LINEAR REGRESSION VI

11.16.2020

HOMEWORK 5

* due FRIDAY

LINEAR REGRESSION READINGS

* Chapter 5 from PDSH, particularly <u>5.6 - “In Depth: Linear Regression”</u>

* Chapter 16 from Inferential Thinking - <u>“Inference for Regression”</u>

RECAP

* np.linalg.lstsq – numpy function that does least squares regression (often bad)

* R^2 is a measure of how good a regression model is

* in-set vs. out-of-set evaluation of a regression model

REGULARIZATION

* we modify the error function to be the sum of a loss term and a penalty term

$$Err(\beta) = \boxed{\sum_{t=1}^{T} (y_t - x_t\beta)^2} + \boxed{\lambda \sum_{i=1}^{P} \beta_i^2}$$

REGULARIZATION

* it also introduces an extra parameter, λ, which is the ***regularization coefficient***, or, in this case, ***ridge coefficient***

$$Err(\beta) = \sum_{t=1}^{T}(y_t - x_t\beta)^2 + \boxed{\lambda}\sum_{i=1}^{P}\beta_i^2$$

RIDGE REGRESSION

* this type of regularization (penalizing the sum of squared weights) is called **ridge regression**

* and because the ridge error function (loss + penalty) is parabolic, it has an analytic solution!

* a nice implementation is in scikit-learn as **sklearn.linear_model.Ridge**

RIDGE REGRESSION

* but when doing ridge regression you have a new issue: how do you choose the ridge parameter, λ?

* if you train and test your regression model on the same piece of data, $\lambda=0$ is always going to be the best

  * ***~bogus~***

RIDGE REGRESSION

* if you train and test on different datasets (as discussed earlier) it’s better

  * but using your test data multiple times (to choose a parameter!) creates an issue of bias (aka **overfitting**)

RIDGE REGRESSION

* the correct solution is **cross-validation:**

  * break your dataset into training and test (X -> [X_trn, X_test])

  * further break up your training set(X_trn -> [X_fit, X_val])

  * fit weights using X_fit, choose λ based on performance on X_val, then finally test on X_test

RIDGE REGRESSION

* In reality you should understand this process (& its philosophical underpinnings), but you probably won’t need to implement it yourself

* sklearn provides functions that solve these problems already!

THE PROBLEM

* Load a dataset containing data from 442 diabetes patients
  * for each patient there are 10 features (e.g. age, sex, bmi, etc.)
  * and 1 outcome ("disease progression after one year")
* We'll be using linear regression to predict disease progression from the 10 features

LINEAR REGRESSION LAB

* If you want to following along, pull the latest version of the **ndap-fa2020** repository from github
  * https://github.com/alexhuth/ndap-fa2020/
* Then see **35-linear_regression-6/35-regression-demos.ipynb**

END