Large-Scale Data Mining: Models and Algorithms **ECE 219 Winter 2018**
Prof. Vwani Roychowdhury TA: Qiujing Lu, Arash Vahabpour
UCLA, Department of ECE Tianyi Wang

Project 4: Regression Analysis

DUE Sunday, March. 4, 2018 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.

Dataset

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 a 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.
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 five features is a categorical variable: Day of the week, hour of the day, work-flow number, file-type, 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 on values $1, \cdots, 7$ corresponding to Monday through Friday. 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 Friday as $[0, 0, 0, 0, 0, 0, 1]$. We will refer to this encoding as One-Hot-Encoding.

   Now for the five variables, when looked at as a set, we have 32 $(= 2^5)$ possible combinations, where in each combination only a subset of the features are encoded using One-Hot-Encoding and the rest of the features are encoded using a scalar encoding.

   For part a-e, for each model you need to **report training and test RMSE from 10-fold cross validation** as basic evaluation of the performance. That is, for each fold you get two numbers: Training RMSE and Test RMSE. In addition, you need to: (i) Plot fitted values against true values scattered over the number of data points and (ii) Plot residuals versus fitted values scattered over the number of data points using the whole dataset for each model with the best parameters you have found. It visualizes how well your model fits 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.

      i. First 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.
      ii. **Data Preprocessing:** Standardize (see the Useful Functions Section) all these numerical features, then fit and test the model. How does the fitting result change as shown in the plots?
      iii. **Feature Selection:** Use f_regression and mutual information regression measure to select three most important variables respectively. Report the

three most important variables you find. Use those three most important variables to train a new linear model, does the performance improve?

iv. **Feature Encoding:** As explained in the preceding discussions, there are 32 possible combinations of encoding the five categorical variables. Plot the average training RMSE and test RMSE for each combination (in range 1 to 32). Which combinations achieve best performance? Can you provide an intuitive explanation?

v. **Controlling ill-conditioning and over-fiting:** You should have found obvious increases in test RMSE compared to training RMSE in some combinations, can you explain why this happens? Observe those fitted coefficients. 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)

For any choice of the hyper-parameter(s) (i.e., $\alpha, \lambda_1, \lambda_2$) you will find one of the 32 models with the lowest Test-RMSE. Optimize over choices of $\alpha, \lambda_1, lambda_2$ to pick one good model. Compare the values of the estimated coefficients for these regularized good models, with the un-regularized best model.

(b) Use a random forest regression model for this 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. For each tree, only a subset of the data set is used to build it (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 sklearn random forest regression, oob_score_ will return out of bag R^2 score, so you can calcalute 1- oob_score_ as 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

Recall that a Random Forest model can handle categorical variables without having to use one-hot or scalar encodings.

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 figure 1 for out of bag error(y axis) against number of trees(x axis), figure 2 for 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
- Activity Function(relu, logistic, tanh)

Plot Test-RMSE as a function of the number of hidden units for different activity functions. 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?

4. 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