SNLP assignment 2: regression

Deadline1: May 30, 2018

The use of regression is not as common as classification in NLP, since most predictions we want to make are discrete. Nevertheless, regression is an important method for understanding other machine learning methods, and there are occasional uses of regression in NLP applications as well. In this exercise, we will work with a plausible regression problem. The aim of the exercise is to predict the number of tweets sent from a geographic area within a particular time window. The work you do in this exercise, in principle, can be applied to, for example, event detection. If you monitor the social media in real-time, and have a model of what the 'normal' number of messages is within a given time window during the day, the observations above (or below) expected values may indicate events that one may want to pay attention to. In this set of exercises, you will built a number of regression models, and evaluate their success.

Why am I doing this?

- More Python practice
- Exercise with a linear/polynomial regression
- Exercise / think about model selection
- Observe the effects of regularization

Data

For this assignment, you are given a data set that lists only the timestamps of the tweets logged by a tweet collector very much like the one you developed in the previous assignment. You will find two files,

- `timestamps.train.gz`
- `timestamps.test.gz`

in your assignment repository. As their names indicate, the former is the training set, and the latter is the test/development set. The files contain one UNIX timestamp (seconds passed since January 1 1970) on each line.[1]

General notes

For all exercises described below, you are recommended to use regression implementations from `sklearn.linear_model`. Use `LinearRegression` in exercises 2–4, and `Ridge` in Exercise 5.

Please implement all exercises in a single Python script. This is handy as most exercises depend on the results of the earlier exercises. Mark the beginning of each exercise with a comment.

The data you get is a time-series data, there are more advanced (and arguably better) approaches for analyzing this sort of data.[2] However, for the sake of exercise, we will approach the problem as simple linear/polynomial regression.

Please read the general assignment instructions in the course syllabus.

[1] You can convert the timestamps to more meaningful time formats using Python standard `time` library, e.g., `time.localtime()`.

[2] Remember that a common assumption in linear regression is that the data points are independent and identically distributed (i.i.d.), which clearly is incorrect for our data. We will discuss a few sequence models (although not for regression) later in this course.

Exercises

Exercise 1. *Preprocessing data*

Write a python function to read in the given data file as described above, and calculate the to number of tweets arrived at each individual hour. Your function should return two `numpy` arrays (both with shape $n \times 1$) for the predictor (beginning of the hour), and the corresponding outcomes (number of tweets). Note that the predictor will repeat for each day in the data set.

Your data should look like:

hour	count
0	30
1	24
2	11
⋮	⋮
21	37
22	69
23	46
0	50
1	53
⋮	⋮

where both 'hour' and 'count' are individual numpy arrays (with the same number of rows).

Exercise 2. *Linear regression*

Fit a simple linear regression model predicting the expected number of tweets received within each hour. Output the following:[3]

- the predicted number of tweets for hours starting at 0, 8, 12, 18, 23.
- the coefficient of determination R^2 on the training data
- the coefficient of determination R^2 on the test data

Exercise 3. *Polynomial regression*

Train and test 9 (additional) polynomial regression models by increasing the exponent of the maximal term in the polynomial. That is, your first model should correspond to, $y = b_0 + b_1x + b_2x^2$, the second one should correspond to $y = b_0 + b_1x + b_2x^2 + b_3x^3$, and so on.[4] y in the model expressions is number of tweets, and x is the beginning of the hour.

Output R^2 for training and test sets for each step.

Exercise 4. *Categorical predictors*

Instead of using the 'hour' predictor as a continuous variable, convert it to a categorical predictor with one-hot encoding.[5] Train another model using the one-hot-encoded predictor, output the R^2 for both training and test sets.

Exercise 5. *Regularization*

Repeat the Exercise 4 with L2 regularization by varying the regularization strength between 0 and 30 (inclusive, with step size of 1).[6]

Epilogue

The above exercises will help you get up to speed with doing practical machine learning. Each exercise also demonstrates some of the concepts we discussed in the class. You should make sure that you understand what you are doing, as well as how to do it. In general, you are highly recommended to think about the problem beyond what the exercises ask you to do. Some interesting questions are left on the side note. Although working on or answering them is not a requirement for the assignment, thinking about these questions will help you understand the topic better, and apply it well in your future work.

More questions to think about:

- Why is plain regression not a good method for this task?
- Is there more information in the data that you can use?
- Are there other ways to improve your results?
- Would L1 regularization help?
- What does the data look like? It is always a good idea to visualize and inspect your data.
- What does my result (e.g., an R^2 of 0.45) mean? Is my model doing anything useful? Can/should it do better?

[3] For all exercises in this assignment, you can write the indicated output to the console. You do not need to write them to a file and submit it.

Although you are not asked to explain the results of the exercises in this assignment, you should always look at the results carefully, and reason about them. The point of the exercises are not printing out some numbers, but comparing them with your expectations, and trying to understand the underlying methods.

[4] Polynomial of degree 1 ($y = b_0 + b_1x$) is already covered in Exercise 2.

[5] You can use `sklearn.preprocessing.OneHotEncoder`, but you are encouraged to implement your own method for the sake of exercise.

[6] Optional: plot the training and test R^2 values against the regularization strength on the same graph.