Project 4: Regression Analysis

DUE Wednesday, March. 6, 2019 by 11:59 pm

Introduction

Regression analysis is a statistical procedure for estimating the relationship between a target variable and a set of potentially relevant variables. In this project, we explore basic regression models on a given dataset, along with basic techniques to handle over-fitting; namely cross-validation, and regularization. With cross-validation, we test for over-fitting, while with regularization we penalize overly complex models. detect

prevent

Dataset 1

We use a Network backup Dataset, which is comprised of simulated traffic data on a backup system over a network. The system monitors the files residing in a destination machine and copies their changes in four hour cycles. At the end of each backup process, the size of the data moved to the destination as well as the duration it took are logged, to be used for developing prediction models. We define a workflow as a task that backs up data from a group of files, which have similar patterns of change in terms of size over time. The dataset has around 18000 data points with the following columns/variables:

- Week index
- Day of the week at which the file back up has started
- Backup start time: Hour of the day
- Workflow ID
- File name
- Backup size: the size of the file that is backed up in that cycle in GB
- Backup time: the duration of the backup procedure in hour

Problem Statement

1. **Load the dataset**. You can download the dataset from this link. To get an idea on the type of relationships in your dataset:

(a) For the first twenty-day period (x-axis unit is day number) plot the backup sizes for all workflows (color coded on the y-axis).

(b) Do the same plot for the first 105-day period.

(c) Can you identify any repeating patterns?

2. **Predict** the backup size of a file given the other attributes. We use all attributes, except `Backup Time`, as candidate features for the prediction of backup size.

   We will try different feature sets, as well as different encoding schemes for each feature and in combination. For example, each of the following five features is a categorical variable: `Day of the Week`, `Hour of the Day`, `Work-flow-ID`, `File Name`, and `Week Number`.

   For each categorical variable, we could convert it into a one dimensional numerical value. For example, `Day of the Week` variable could take values $1, \cdots, 7$ corresponding to Monday through Sunday. Similarly, the `Hour of the Day` could be encoded as $1, \cdots, 24$. We will refer to this as a scalar encoding.

   For each categorical variable that takes one of M values we can also encode it as an M dimensional vector, where only one entry is 1 and the rest are 0's. Thus for the `Day of the Week`, Monday could be encoded as $[1, 0, 0, 0, 0, 0, 0]$ and Sunday as $[0, 0, 0, 0, 0, 0, 1]$. We will refer to this encoding as one-hot-encoding.

   For part (a)-(e), in each part you will use one type of models to predict the backup sizes.

   In each part, you need to **report training and test Root Mean Squared Error (RMSE) from 10-fold cross validation** as the basic evaluation of the performance. That is, for each fold you get two numbers: training RMSE and test RMSE.

   In addition, you need to **plot the following two types of plots** using the whole dataset for **each part** with the best parameters and hyper-parameters found for the model (use RMSE as the standard for the selection of the "best"):

   i Plot fitted values against true values as **scatter plots**

   ii Plot residuals versus fitted values as **scatter plots**

   These plots visualize how well your models fit the data.

   (a) Fit a **linear regression model**. We use ordinary least square as the penalty function.

   $$\min_{\beta} \|Y - X\beta\|^2$$

   where the minimization is on the coefficient vector β.

   Convert each categorical feature into one dimensional numerical values using scalar encoding (e.g. Monday to Sunday can be mapped to 1-7), and then directly use them to fit a basic linear regression model.

   (b) Use a random forest regression model for the same task.

   Feature importance in random forest algorithm: during the training process, for each node, a branching decision is made based on only one feature that minimized a chosen measure of impurity. For classification, it is typically Gini impurity or information gain (entropy) and for regression task, it is variance (see lecture notes). The importance of each feature will be the averaged decreased variance for each node split with this feature in the forest and weighted by the number of samples it splits.

Out of bag error: In the random forest regression, since we use bootstrapping, it's easier and faster to evaluate the generalization ability. Each tree is only built from a subset of the data set (because of sampling) so the data points that are left out can be used as the test set. One can then define prediction-RMSE for each tree and then average over all trees. In a `sklearn`'s random forest regression object, `oob_score_` will return out-of-bag R^2 score, so you can calculate $1 -$ `oob_score_` as the **out of bag error**.

Set the parameters of your model with the following initial values.

- Number of trees: 20
- Depth of each tree: 4
- Bootstrap: True
- Maximum number of features: 5

Note that a Random Forest model can handle categorical variables without the need of one-hot or scalar encoding.

i. Report Training and average Test RMSE from 10 fold cross validation (sum up each fold's square error, divide by total number of data then take square root) and Out Of Bag error you get from this initial model.

ii. Sweep over number of trees from 1 to 200 and maximum number of features from 1 to 5, plot

   1) out-of-bag error (y axis) against number of trees (x axis)

   2) average test-RMSE (y axis) against number of trees (x axis)

iii. Pick another parameter you want to experiment on. Plot similar figure 1 and figure 2 as above. What parameters would you pick to achieve the best performance?

iv. Report the feature importances you got from the best random forest regression you find.

v. Visualize your decision trees. Pick any tree (estimator) in best random forest (with max depth=4) and plot its structure, which is the root node in this decision tree? Is it the most important feature according to the feature importance reported by the regressor?

(c) Now use a neural network regression model (one hidden layer) with all features one-hot encoded. Parameters:

- Number of hidden units. Try $[2, 5, 10, 50, 100, 150, 200, \ldots, 600]$.
- Activation function ('relu', 'logistic', 'tanh')

Plot test-RMSE vs the number of hidden units for each activation function. Report the best combination.

(d) Predict the Backup size for each of the workflows separately.

i. Using linear regression model. Explain if the fit is improved?

ii. Try fitting a more complex regression function to your data. You can try a polynomial function of your variables. Try increasing the degree of the polynomial to improve your fit. Again, use a 10 fold cross validation to evaluate your results. Plot the average train and test RMSE of the trained model against the degree of the polynomial you use. Can you find a threshold on the degree of the fitted polynomial beyond which the generalization error of your model gets worse? Can you explain how cross validation helps controlling the complexity of your model?

   (e) Use k-nearest neighbor regression and find the best parameter.

3. Compare these regression models you have used and write some comments, such as which model is best at handling categorical features, which model is good at handling sparse features or not? Which model overall generates the best results?

Dataset 2: Boston Housing Dataset

This dataset concerns housing values in the suburbs of the greater Boston area and is taken from the StatLib library which is maintained at Carnegie Mellon University. There are around 500 data points with the following features:

- CRIM: per capita crime rate by town
- ZN: proportion of residential land zoned for lots over 25,000 sq. ft.
- INDUS: proportion of non-retail business acres per town
- CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- NOX: nitric oxides concentration (parts per 10 million)
- RM: average number of rooms per dwelling
- AGE: proportion of owner-occupied units built prior to 1940
- DIS: weighted distances to five Boston employment centers
- RAD: index of accessibility to radial highways
- TAX: full-value property-tax rate per $10,000
- PTRATIO: pupil-teacher ratio by town
- B: $1000(\mathrm{Bk} - 0.63)^2$ where Bk is the proportion of blacks by town
- LSTAT: % lower status of the population
- MEDV: Median value of owner-occupied homes in $1000's

1. Load the dataset. You can download the dataset from this link.
2. Fit a linear regression model:
   (a) Set MEDV as the target variable and the other attributes as the features and ordinary least square as the penalty function.

(b) Perform a 10 fold cross validation, analyze the significance of different variables with the statistics obtained from the model you have trained, and the averaged Root Mean Squared Error (RMSE), and plot 1) fitted values against true values as scatter plots using the whole dataset; 2) residuals versus fitted values as scatter plots using the whole dataset.

3. In this part, we try to control overfitting via regularization of the parameters.

   (a) Perform one-hot encoding on all features, then fit and test. Report the test RMSE and train RMSE.

   (b) You should have found obvious increases in test RMSE compared to training RMSE in some combinations, can you explain why this happens?

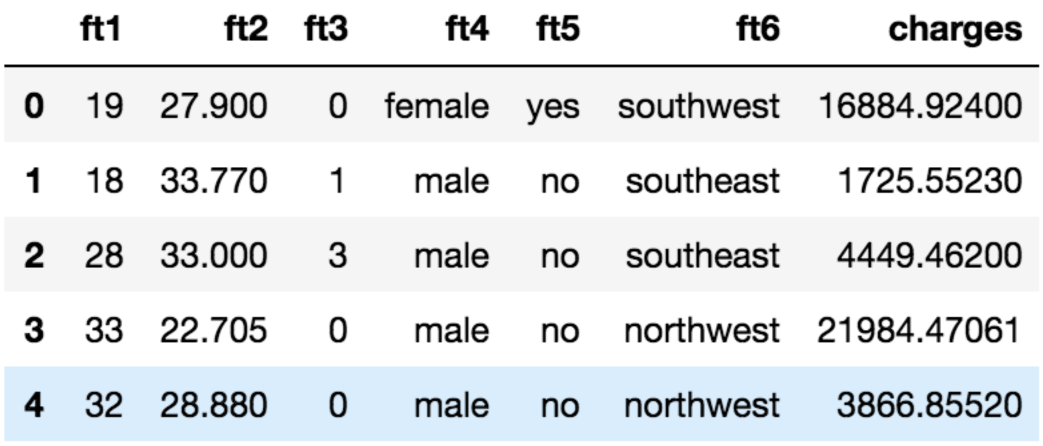
   (c) To solve this problem, you can try the following regularizations with suitable parameters.

      1. Ridge Regularizer: $\min_{\beta} \|Y - X\beta\|^2 + \alpha \|\beta\|_2^2$
      2. Lasso Regularizer: $\min_{\beta} \|Y - X\beta\|^2 + \alpha \|\beta\|_1$
      3. Elastic Net Regularizer: $\min_{\beta} \|Y - X\beta\|^2 + \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2$ (optional)

      Optimize over choices of $\alpha, \lambda_1, \lambda_2$ to pick one good model, report the best RMSE obtained via 10-fold cross validation. Compare the values of the estimated coefficients for these regularized good models, with the unregularized best model.

Dataset3

This is one car insurance dataset, you can download from this link. We try to train models to predict user's car insurance charges given the 6 features from the user's profile. As shown below, each row contains one user's 6 features and car insurance charges. **ft1,ft2 and ft3 are numerical features, ft4 and ft5 and ft6 are categorial features.**



	ft1	ft2	ft3	ft4	ft5	ft6	charges
0	19	27.900	0	female	yes	southwest	16884.92400
1	18	33.770	1	male	no	southeast	1725.55230
2	28	33.000	3	male	no	southeast	4449.46200
3	33	22.705	0	male	no	northwest	21984.47061
4	32	28.880	0	male	no	northwest	3866.85520

Figure 1: samples of car insurance data

1. **Feature Preprocessing**

(a) **Feature Encoding**: Use one-hot-encoding for the following 3 categorical features: ft4, ft5, ft6. Use the encoded features and the numerical features to fit a linear regression model.

(b) **Standardization**: Standardize (see the Useful Functions Section) all these numerical features and keep the one-hot-encoded features. Fit a linear regression model.

(c) Divide **ft1** into 3 ranges: < 30, [30,50] and > 50. That is, set the new values to 1 for original values below 30, 2 for values between 30 and 50 and 3 for values above 50. Standardize **ft2** and **ft3** feature. One-hot encoding the rest three categorical features. Fit a linear regression model.

For each model,

1) Report the average training RMSE and average test RMSE for 10 fold cross validation.
2) Plot fitted values against true values as scatter plots using the whole dataset
3) Plot residuals versus fitted values as scatter plots using the whole dataset

2. **Correlation exploration:**

   (a) Convert each categorical feature into a one dimensional numerical value. Now we have 5 numerical features. Use `f_regression` and mutual information regression measure to select two most important variables respectively. Report the two most important variables you find.

   (b) Scatter plot charges (y axis) vs ft2 (x axis), and color points based on ft5 (Yes or No).

   (c) Scatter plot charges (y axis) vs ft1 (x axis), and color points based on ft5 (Yes or No).

3. **Modify the target variable** As we have seen so far, the target variable: charges (y) spans a wide range, so here instead of fitting the original value, we consider fitting $\log(y)$. Note here instead of calculating the difference between predicted value ($\log(y)_{predict}$) and transformed target values ($\log y$), we calculate the difference between $\exp(\log(y)_{predict})$ and y to set up a fair comparison.

   (a) Pick one method of feature preprocessing from question1 to train a linear regression model on this new target. Does the performance improve?

   (b) Repeat the correlation exploration part for the new target.

4. **Bonus questions:**

   (a) Considering current results, can you further improve your results by better feature encoding? You can try polymonial features or different combinations of encoding methods.

   (b) Can you further improve your results by picking a better model?

      i. You should try at least three different types of models (e.g. random forest, neural network, gradient boosting tree)

      ii. You can also try to modify the hyper-parameters of the models you have tried. Hint: you can use `GridSearchCV` as in project 1.

Conclusion

Until now, we have experimented on three different datasets. Based on the experiences, comment on what kinds of model is more suitable for what specific type of data and what is the best feature preprocessing stage.

Useful functions

- Linear Regression Model: `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html`
- OneHotEncoder: `http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html`
- Random Forest Model: `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`
- Neural Network Models: `http://scikit-learn.org/stable/modules/neural_networks_supervised.html`
- Polynomial Transformation: `http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html`
- KNN Regressor: `http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html`
- Standardization: `http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler`

Submission: Please submit a zip file containing your **report**, and your **codes** with a **readme file** on how to run your code to CCLE. The zip file should be named as "Project4 UID1 UID2 ... UIDn.zip" where UIDx's are student ID numbers of the team members. One submission per team is required. If you have any questions you can contact the TAs or post on Piazza