Allstate Claims Severity

Description

This report focuses on the problem-solving for a personal insurer, Allstate. It uses regression technique to predict claims severity, based on hundreds of known variables. The data contains 116 categorical variables, and 14 continuous variables. In the training set, loss is provided, which is used for learning model parameters. The size of data is pretty large with a dimension of 188318 and 132.

Solution

To solve this problem, I use xgboost package in R to do regression and model selection. Extreme Gradient Boosting is an efficient implementation of gradient boosting framework. The package includes efficient linear model solver and tree learning algorithms.

In this problem, we need to predict $\hat{y}_i$ given x_i, which is $\hat{y}_i=\sum_j W_j * x_{ij}$, and learn model parameters θ. The objective function is given by

Obj(θ)= L(θ)+ G(θ), where L is the training loss function, and G is the regularization.

Suppose we have K trees, then the objective function becomes:

$\text{Obj}=\sum l(yi, \hat{y}_i)+ \sum G(f_k)$, where $=\sum f_k(x_i)$.

Since f_k are trees instead of numerical vectors, we cannot use stochastic gradient descent method. The solution is using boosting. First we initialize a constant prediction for $\hat{y}_i^{(0)}=0$, and then do the following iterations:

$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$

$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$

…

$\hat{y}_i^{(t)} = \sum f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$

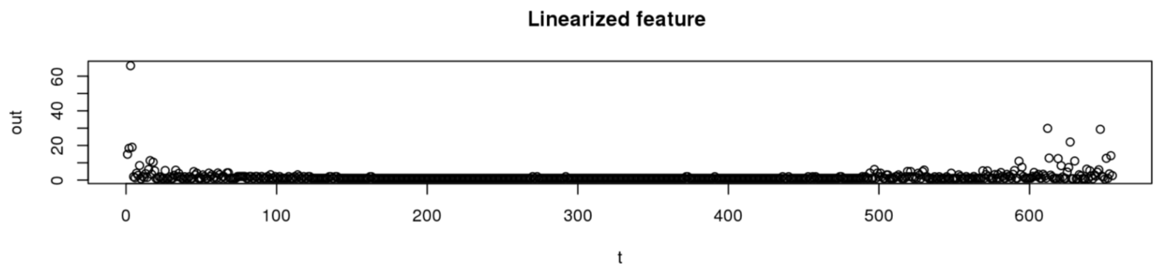
Then our objective function is $Obj^{(t)} = \sum l(yi, \hat{y}_i^{(t-1)} + f_t(x_i)) + \sum G(f_t)$ + constant. Our goal now is to find f so that the objective function is minimized. In R, we can easily do this by “xgb.cv”, which seeks the best iteration that has the smallest error. It is a process of model selection. We stop the cross validation process when test error has not improved in 100 rounds.

```r
#regression parameter#
para_learning = list(
  seed = 0,
  colsample_bytree = 0.7,
  subsample = 0.7,
  eta = 0.075,
  objective = 'reg:linear',
  max_depth = 6,
  num_parallel_tree = 1,
  min_child_weight = 1,
  base_score = 7
)

xgboosting_evaluation <- function (yhat, data_train) {
  y = getinfo(data_train, "label")
  err= mae(exp(y),exp(yhat) )
  return (list(metric = "error", value = err))
}

#do iteration to find best parameter#
error = xgb.cv(para_learning,
             data_train,
             nrounds=750,
             nfold=4,
             early_stopping_rounds=100,
             print_every_n = 10,
             verbose= 1,
             feval=xgboosting_evaluation,
             maximize=FALSE)

best_nrounds = error$best_iteration
gbdt = xgb.train(para_learning, data_train, best_nrounds)
prediction = fread("sample_submission.csv", colClasses = c("integer", "numeric"))
prediction$loss = exp(predict(gbdt,data_test))
```

Method from Other Community

I learned the method benchmark from Dmi3kno. He selected minimum value on a certain region surrounding the point in question and then return logical value of whether the current point is within (TRUE) or above (FALSE) the fragment of the lowest curve, formed by the points evaluated in the window. After separating the curves, he used lm function to learn the model parameters, and then did the following predictions for loss. Assumption is made that the points residing on the lowest curve have a scaling coefficient of 1 (should be decoded into the value of 1). The strength is that we can linearize the data first and then run the easy lm function to learn the model.

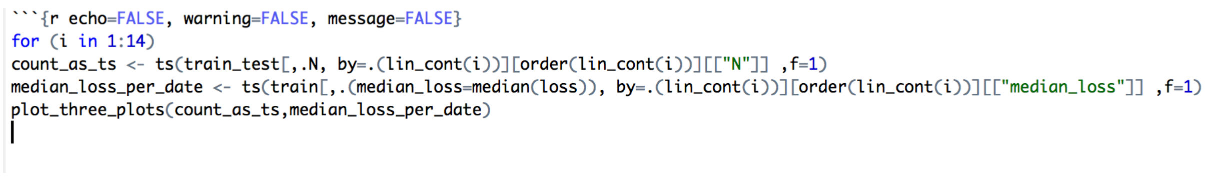
In his code, he plotted 10 curves for each feature, and then chose the lowest curve. Take cont1 as an example, the output show the linearity of data.



Frankly speaking, this method is much harder and time-consuming for implementation, but stronger for prediction, because the author introduced time series trend, so that prediction is not only based on the current data, but also the previous data. Compared to my solution, I think mine is much easier when running the results, but maybe less accurate, since I did not do the linearity.

Improvements

Actually speaking, his method is really good for me so far. The only thing that I tried is to add the categorical data, but failed. The reason is that we cannot plot the linearity for those data. I did one simple and successful process is loop, so that codes can be reduced.



~~~
```{r echo=FALSE, warning=FALSE, message=FALSE}
for (i in 1:14)
count_as_ts <- ts(train_test[,.N, by=.(lin_cont(i))][order(lin_cont(i))][["N"]] ,f=1)
median_loss_per_date <- ts(train[,.(median_loss=median(loss)), by=.(lin_cont(i))][order(lin_cont(i))][["median_loss"]] ,f=1)
plot_three_plots(count_as_ts,median_loss_per_date)
~~~

It produces the same result, but fewer codes are needed.

Thinking

By doing this project, I know more about R in detail. I learned regression analysis in another course, but the direct lm function dose not work here, since the assumptions for linear regression are not satisfied. I tried to use BIC for model selection, but it seems to make this problem more complicated, so I turned to xgboosting, which I touched somehow before. Compared to python, I am more familiar with R, that is why I chose to use R to do the analysis, even though it works slowly for some certain problems. In this project, I get the opportunity to touch the large data. To some point, it is an easy task, since there are no missing data.