Homework assignment 1

A personalization case study, and optimization & regularization

E4571: Columbia University, Fall 2019
Data Science Institute
Industrial Engineering and Operations Research

Due:
7pm, October 2nd, 2019

Objective

Optimization is an essential step for learning in all machine learning algorithms in one form or another. Most widely used machine learning packages hide their fitting procedures from the user, and only occasionally expose the choice of optimization solver as an input parameter. In this exercise you will perform a simple linear regression, but you will perform the optimization yourself without the help of a regression package. You will also include regularization penalties to observe how these affect fitting.

Materials

Read (required)

- Trevor, Hastie, Tibshirani Robert, and Friedman JH. "The elements of statistical learning: data mining, inference, and prediction." (2009).
  [https://web.stanford.edu/~hastie/Papers/ESLII.pdf]
  - Chapters 1, 2, and 3-3.4.3

- Scipy's Minimize method documentation
  - https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

Watch (optional)

- Andrew Ng's lecture on regularization (7.2)
  [https://www.youtube.com/watch?v=KvtGD37Rm5I]

Assignment

[You are free to work with and consult other students in the class. However, every student should turn in their own, unique work.]

For this exercise we will try to predict the subjective quality of Portuguese red wine using only the physiochemical properties of each wine. We will not use any the common regression methods and packages for Python or R - instead we will use minimization methods in order to minimize the error of a linear model. In this case, we will define error as Residual Sum of Squares (RSS), which is also often known as Squared Error.

Though the instructions below are specific to Python, the deliverable for this assignment can be either a Python or R notebook (markdown or pdf). Submit only this notebook (all code and question answers should be in this one file). This exercise should be relatively straight forward, so don't overthink it, and keep it short and clear. Include the following steps.

Setup

- Download and load the data set for red wine into a data frame
    - http://archive.ics.uci.edu/ml/datasets/Wine+Quality
    - Use only the red wine data, not the white wine data
- Split the data by columns into features that you will use for prediction, X, and the feature you will try to predict ('quality'), y
- Split both X and y by rows into training sets a testing sets
    - Randomly split the data, keeping 80% of instances for training and 20% for testing
- At the end, you should have 4 data sets: X_train, y_train, X_test, and y_test

Regression equations and functions

- Write out two equations: (1) the equation for a the linear model that predicts y from X, and (2) the equation for computing the Residual Sum of Squares (RSS) for the linear model, given data, vector *x*, and parameters, vector β.
    - See equations 3.1 and 3.2 in the Elements of Statistical Learning book
    - Feel free to ignore the intercept term for this homework (e.g. β_0)
- Translate these equations into code in the form of two functions
    - The first function should compute the estimated value of *y*, which is *y_hat*, for particular values of *x*, and β. That is, there should be two arguments, one for the data and one for the linear function parameters.
    - The second function should compute the RSS for the first function

Optimizing the model

- Use Scipy's minimize function to find the value of β that minimize the RSS
    - https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html
    - Your call to minimize method will take three arguments:
        - (1) fun: the RSS function you defined above that you are trying to minimize
        - (2) x0: your initial values of β
        - (3) args: pass in all the data a tuple here. For example:
            - `args=(y_train, X_train)`
    - For the second argument you will need to initialize β to some starting value. Try using a random vector with Numpy random methods
        - `numpy.random.normal(0, 1, X_train.shape[1])`

    - Your final set of functions to fit your model should have the form:

```
def RSS(beta, X, y):
    return <some_results>

res = minimize(fun=RSS, x0=beta0, args=(X_train,y_train))

beta_hat = res.x
```

- Questions:
    - What are the qualitative results from your model? Which features seem to be most important? Do you think that the magnitude of the features in X may affect the results (for example, the average total sulfur dioxide across all wines is 46.47, but the average chlorides is only 0.087).
    - How well does your model fit? You should be able to measure the goodness of fit, RSS, on both the training data and the test data, *but only report the results on the test data*. In Machine Learning we almost always only care about how well the model fits on data that has not been used to fit the model, because we need to use the model in the future, not the past. Therefore, we only report performance with holdout data, or test data.
    - Does the end result or RSS change if you try different initial values of β? What happens if you change the magnitude of the initial β?
    - Does the choice of solver method change the end result or RSS?

Regularizing the model

Regularization seeks to simplify a model by decreasing the model's complexity and degrees of freedom. While lowering the degrees of freedom also decreases the flexibility of the model, and therefore the performance of the model on training data, it increases generalizability, and thus it often increases performance on test data. One common method of regularization is called shrinkage, and is defined in section 3.4 of Elements of Statistical Learning.

- Try adding in an L2 (aka Ridge) regularization penalty to your model above to create a new, regularized model. See equation 3.41 for guidance. You will need to choose a value of lambda, so start with something small, like 0.01.
- How does RSS on the training data change? How does RSS on the test data change?
- What happens if you try different values of lambda? Can you tune lambda to get the best results on the test data?
- Now try using an L1 (aka Lasso) regularization penalty instead. See equation 3.51 for example. Report your findings on how RSS changes, and if you can roughly tune lambda.
- Again, do you think that the magnitude of the features in X may affect the results with regularization?