Feature-based methods for Semantic Modeling
CS420 Coursework: Text Classification

ZhiTian Xu
monty,515030910636
Shanghai Jiao Tong University
voidxzt@sjtu.edu.cn

April 23, 2017

Abstract

In terms of a classification model for recommending selected acticles,I think of an easy and fast feature-based model called Logistic Regression at the beginning and try to adjust the parameters to enhance its performance.When I reach the bottleneck and can't improve the model any more,I try to ensemble it with other linear models or boosting methods,getting a little improvement.Afterwards,another strong and powerful model named XGBoost comes up to me. By ensembling it with Logistic Regression and Randomforest and reasonably allocating respective weights,I get the best score I have ever got.

1 Introduction

1.1 Background

As aggregators, online news portals face great challenges in continuously selecting a pool of candidate articles to be shown to their users.
Typically, those candidate articles are recommended manually by platform editors from a much larger pool of articles aggregated from multiple sources. Such a hand-pick process is labor intensive and time-consuming. In this task, I study the editor article selection behavior and propose a learning by demonstration system to automatically select a subset of articles from the large pool.

Table 1: Notations and descriptions

Notation	Description
tot_word	The total number of words extracted from all the articles.
θ	The parameter of Logistic Regression.
$\boldsymbol{x}$	The pre-defined value of positive user response.
y	The true label of user response.
$L(\boldsymbol{x})$	The loss function of the objective.
p	The predicted probability $Pr(\hat{y} = 1\|\boldsymbol{x})$.

1.2 Formulation

First of all,a piece of software used for splitting words is needed since the input data is presented in the form of articles.I choose jieba and use the TF-IDF weight function(I use Textrank at first,but the practice proves that TF-IDF is much better than Textrank) to deal with articles.After handling all the articles,I calculate *tot_word* and regard each article as a *tot_word*-dimension vector,the value of each dimension is determined by its TF-IDF weight.Therefore,I get a feature matrix.Afterwards I divide the train set into two parts with a ratio of $4:1$,the former for training the model and the latter for calculating AUC.As for Logistic Regression,we have

$$Predict\ probability: Pr(\hat{y}=1|\boldsymbol{x}) = \sigma(\theta^T\boldsymbol{x}) = \frac{1}{1+e^{-\theta^T\boldsymbol{x}}}$$

$$Cross\ Entropy\ loss\ function: L(\boldsymbol{x}) = -ylog\sigma(\theta^T\boldsymbol{x}) - (1-y)log(1-\sigma(\theta^T\boldsymbol{x}))$$

2 Methodology

Firstly,I don't use the TF-IDF weight directly.For each dimension of each vecotr,I rewrite its value as a quadratic function in order to magnify the differences.I use Logistic Regression and XGBClassifier packaged in sklearn and adjust parameters to make each model best.I also try bagging methods,Adaboost methods,decision tree models and random forest models.After that,I use the linear combination of these models to uplift my performance.

3 Experiments and Results

As for the quadratic function mentioned above,I test the utility of the quadratic function in Logistic Regression.With careful manual adjustment,$f(x) = (x+1.0)*(x+0.8)$ beats other quadratic functions.

Table 2: Parameters I adjust in Logistic Regression

Parameters	Description
penalty	Used to specify the norm used in the penalization.
C	Inverse of regularization strength; must be a positive float.

As for the parameters I adjust in Logistic Regression,$penalty = l1$ not only runs faster but also gets better grades than $penalty = l2$.I guess the relationship between performance and C is unimodal,so I just decrease C in a random step until find a rough local minimum,which is about $C = 0.12$

Table 3: Parameters I adjust in Bagging

Parameters	Description
base_estimator	The base estimator to fit on random subsets of the dataset.
n_estimators	The number of base estimators in the ensemble.
max_samples	The number of samples to draw from X to train each base estimator.
max_features	The number of features to draw from X to train each base estimator.

I also try bagging method,using Logistic Regression as basic estimator.I deduce the relationship between each parameter and performance is unimodal.With careful manual adjustment,$n\_estimators = 4, max\_samples = 0.9, max\_features = 0.9$ seems to be best,but only improve a little bit compared to pure Logistic Regression.And it doesn't play a role in my final result.

Table 4: Parameters I adjust in Adaboost

Parameters	Description
base_estimator	The base estimator from which the boosted ensemble is built.
n_estimators	The maximum number of estimators at which boosting is terminated.

For the Adaboost method,*base_estimator* I use is decision tree.However,Adaboost not only runs slow,maybe ten times slower than random forest,but also gets a similiar performance as random forest.No matter how I adjust *n_estimators*,the best AUC I can get is at most 80,so I just abandon it.

Besides,I give a try to decision tree model,but random forest model is far beyond it,with the best AUC of decision tree less than 75,so I abandon it.

Table 5: Parameters I adjust in Randomforest

Parameters	Description
max_depth	The maximum depth of the tree.
n_estimators	The number of trees in the forest.

Apart from that,Randomforest is also a good model.Since my feature matrix is really big,3 or 4 hours is needed to run a test.I suppose that the relationship between each parameter I adjust and performance is unimodal.So with careful manual adjustment,$max\_depth = 6, n\_estimators = 300$ seems to be best.And I integrate it with my final result.

Table 6: Parameters I adjust in XGBClassifier

Parameters	Description
max_depth	Maximum tree depth for base learners.
learning_rate	Boosting learning rate (xgbs eta).
n_estimators	Number of boosted trees to fit.
gamma	Minimum loss reduction required to make a further partition on a leaf node of the tree.
min_child_weight	Minimum sum of instance weight(hessian) needed in a child.
subsample	Subsample ratio of the training instance.
colsample_bytree	Subsample ratio of columns when constructing each tree.

As for the parameters I adjust in XGBClassifier,$gamma, max\_depth, min\_child\_weight, n\_estimators$ are used to control the complexity of the model,$colsample\_bytree, subsample$ are used to introduce randomization.I guess the relationship between each parameter and performance is unimodal.With careful manual adjustment, $learning\_rate = 0.05, n\_estimators = 250, gamma = 0.0, colsample\_bytree = 0.8, subsample = 0.8, min\_child\_weight = 7, max\_depth = 42$ performs best.

Finally,my result is a linear combination of Logistic Regression,XGBoost and Randomforest,and respective weights are $0.08, 0.88, 0.04$.

4 Discussion and Future works

From my perspective,with more different types of models and features involved,linear combination of different models will be more powerful.However,it will eventually reach the limit because it simply uses linear combination and neglect the effect of the feature.If we integrate different models and features with neural network,it's likely to achieve a better performance.But it's rather time-consuming since you have to train large quantities of models and each requires you much time to adjust its parameters.
With regard to future works,I think classification works like item classification,movie classification and clothes classification will be more general.And these classification problems are not linearly separable and are too complicated that easy feature-based methods can't effectively make reasonable decisions.In my opinion,tree models and network models will be more welcome because they can deal with complicated situations that human beings can't imagine.

5 References

1.$http://xgboost.readthedocs.io/en/latest/python/python\_api.html$
2.$http://www.2cto.com/kf/201607/528771.html$