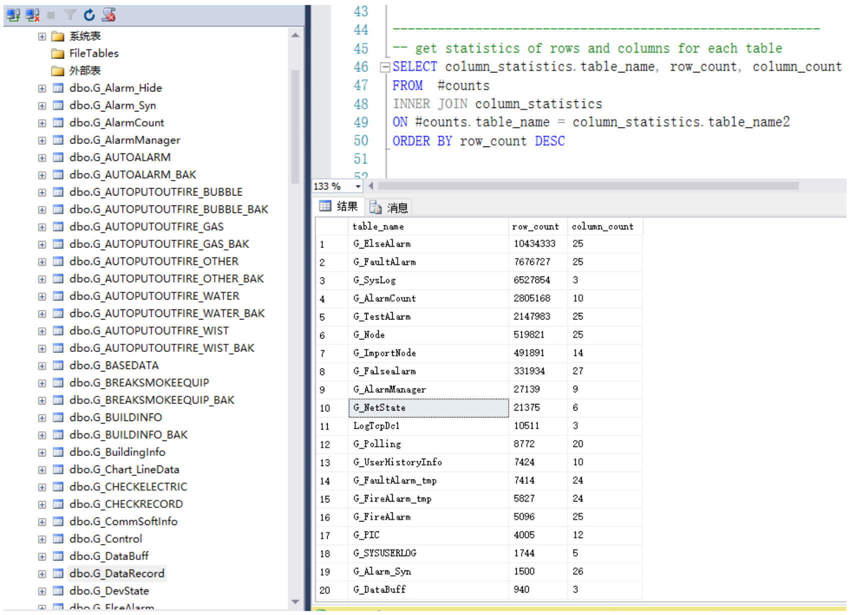
Prepare and Preprocess the Data

The data we used were collected from some commercial building all over China by Gulf Security Technology Co., Ltd. since 2011, see Fig.1. There are more than 130 tables in the database. The number of rows of the tables varies from 1 to 10,000,000, and that of columns varies from 3 to 70. The raw data files in the database are shown in Fig. 2.

Some of the tables describe the topological configuration of the objects composing the fire safety system, such as alarm ID, user ID, Monitor ID, Alarm code, Alarm SN, City and Region, Node ID, Operator, Building ID, Alarm time, Reaction time, etc. An issue of the data is that we do not collect enough information about the changes of each sensor state. The real time data is gathered only when an item turns into some failure state, including debugging, testing, alarming, etc.

Through monitoring 300 independent fire safety systems, we get 7,676,727 typical fault alarm observations, and 331,934 false alarm observations. These are the most important information relevant to the reliability of the fire safety systems.

Each component of the random multivariable Z is a single variable, some of which only have several possible values. Single variable, such as alarm_type, sub_area_type and monitor_type can be encoded by one-hot-encoding. E.g., the value of alarm_type varies from 1 to 3, then we can reconstruct it for each observation. One hot encoding, see Table 2., is a sort of sparse representation schema for a single variable. Different values of the newly adding variables can activate the learn algorithm pay attention to different subset of features.



```
系统表
FileTables
外部表
dbo.G_Alarm_Hide
dbo.G_Alarm_Syn
dbo.G_AlarmCount
dbo.G_AlarmManager
dbo.G_AUTOALARM
dbo.G_AUTOALARM_BAK
dbo.G_AUTOPUTOUTFIRE_BUBBLE
dbo.G_AUTOPUTOUTFIRE_BUBBLE_BAK
dbo.G_AUTOPUTOUTFIRE_GAS
dbo.G_AUTOPUTOUTFIRE_GAS_BAK
dbo.G_AUTOPUTOUTFIRE_OTHER
dbo.G_AUTOPUTOUTFIRE_OTHER_BAK
dbo.G_AUTOPUTOUTFIRE_WATER
dbo.G_AUTOPUTOUTFIRE_WATER_BAK
dbo.G_AUTOPUTOUTFIRE_WIST
dbo.G_AUTOPUTOUTFIRE_WIST_BAK
dbo.G_BASEDATA
dbo.G_BREAKSMOKEEQUIP
dbo.G_BREAKSMOKEEQUIP_BAK
dbo.G_BUILDINFO
dbo.G_BUILDINFO_BAK
dbo.G_BuildingInfo
dbo.G_Chart_LineData
dbo.G_CHECKELECTRIC
dbo.G_CHECKRECORD
dbo.G_CommSoftInfo
dbo.G_Control
dbo.G_DataBuff
dbo.G_DataRecord
dbo.G_DevState
dbo.G_ElseAlarm
```

```sql
-----------------------------------------------------------
-- get statistics of rows and columns for each table
SELECT column_statistics.table_name, row_count, column_count
FROM  #counts
INNER JOIN column_statistics
ON #counts.table_name = column_statistics.table_name2
ORDER BY row_count DESC
```

133 %

结果 消息

	table_name	row_count	column_count
1	G_ElseAlarm	10434333	25
2	G_FaultAlarm	7676727	25
3	G_SysLog	6527854	3
4	G_AlarmCount	2805168	10
5	G_TestAlarm	2147983	25
6	G_Node	519821	25
7	G_ImportNode	491891	14
8	G_Falsealarm	331934	27
9	G_AlarmManager	27139	9
10	G_NetState	21375	6
11	LogTcpDc1	10511	3
12	G_Polling	8772	20
13	G_UserHistoryInfo	7424	10
14	G_FaultAlarm_tmp	7414	24
15	G_FireAlarm_tmp	5827	24
16	G_FireAlarm	5096	25
17	G_PIC	4005	12
18	G_SYSUSERLOG	1744	5
19	G_Alarm_Syn	1500	26
20	G_DataBuff	940	3

Fig. 2. Some tables in the database.

Table 2. One hot encoding for the variable *alarm_type* uses another 3 new variables v1, v2 and v3 to substitute the original variable.

alarm_type	V1	V2	V3
1	1	0	0
2	0	1	0
3	0	0	1

Another important feature transformation is to split date into several new variables such as year, month, day, and hour, by which some special pattern, such as the seasonal law can be easily captured by the learning machine.

Selecting, Training and Evaluating a Model

We use the R package for executing the procedure of cross validation automatically. Cross validation helps to find the best fitting function out with more capacity to avoid the overfitting. We separate the data containing all 300 independent fire safety systems into 3 mutually disjoint subsets: training set, validation set, and test set. In view of stability of the result, this procedure should execute many times on different data sets. Fortunately, user only need to set a parameter, and XGBoost will do it automatically.

Conclusion and Outlook

In this paper, we elaborate on the reliability problem, and point out the machine learning methods will help to enhance the performance of the predictor. Because thousands of machine learning algorithms and more tricks can be candidate, there is capacity for expansion.

References

[1] Joglar, Francisco. “*Reliability, availability, and maintainability. SFPE Handbook of Fire Protection Engineering*”. Springer New York, 2016. 2875-2940.

[2] Ericson, Clifton A. “*Hazard analysis techniques for system safety*”. John Wiley & Sons, 2015.

[3] Silver, David, et al. “*Mastering the game of Go with deep neural networks and tree search*”. Nature 529. 7587 (2016): 484-489.

[4] Bahr, Nicholas J. “*System Safety Engineering and Risk Assessment: A Practical Approach*”. CRC Press, 2014.

[5] Friedman J H. “*Greedy function approximation: a gradient boosting machine*”. Annals of statistics, 2001: 1189-1232.

[6] Tianqi Chen and Carlos Guestrin. “*XGBoost: A Scalable Tree Boosting System*”. In 22nd SIGKDD Conference on Knowledge Discovery and Data Mining, 2016: 785-794.

[7] Cha Sung-Hyuk, Tappert Charles C. "*A Genetic Algorithm for Constructing Compact Binary Decision Trees*". Journal of Pattern Recognition Research. 2009, 4 (1): 1–13.