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://d18ky98rnyall9.cloudfront.net/_026a0fd773a5fdd104e1a6ca3cfb2622_kc_house_data.gl.zip?Expires=1547337600&Signature=IOPVv0Rpe7fELYYGpfz1wfSwO-RGKOjaPuuBi5YWUnXaL0BvJnRNafmFo7lcF9JqCAVU8C-r989ooyorrXcF8v2yn4c7wHMi0dlgwchxBr8rD7keAGe9AfpnDLv5UR5-3elCRARJYmZiYAA2oJkmQyYlMuenKJFNVSiDAbGCyos_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

* Download the companion IPython Notebook:

[week-5-lasso-assignment-1-blank.ipynb.zip](https://d18ky98rnyall9.cloudfront.net/_87a6d2d011793ce32339c0935afe6b7c_week-5-lasso-assignment-1-blank.ipynb.zip?Expires=1547337600&Signature=hOrgG6jYCWu9yrSIxNKp-otn8BQGfJp0pbzp66nVqcWxCT9xLQ3RQTvyrYHpEu1aXmArx8FJ83uoHyWfdd6oCkqBixQiZK6QtJ6CzrkYW8ArHcyMxw3tN0l1bcrT7kAZgQ18gSr5j1mMnQF~V4a~9pU9tkk9V7RP~tVWrvwBH3k_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

* 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.zip](https://d18ky98rnyall9.cloudfront.net/_46994807796a1213d2699c6d9a09667c_kc_house_data.csv.zip?Expires=1547337600&Signature=J-gUsZYVwwPwf3jywBjwgfQgWyg9ySAVisMVPe0Txh80gRZiNZStapUnf7DSPGEt-75KwndcqCBmO7gc3~oPWvK4r355ZDX2lVtunnRy-w5quMZy47xCdxLplIKZX~25DNTdbIwne14TODKKvm8mMdT~S3p7QKXOUY-3BCfu1gw_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

* Download the King County House Sales training data csv file:

[wk3\_kc\_house\_train\_data.csv.zip](https://d18ky98rnyall9.cloudfront.net/_f626f6faf3c1039d014563b39ede3037_wk3_kc_house_train_data.csv.zip?Expires=1547337600&Signature=KF7BxyauxZiuB~jthJtDc4oV9v6HB0h0OToqahs~wFLv3lNS0utxT8HDEL4DaQQaLB2D45ponpvCuppWvz204LEsiT84-SqxrDFFe6D7VY3PTQMJCgqfcD5QSIDMptpKtuPdMwEvMHSb2qRw0OZpxg6juKUx8F8ntNzyq83upZU_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

* Download the King County House Sales validation data csv file:

[wk3\_kc\_house\_valid\_data.csv.zip](https://d18ky98rnyall9.cloudfront.net/_f626f6faf3c1039d014563b39ede3037_wk3_kc_house_valid_data.csv.zip?Expires=1547337600&Signature=bxNJwtUotzArqzdHFzdKEwXf6h-DxTniPMEz3ohrBMbCojkfLZP30L-VO2NRfFtcVjmGn06VGeFGuSgeZSJB1bRB28L2TDhScZ1QbUddRFGs-1xF8pk7PbW~Q-cMqh2xwXTPal5zPxOSShLyklbKg406I-XJdnaG~D3q3yvxoOM_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

* Download the King County House Sales testing data csv file:

[wk3\_kc\_house\_test\_data.csv.zip](https://d18ky98rnyall9.cloudfront.net/_f626f6faf3c1039d014563b39ede3037_wk3_kc_house_test_data.csv.zip?Expires=1547337600&Signature=YgsuP-iAIW64jgfKReWZDneq7QxgGPypInYtLQlDfi4gZkEFsTA-5Vo8glWct9LDx7TSQQI9Btse0RdH3NYIDgmgroPxh6O4vZpkKt89pyVMVoWFC1W9f9HnD9Nk~-G7Y-pArFEHB2b1oVDksrMhdQ367vojvBfRKJT1nqLMNLo_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

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.zip](https://d18ky98rnyall9.cloudfront.net/_f6e2fce565c2e2d9e75019560739b126_numpy-tutorial.ipynb.zip?Expires=1547337600&Signature=LAC10mx6w1QaCucH-Y-aRwA~eX9QgCFxeWPKIKDxTg6ViHcpNUS~B~ZNIOHAjZfIUSxMWNs~ElZFJljywQgW~CWz~BgCWRoMRK83VSJbxEQ4-OlmHxXFukrHZgLD-YSFsCWQ0EBgLqNZQ5sIuDDYQpl5RT0vJc1pwUl154m3-vo_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A)

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:

1

2

3

4

5

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:

1

2

3

4

5

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 for the list of features.

**Note.** From here on, the list 'all\_features' refers to the list defined in this snippet.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

from sklearn import linear\_model # using scikit-learn

all\_features = ['bedrooms', 'bedrooms\_square',

'bathrooms',

'sqft\_living', 'sqft\_living\_sqrt',

'sqft\_lot', 'sqft\_lot\_sqrt',

'floors', 'floors\_square',

'waterfront', 'view', 'condition', 'grade',

'sqft\_above',

'sqft\_basement',

'yr\_built', 'yr\_renovated']

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.

1

2

3

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:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

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:

1

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

1

2

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.

1

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?**