Regression Week 5: LASSO Assignment 1

In this assignment, you will use LASSO to select features, building on a pre-implemented solver for LASSO (using GraphLab Create, though you can use other solvers). You will:

* Run LASSO with different L1 penalties.
* Choose best L1 penalty using a validation set.
* Choose best L1 penalty using a validation set, with additional constraint on the size of subset.

In the second assignment, you will implement your own LASSO solver, using coordinate descent.

IMPORTANT: Choice of tools

**For the purpose of this assessment, you may choose between GraphLab Create and scikit-learn (with Pandas). You are free to experiment with other tools (e.g. R or Matlab), but they may not produce correct numbers for the quiz questions.**

* If you are using GraphLab Create, download the IPython notebook and follow the instructions contained in the notebook.
* If you are using Pandas+scikit-learn combination, follow through the instructions in this reading.

What you need to download

If you are using GraphLab Create:

* Download the King County House Sales data In SFrame format: [kc\_house\_data.gl.zip](https://eventing.coursera.org/api/redirectStrict/R7fvs6lxx8p19m0DUEYlnzhOwnBYtxk-5zb6lt4Mdc-Qbamtz_78Oti4Le5LZjU7plw4sAiRccCsMiIPGAyxHA.y3u_dYqfR-XI2xri9dT-Vg.YfgFKspYekImtlUhYWbSZekZvJGjf0SYA84kv1_oKZZmcG8J3N4un0L43tWA4vJzEGiHoezSmrbZpPaeR8iKaPdLGly3iCPtUfIhTh1Xr1COReBewp6cN24ipouIOKnGG47JrWla4RKgZJzMk1Pz1E93mtZA3aCZqdTmP7SJ_wP9L73GUL9mAm8xIPVJ-moD1OvlClEpdRJ5TSv7VOTUaOY12UlBU6nCcJufJvHUW_LOPYUgACfepKnDizVBW4c-pFUBNdukv9Ube-j7NxW1uZLM9XL8sujvCIwYV9WFo1jmRkT7LdGIntlphGd2tHbwjUvwUfosWw4632SJZv4cdQhlrs_V2-DwKmo_gj8bleIFJ8ytBv_GMuSP3V7NZa54_Bp5KHmiA9skY8ZcxO9wA2-dbjOds6axRGIoJHPuZOXlTCka9mrR5gK07RMfjvag)
* Download the companion IPython Notebook: [week-5-lasso-assignment-1-blank.ipynb](https://eventing.coursera.org/api/redirectStrict/K0qGnGVnBVvS0ZUohNqsgkanF6Vr0OP3LCOi3hUJpBY8qifUaReFTZxLHh8SYvwwmMl501p5ce4Is0IYNpTlWw.WLO-51hflw0NmZ1wU7sXBw.XHbQSiXAsG9TtByKDPhzQR8rD84g0sXN9I1m2sTIbLdMtl4lIE8hiL3Xb_OKHjIpZ5PlMpep7sv3PL83GmqQM5GezFVIhK_HpmQQOzMfi9HpbmXxOzEqvaSxMiU-Ywu8zmXEWQEbW-C5o9XCOjOD0dedy-L0uP0R6s7pcPBUvJcnMA2lG4jrfH_cmF84dxzNkUEjImhd61_bFTlsDgpnVjMGtFlAd8b9gOjz9Wrk3lPeES5dhBoor2cgH1QKpSzC3JoGDo9ukah4vgKHTm8gYN-kpHKtyoQqHeL3swruAjGecuSa9dbVrQC2rydmzm6SybC4RFZQlN3dKp4Ao47hqcKkeOBl8pDdI_g-FlNsm4mGWo4c6T6N_S-Xr1r7xyPzFfAOzlihIN_Z0PNXLFc_RfYnIEDBRpqnTxTEb_LwlLzIBWqd9vGMeXdVmLAdrI9MhC8STzXZ0Pnt16ENQM4JU6Rj-2Vj22JXdygTbSUGz_0)
* Save both of these files in the same directory (where you are calling IPython notebook from) and unzip the data file.

If you are using scikit-learn with Pandas:

* Download the King County House Sales data csv file: [kc\_house\_data.csv](https://eventing.coursera.org/api/redirectStrict/lgy1-4CeHfO1QIghk8XJeY-8Zbh2hp44_mvoIc5hx57-hMlTw0zSA-VfiYlSS1NRZUgLrgKsDtXVXDhKsp0pRw.-P35U6B-aYN8oRfhcvIG7Q._-zha7LZ_BlrSAeJMpODeeuEwyWmzAZyPI8S-8M8axfJoq69mXFlIbi6z5b4297JjqGpVDSqrtQBuf2uxTQc3eEC1V4F4img1SqxabpbyODOYHpLp4Ep4X-qoFewrDIL8QMR5sc1TqrB0XykSYQ_ddkIJRAcdq4Tp0U5CzCOBRH-aj5tfLhlIvz3rkoYpCDJG6UdGBhFecm7mif5xxjGcOa6V9F7sf_xbiqnJd6ZCdocbaWC1DAtYrbHUfB_4Q6FfWLiIxFiyAAqSGA4inRFD3itGOJjvfnBU_UwThOVQk0TXsoj-hxVEkxOkGp2YARW8VFfpNnv2Cc2uOuve8GR53_l8KboXFgaBJ97eEJhS5u8S82Aq1kmX4v24Bcf0toZldE6wWDGSEhAHozttg3SH6LRD7oeytr7OU3r1parICsBLR_R25dLc34g2KITmegm)
* Download the King County House Sales training data csv file: [wk3\_kc\_house\_train\_data.csv](https://eventing.coursera.org/api/redirectStrict/qazFaWLAhuA-bLBe1RSHlSVTjycDFHqJSj3QDahT5dFCsFNRYstvH87weC8aYStXM2V9-_KEJKLEwYco-sq3nw.WxXbd0Cx0mv0Dwzn0tpwpg.dxkajQZPR6k_7eEt05utieGNq10w4zu5SarTsgqXWp-1vDNJrWZg5VefSenk9OcRyj9ZOYkqDcsET4Zdz4z6ML5IT9Xdytfb_aZ2Y5QMdqqFU0rQYxFDx6LD02qoHAvOAJDmMhyIcQdlo5RHEq-UPLgQYA7qrCqkETArpkiAj82mfCrGmLHmitHs3a_0nQVOIbd2_3OG_Q6SIs6Kvf3c7iEAuLKkGqHVclOTmxB5vRfgnRmbdMCgU15HsZAgPwy5mshESIrIXWQz5yoZVtD5pEKWpPMCncU58BSZkV3fsz4wSift4e09D1gTRUKWI9EohiZuPSsF3923JpSD1BiRRZq093HjJMG98HfvZ3fW9Y9UZWMRjg9t731nwEjold96LRkdFua3puvI8_qpoxq8PDulXPF3ljELGJmpmRk_DcSSjcHPBdmf5t_25Diate7RYRmJy4pKoEwAzPZrCu07iA)
* Download the King County House Sales validation data csv file: [wk3\_kc\_house\_valid\_data.csv](https://eventing.coursera.org/api/redirectStrict/BEM8ndOqCithsa_iG6XSe84ufLtDytkCCVFEZ8yd8eN30Z8inZmEn3VDG9Oa6P6BRaRorgd7LFXMWR-58nilgg.Wq1c_3ESC4TSZ6gGrOYvhw.ZQ0q6hO0N1vl2UarQKjH6mVMM4y-Ge3afAMQk-w1wGk6IozEdeZH8ESvqQGL9slCW2yPBeEFuPC3ZsHKFpq0dqHhk3LKmAxmVDzDZI-fo_Qa8Nb3eMyPRdqYJzGoJOW28_Er6pFKB1fnkbcX-PDq0RPBCC2mqx5IZ7yrAV3LSEb1B0f09pwXCqZUi_wtFZGjXcW3uXpqhReriR7lkapo0Xd9XU-8IqbKPpXm_JhdxOptAfOqZLoipUKBsh0EuzriYQsNFcdOKC4thhx3HLZ3nA_fDE21jhX-_Mu9IGxV3fNrsgw12LgXIxkaIW0drNtf7yA_C0l3nw65zNvZ2PQBb48BNiXm7G49WcrdNsc4pN7J7WXTgDSK6_8sntdowh1SuGP9Hrb5LOJiYQ5eBBUOk8abiRKnQhr5fHtIA9OxJ7rj44S9FPIBrIxT2CrYd_BuGAbWpzSHyGbkz6EC_4EMMg)
* Download the King County House Sales testing data csv file: [wk3\_kc\_house\_test\_data.csv](https://eventing.coursera.org/api/redirectStrict/I0IjBIZtZbOaZdoJKvQkJu2iDcUnC9HmOhtJQUNQaY5ChZaJB3tkKETQVJJ5vg0aKE4oPrGDV2zD4eJwFCGP7A.JpudCpO8IZohRTcVJ0Eg8g.ytCt0uJc31tR5u25dNDvWskvrB-QfnEYi0EF0hLi9s0kIO93dlHGSU4otHMGebo6L4B85j4nNQD1tHgUmsTPP0baEt51o2rzkSXKESFCP0Q67UiHwBwAHx612xPE6xRJZfhuSZJHe4pCxCCJe-U1VyHJdvjyCoKNpmaXAmS3-McwbZDVPUDWVtGlV8X9S5v45vC6AZPZPnix2RKQXqoIMG7XDJCuMq3ZjhML10qrHo4MDhlka_-wiGxWAY2bKpZ9jYnLAHHOv-6TcaQkji2gWLoXM7m2hWA9UPbUJuYBScVgse2qFbc04eYk-t90rOwKD-CSfcTQYgO9X6w9k8LmgQczYX5IR4XAuz3utWrGolq_OZb7wlhoZhdS3USE-QfLddCyhEF7OfLLgAXasVM_VcWVlW2tOjTv3jnvrEcNHgBZgh4gQcFRTzNPiptQ55ec6aD6SBoGCCNRYlI46KuSPg)

Useful resources

You may need to install the software tools or use the free Amazon EC2 machine. Instructions for both options are provided in the reading for Module 1 (Simple Regression).

If you are following the IPython Notebook and/or are new to numpy then you might find the following tutorial helpful: numpy-tutorial.ipynb

If you are using GraphLab Create and the companion IPython Notebook

Open the companion IPython notebook and follow the instructions in the notebook.

If you are using scikit-learn with Pandas:

The instructions may apply to other tools, but the set of parameters are specific to scikit-learn.

**0**. Load the sales dataset using Pandas:

import pandas as pd

dtype\_dict = {'bathrooms':float, 'waterfront':int, 'sqft\_above':int, 'sqft\_living15':float, 'grade':int, 'yr\_renovated':int, 'price':float, 'bedrooms':float, 'zipcode':str, 'long':float, 'sqft\_lot15':float, 'sqft\_living':float, 'floors':float, 'condition':int, 'lat':float, 'date':str, 'sqft\_basement':int, 'yr\_built':int, 'id':str, 'sqft\_lot':int, 'view':int}

sales = pd.read\_csv('kc\_house\_data.csv', dtype=dtype\_dict)

**1.**Create new features by performing following transformation on inputs:

from math import log, sqrt

sales['sqft\_living\_sqrt'] = sales['sqft\_living'].apply(sqrt)

sales['sqft\_lot\_sqrt'] = sales['sqft\_lot'].apply(sqrt)

sales['bedrooms\_square'] = sales['bedrooms']\*sales['bedrooms']

sales['floors\_square'] = sales['floors']\*sales['floors']

* Squaring bedrooms will increase the separation between not many bedrooms (e.g. 1) and lots of bedrooms (e.g. 4) since 1^2 = 1 but 4^2 = 16. Consequently this variable will mostly affect houses with many bedrooms.
* On the other hand, taking square root of sqft\_living will decrease the separation between big house and small house. The owner may not be exactly twice as happy for getting a house that is twice as big.

**2.**Using the entire house dataset, learn regression weights using an L1 penalty of 5e2. Make sure to add "normalize=True" when creating the Lasso object. Refer to the following code snippet:

from sklearn import linear\_model # using scikit-learn

model\_all = linear\_model.Lasso(alpha=5e2, normalize=True) # set parameters

model\_all.fit(sales[all\_features], sales['price']) # learn weights

**3. Quiz Question: Which features have been chosen by LASSO, i.e. which features were assigned nonzero weights?**

**4.**To find a good L1 penalty, we will explore multiple values using a validation set. Let us do three way split into train, validation, and test sets. Download the provided csv files containing training, validation and test sets.

testing = pd.read\_csv('wk3\_kc\_house\_test\_data.csv', dtype=dtype\_dict)

training = pd.read\_csv('wk3\_kc\_house\_train\_data.csv', dtype=dtype\_dict)

validation = pd.read\_csv('wk3\_kc\_house\_valid\_data.csv', dtype=dtype\_dict)

Make sure to create the 4 features as we did in #1:

testing['sqft\_living\_sqrt'] = testing['sqft\_living'].apply(sqrt)

testing['sqft\_lot\_sqrt'] = testing['sqft\_lot'].apply(sqrt)

testing['bedrooms\_square'] = testing['bedrooms']\*testing['bedrooms']

testing['floors\_square'] = testing['floors']\*testing['floors']

training['sqft\_living\_sqrt'] = training['sqft\_living'].apply(sqrt)

training['sqft\_lot\_sqrt'] = training['sqft\_lot'].apply(sqrt)

training['bedrooms\_square'] = training['bedrooms']\*training['bedrooms']

training['floors\_square'] = training['floors']\*training['floors']

validation['sqft\_living\_sqrt'] = validation['sqft\_living'].apply(sqrt)

validation['sqft\_lot\_sqrt'] = validation['sqft\_lot'].apply(sqrt)

validation['bedrooms\_square'] = validation['bedrooms']\*validation['bedrooms']

validation['floors\_square'] = validation['floors']\*validation['floors']

**5.**Now for each l1\_penalty in [10^1, 10^1.5, 10^2, 10^2.5, ..., 10^7] (to get this in Python, type np.logspace(1, 7, num=13).)

* Learn a model on TRAINING data using the specified l1\_penalty. Make sure to specify normalize=True in the constructor:

model = linear\_model.Lasso(alpha=l1\_penalty, normalize=True)

* Compute the RSS on VALIDATION for the current model (print or save the RSS)

Report which L1 penalty produced the lower RSS on VALIDATION.

**6. Quiz Question: Which was the best value for the l1\_penalty, i.e. which value of l1\_penalty produced the lowest RSS on VALIDATION data?**

7. Now that you have selected an L1 penalty, compute the RSS on TEST data for the model with the best L1 penalty.

**8. Quiz Question: Using the best L1 penalty, how many nonzero weights do you have? Count the number of nonzero coefficients first, and add 1 if the intercept is also nonzero.**A succinct way to do this is

np.count\_nonzero(model.coef\_) + np.count\_nonzero(model.intercept\_)

where 'model' is an instance of linear\_model.Lasso.

**9.**What if we absolutely wanted to limit ourselves to, say, 7 features? This may be important if we want to derive "a rule of thumb" --- an interpretable model that has only a few features in them.

You are going to implement a simple, two phase procedure to achieve this goal:

* Explore a large range of ‘l1\_penalty’ values to find a narrow region of ‘l1\_penalty’ values where models are likely to have the desired number of non-zero weights.
* Further explore the narrow region you found to find a good value for ‘l1\_penalty’ that achieves the desired sparsity. Here, we will again use a validation set to choose the best value for ‘l1\_penalty’.

**10.**Assign 7 to the variable ‘max\_nonzeros’.

**11.**Exploring large range of l1\_penalty

For l1\_penalty in np.logspace(1, 4, num=20):

* Fit a regression model with a given l1\_penalty on TRAIN data. Add "alpha=l1\_penalty" and "normalize=True" to the parameter list.

model = linear\_model.Lasso(alpha=l1\_penalty, normalize=True)

* Extract the weights of the model and count the number of nonzeros. Take account of the intercept as we did in #8, adding 1 whenever the intercept is nonzero. Save the number of nonzeros to a list.

**12.**Out of this large range, we want to find the two ends of our desired narrow range of l1\_penalty. At one end, we will have l1\_penalty values that have too few non-zeros, and at the other end, we will have an l1\_penalty that has too many non-zeros.

More formally, find:

* The largest l1\_penalty that has more non-zeros than ‘max\_nonzeros’ (if we pick a penalty smaller than this value, we will definitely have too many non-zero weights)Store this value in the variable ‘l1\_penalty\_min’ (we will use it later)
* The smallest l1\_penalty that has fewer non-zeros than ‘max\_nonzeros’ (if we pick a penalty larger than this value, we will definitely have too few non-zero weights)Store this value in the variable ‘l1\_penalty\_max’ (we will use it later)

Hint: there are many ways to do this, e.g.:

* Programmatically within the loop above
* Creating a list with the number of non-zeros for each value of l1\_penalty and inspecting it to find the appropriate boundaries.

**13. Quiz Question: What values did you find for l1\_penalty\_min and l1\_penalty\_max?**

**14.**Exploring narrower range of l1\_penalty

We now explore the region of l1\_penalty we found: between ‘l1\_penalty\_min’ and ‘l1\_penalty\_max’. We look for the L1 penalty in this range that produces exactly the right number of nonzeros and also minimizes RSS on the VALIDATION set.

For l1\_penalty in np.linspace(l1\_penalty\_min,l1\_penalty\_max,20):

* Fit a regression model with a given l1\_penalty on TRAIN data. As before, use "alpha=l1\_penalty" and "normalize=True".
* Measure the RSS of the learned model on the VALIDATION set

Find the model that the lowest RSS on the VALIDATION set and has sparsity equal to ‘max\_nonzeros’. (Again, take account of the intercept when counting the number of nonzeros.)

**15. Quiz Question: What value of l1\_penalty in our narrow range has the lowest RSS on the VALIDATION set and has sparsity equal to ‘max\_nonzeros’?**

**16. Quiz Question: What features in this model have non-zero coefficients?**