COMPSCI5014 - Machine Learning (Level M)

Assessed Coursework

Submission Report – UoG Kaggle Competition 2018

School of Computing Science

Presented by

Ernst Werner (MSc Data Analytics) – 2383746W

Abhishek Sharma (MSc Data Science) – 2419831S

Geetha Aredla (MSc Data Analytics) – 2418643A

Aditya Hasurkar (MSc Data Analytics) – 2396064H

Presented to

Dr Ke Yuan (course convenor)

November 24th, 2018

Introduction

As part of the coursework, the goal is to train two given datasets, namely X_train and X_test in order to maximise the overall predictive accuracy for our training data y_train. In satisfaction of the task requirements, we provide two model predictions each for the Regression and Classification problems respectively. Overall, we followed a stepwise approach based on repeated accuracy measuremets of different classifiers and regressors as supported by sklearn in Python. Through consecutive parameter adjustments, it was possible to reach higher levels of accuracy in the involved metrics, but also to express more general models. Apart from our analysis, Kaggle submission scores gave clear indication of highly relevant predictions with satisfactory error margins in our chosen implementation.

The provided datasets contain the following information.

Regression dataset:

X_train	**X_test**	**y_train**
168 rows (observations)	41 rows (observations)	168 rows (output)
6 predictor variables	6 predictor variables	1 predictand
type: numeric	type: numeric	type: numeric

Cassification dataset:

X_train	**X_test**	**y_train**
168 rows (observations)	41 rows (observations)	168 rows (output)
6 predictor variables	6 predictor variables	1 predictand
type: numeric	type: numeric	type: numeric

Regression Problem

In the regression task, we have particulary referred to the concepts of Random Forest Regressors, Random Forest Classifiers and Support Vector Regression. Surprisingly, our fitted data indicates that despite good predictive outcomes under SVR, Random Forest Classifiers outperformed these. Given that Random Forest Classifiers are more extensively used in classification contexts this result seems even more interesting. The high predictive accuracy of Random Forest Classifiers in particular made this algorithm the prefferred choice in the regression framework.

Algorithm choice 1 - Support Vector Regression

As a subcategory of Support Vector Machines, SVR is most often applied to the context of linear regression models , where it is of interest to restrict a deviance margin (lower the cost) to the target variable y. Being a conceptual extension to the classical Least Squares estimate, SVR respresents a convex optimization problem where underfitting or overfitting can be adressed by inclusion of a regularization term such as found in Lasso or Ridge regressions. The general expression for SVR can be defined as follow:

Given a training pair $(x_i,\ y_i)$ where $x_i \in R^n$, $y_i \in R$ for $i = 1, \dots, l$. Then a linear Support Vector Regression finds a model w s.t. $w^T x_i$ is minimized in distance to the target value y_i.

<u>It solves the function dependent on w as follow.</u>

$$\underbrace{min}_{w}\ f(w) \quad , \quad where \quad f(w) = \frac{1}{2} w^T w + K \sum_{i=1}^{l} \varepsilon_{\epsilon}\,(w;\ x_i,\ y_i)$$

with $K > 0$ being a regularization parameter and,

$$\varepsilon_{\in}(w;\ x_i,\ y_i) = \begin{cases} max(|w^T x_i - y_i| - \epsilon, 0) \\ max(|w^T x_i - y_i| - \epsilon, 0)^2 \end{cases} \quad \text{or}$$

Although clearly applicable to non-linear cases, in our particular example SVR delivered best results under linearity consistent with a hyperplane. Since the number of observations in our datasets is small, complexity issues based on primal input dimension are low as opposed to higher dimensional cases (such as Polynomial Kernels or RBF Kernels), which require a high number of support vectors.

Apart from being computationally simpler, linearity conditions under SVR have proven to be advanatgous in effectively generalizing unseen data. Since SVR only requires a subset of the training data, learning rates were generally consistent troughout our model runs.

Algorithm choice 2 – Random Forest Classifier

Random Forest belongs to the class of supervised learning algorithms. Several combinations of different decision trees are used for nonlinear multiple regression as in our example. Each leaf node contains a probability distribution for the continuous output variable. Based on the provided data sample, it generates from the training data. The key aspect is to control the variance by classifying the data into decision trees, the latter are then averaged and thus the variation in the model will be reduced. To train decision trees, we can refer Bootstrap aggregation.

The general idea of the bagging method is that a combination of learning models increases the overall result. The training algorithm for random forests applies the general technique of aggregating tree learners. Given a training set $X = x_1, ..., x_n$ with responses $Y = y_1, ..., y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $b = 1, ..., B$:

Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.

Train a classification or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

Additionally, an estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f}\,)^2}{B-1}}.$$

In simple words, random forest models build many (sometimes even thousands) individual trees (trees can be regressions and classifications) that are built using some fraction of the data. Then they predict the output by combining the outputs of these trees, by voting or average or anything suitable.

Performance Analysis and Metrics

Despite achieving accurate results under SVR, we observed the Random Fofrest Classifier to be superior in almost all runs,……..

<u>Statistical results</u>

Algorithms	R-squared Score (X_train)	R-squared Score (X_test)
Support Vector Regression (SVR)	0.88095	0.71428
Random Forest Classifier (RFC)	0.88690	0.85714

As the above table inidcates, the R-squared values achieved by the RFC-Algorithm outperform those obtained by SVR. Thus, we are able to explain more variation in our data using RFC. It is worth mentioning that with different choices of random state values, the accuracy scores fluctuate without any specific pattern. However, in some cases there is a negative relationship between the accuracy level and error score, so it represents a tradeoff problem.

Classification Problem

Algorithm 1 – Random Forest Classifier

Random Forest is a strong algorithm to train the model in an early development process, in order to judge its performance. It provides a good indicaton of the importance it assigns to your observations.
The forest error rate is dependent on:

- Correlation between any two trees.
- The strength of each individual tree in the forest.

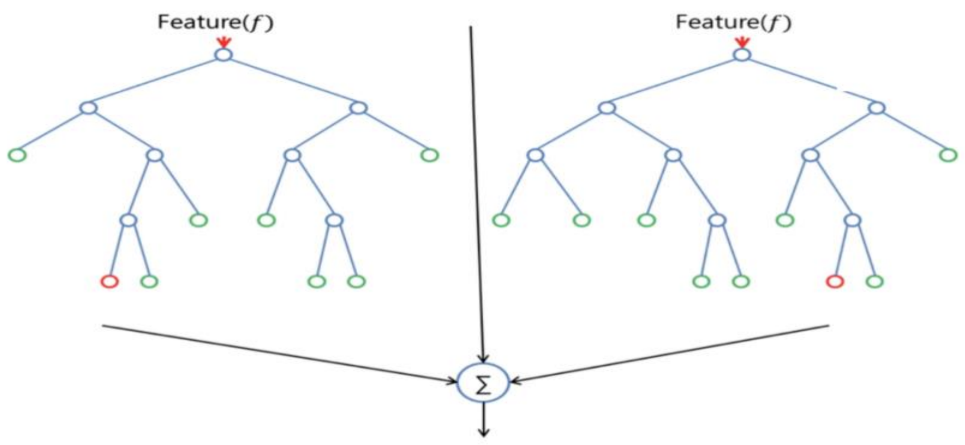
Correlation is directly proportional to the forest error rate whereas the strength is indirectly proportional to it. Therefore, we aim to achieve higher strength while having a lower correlation between the trees.

Reducing m reduces both the correlation and the strength. Increasing it increases both. Somewhere in between is an "optimal" range of m - usually quite wide. (where m represents a number m<<M-input variable is specified such that at each node, m variables are selected at random out of the M-input variables and the best split on these m is used to split the node.) The value of m is held constant during the forest growing.

Random Forests are also very hard to beat in terms of performance. Of course you can probably always find a model that can perform better, like a neural network, but these usually take much more time in the development. And on top of that, they can handle a lot of different feature types, like binary, categorical and numerical.

Overall, Random Forest is a (mostly) fast, simple and flexible tool, although it has its limitations.

RF Classification - a random forest would look like with two trees:



Algorithm 2 – Extra Trees Classifier

Extra Trees classifier (Extremely randomized trees) is an alternative form of a random forest algorithm where at each step the entire dataset is used and decision boundaries are picked up at random. It always computes the splits and thresholds at random for each feature and the best of the generated thresholds are used for splitting the trees. This algorithm allows to reduce the variance of the model compared to other trees and also reduces computational burdens.

The main objective of extra trees is to further randomize tree building in the context of numerical input features, where the choice of the optimal cut-point is responsible for a large proportion of the variance of the induced tree. It often provides accuracy because of its smoothing. Extra trees are generally cheaper to train from a computational point of view but can grow much bigger. Sometimes it can generalize better than Random Forests.

The ExtraTreesClassifier class implements an estimator that fits a number of randomized decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. We have parameters n_estimators(The number of trees in the forest), max_features , bootstrap and also random_state and we calculate the score of the classifier of our model using cross_val_score.

Algorithmic steps

<u>Split a node(Dataset)</u>

In: Dataset

Out: if split [n <nc] do nothing nc is random cut.

– If Stop split(dataset) is TRUE then return nothing

– Otherwise select K attributes {n1,...,nK } among the dataset

– Draw K splits {d1,...,dK }, where split = Pick a random_split(dataset, ni), for i = 1,..., K

– Return a split and max of score(dataset,split)

<u>Random_split(Dataset,n)</u>

In: Subset S and attribute n

Out: n split

– Let nS max and nS min value of S

– Draw a random cut-point nc uniformly in [nS min, nS max]; – Return the split [n < nc].

<u>Stop split(Dataset)</u>

Input: subset S

Output: a Boolean

– If |S| < nmin or constant output or attributes then return TRUE;

– Otherwise, return FALSE.

Performance Analysis and Metrics

Conclusion

Literature Review

ARTICLES

Mikolov, T. et al (2013) Distributed Representations of Words and Phrases and their Compositionality. NIPS

What is the problem;

- inherent limitation of word representations is their indifference to word order and their inability to represent idiomatic phrases that are not compositions of the individual words

What is the state-of-the-art before the paper;

- Specialists in the field of neural network based language models (RNNLM) have relied on the Skip-gram model as an efficient method for learning high-quality distributed vector representations that capture a large number of precise syntactic and semantic word relationships
- Original Skip-gram model: neural network architecture for learning word vectors that does not involve dense matrix multiplications, therefore making it an efficient method of training

What is new about the solution;

- They expand the research frontier in neural networks analysis and computation of distributed word representations by presenting several extensions to the original Skip-gram model which result in improved quality of the vectors and of the training speed
- extension that allow to go from word based to phrase based models
- one extension involved the subsampling of frequent words during training which resulted in a significant speedup (around 2x - 10x), and improved accuracy of the representations of less frequent word

Validity of supporting evidence