Homework 1: Smoothers and Generalized Additive Models

Harvard CS 109B, Spring 2018

Jan 2018

Homework 1 is due Feburary 7, 2018

Problem 1: Modeling Seasonality of Airbnb Prices

In this problem, the task is to build a regression model to predict the price of an Airbnb rental for a given date. The data is provided in `calendar_train.csv` and `calendar_test.csv`, which contains availability and price data for Airbnb units in the Boston area from 2017 to 2018. Note that some of the rows in the .csv file refer to dates in the future. These refer to bookings that have been made far in advance.

Exploratory Analysis

Visualize the average price by month and day of the week (i.e. Monday, Tuesday etc.) for the training set. Point out any trends you notice and explain whether or not they make sense.

Hint: You will want to first convert the `date` column into an R Date object using `as.Date()`.

Part 1a: Explore different regression models

Fit a regression model that uses the date as a predictor and predicts the average price of an Airbnb rental on that date. For this part of the question, you can ignore all other predictors besides the date. Fit the following models on the training set and compare the R^2 of the fitted models on the test set. Include plots of the fitted models for each method.

Hint: You may want to convert the `date` column into a numerical variable by taking the difference in days between each date and the earliest date in the column, which can be done using the `difftime()` function.

1. Regression models with different basis functions:

- Simple polynomials with degrees 5, 25, and 50
- Cubic B-splines with the knots chosen by visual inspection of the data.
- Natural cubic splines with the degree of freedom chosen by cross-validation on the training set

2. Smoothing spline model with the smoothness parameter chosen by cross-validation on the training set
3. Locally-weighted regression model with the span parameter chosen by cross-validation on the training set

In each case, analyze the effect of the relevant tuning parameters on the training and test R^2, and give explanations for what you observe.

Is there a reason you would prefer one of these methods over the other?

Hints: - You may use the function `poly` to generate polynomial basis functions (use the attribute `degree` to set the degree of the polynomial), the function `bs` for B-spline basis functions (use the attribute `knots` to specify the knots), and the function `ns` for natural cubic spline basis functions (use the attribute `df` to specify the degree of freedom). You may use the `lm` function to fit a linear regression model on the generated basis functions. You may use the function `smooth.spline` to fit a smoothing spline and the attribute `spar`

to specify the smoothness parameter. You may use the function `loess` to fit a locally-weighted regression model and the attribute `span` to specify the smoothness parameter that determines the fraction of the data to be used to compute a local fit. Functions `ns` and `bs` can be found in the `splines` library.

- For smoothing splines, `R` provides an internal cross-validation feature: this can be used by leaving the `spar` attribute in `smooth.spline` unspecified; you may set the `cv` attribute to choose between leave-one-out cross-validation and generalized cross-validation. For the other models, you will have to write your own code for cross-validation. Below, we provide a sample code for k-fold cross-validation to tune the `span` parameter in `loess`:

```
# Function to compute R^2 for observed and predicted responses
rsq = function(y, predict) {
  tss = sum((y - mean(y))^2)
  rss = sum((y-predict)^2)
  r_squared = 1 - rss/tss

  return(r_squared)
}

# Function for k-fold cross-validation to tune span parameter in loess
crossval_loess = function(train, param_val, k) {
  # Input:
  #   Training data frame: 'train',
  #   Vector of span parameter values: 'param_val',
  #   Number of CV folds: 'k'
  # Output:
  #   Vector of R^2 values for the provided parameters: 'cv_rsq'

  num_param = length(param_val) # Number of parameters
  set.seed(109) # Set seed for random number generator

  # Divide training set into k folds by sampling uniformly at random
  # folds[s] has the fold index for train instance 's'
  folds = sample(1:k, nrow(train), replace = TRUE)

  cv_rsq = rep(0., num_param) # Store cross-validated R^2 for different parameter values

  # Iterate over parameter values
  for(i in 1:num_param){
    # Iterate over folds to compute R^2 for parameter
    for(j in 1:k){
      # Fit model on all folds other than 'j' with parameter value param_val[i]
      model.loess = loess(price ~ days_since, span = param_val[i],
                          data = train[folds!=j, ],
                          control = loess.control(surface="direct"))

      # Make prediction on fold 'j'
      pred = predict(model.loess, train$days_since[folds == j])

      # Compute R^2 for predicted values
      cv_rsq[i] = cv_rsq[i] + rsq(train$price[folds == j], pred)
    }

    # Average R^2 across k folds
    cv_rsq[i] = cv_rsq[i] / k
```

```
  }

  # Return cross-validated R^2 values
  return(cv_rsq)
}
```

Part 1b: Adapting to weekends

Does the pattern of Airbnb pricing differ over the days of the week? Are the patterns on weekends different from those on weekdays? If so, we might benefit from using a different regression model for weekdays and weekends. Split the training and test sets into two parts, one for weekdays and one for weekends, and fit a separate model for each training subset using locally-weighted regression. Do the models yield a higher R^2 on the corresponding test subsets compared to the (loess) model fitted previously? (You may use the loess model fitted in 1A (with the span parameter chosen by CV) to make predictions on both the weekday and weekend test sets, and compute its R^2 on each set separately, you may also use the same best_span calculated in 1A)

Part 1c: Going the Distance

You may have noticed from your scatterplots of average price versus day on the training set that there are a few days with abnormally high average prices.

Sort the training data in decreasing order of average price, extracting the 3 most expensive dates. Given what you know about Boston, how might you explain why these 3 days happen to be so expensive?

Problem 2: Predicting Airbnb Rental Price Through Listing Features

In this problem, we'll continue our exploration of Airbnb data by predicting price based on listing features. The data can be found in `listings_train.csv` and `listings_test.csv`.

First, visualize the relationship between each of the predictors and the response variable. Does it appear that some of the predictors have a nonlinear relationship with the response variable?

Part 2a: Polynomial Regression

Fit the following models on the training set and compare the R^2 score of the fitted models on the test set:

- Linear regression
- Regression with polynomial basis functions of degree 3 (i.e. basis functions x, x^2, x^3 for each predictor x) for quantitative predictors

Part 2b: Generalized Additive Model (GAM)

Do you see any advantage in fitting an additive regression model to these data compared to the above models?

1. Fit a GAM to the training set, and compare the test R^2 of the fitted model to the above models. You may use a smoothing spline basis function on each predictor, with the same smoothing parameter for each basis function, tuned using cross-validation on the training set.
2. Plot and examine the smooth of each predictor for the fitted GAM, along with plots of upper and lower standard errors on the predictions. What are some useful insights conveyed by these plots, and by the coefficients assigned to each local model?
3. Use a likelihood ratio test to compare GAM with the linear regression model fitted previously. Re-fit a GAM leaving out the predictors `availability_365` and `number_of_reviews`. Using a likelihood ratio test, comment if the new model is preferred to a GAM with all predictors.

Hint: You may use the `gam` function for fitting a GAM, and the function `s` for smoothing spline basis functions. These functions are available in the `gam` library. For k-fold cross-validation, you may adapt the sample code provided in the previous question. The `plot` function can be used to visualize the smooth of each predictor for the fitted GAM (set the attribute `se` to `TRUE` to obtain standard error curves). You may use the `anova` function to compare two models using a likelihood ratio test (with attribute `test='Chi'`).

Part 2c: Putting it All Together

Based on your analysis for problems 1 and 2, what advice would you give a frugal visitor to Boston looking to save some money on an Airbnb rental?

Part 3: Backfitting [AC209b students only]

For the model in Part 2b, rather than using the `gam` function to estimate the component smooths write an iterative function to perform the backfitting algorithm. The backfitting algorithm for fitting a generalized additive model is described on page 25 of the lecture notes.

- Rerun the model in Part 2b using your code, and using the smoothing spline smoother (*Hint*: Use the `smooth.spline` function). Do you obtain the same fitted response values using the same tuning parameter?