Assignment 2: More Regressions

UVA CS 6316 :
Machine Learning (Fall 2018)

Out: W4
Due: 1001 Tue midnight 11:59pm @ Collab

a *The assignment should be submitted in the PDF format through Collob. If you prefer hand-writing QA parts of answers, please convert them (e.g., by scanning or using an app like Genuis Scan) into PDF form.*

b *For questions and clarifications, please post on Piazza.*

c *Policy on collaboration:*

Homework should be done individually: each student must hand in their own answers. It is acceptable, however, for students to collaborate in figuring out answers and helping each other solve the problems. We will be assuming that, with the honor code, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.

d *Policy on late homework:*

Homework is worth full credit at the midnight on the due date. Each student has three extension days to be used at his or her own discretion throughout the entire course. Your grade would be discounted by 15% per day when you use these 3 late days. You could use the 3 days in whatever combination you like. For example, all 3 days on 1 assignment (for a maximum grade of 55%) or 1 each day over 3 assignments (for a maximum grade of 85% on each). After you've used all 3 days, you cannot get credit for anything turned in late.

e *Policy on grading:*

***1**: 45 points in total. 20 points for code submission (and able to run). 10 points for each question answered in your report.*
***2**: 45 points in total. 5 points for answering the primer question. 20 points for code submission (and able to run). 10 point for displaying figures. 10 points for answering questions in your report.*
***3**: 10 points in total. 5 points for each question.*
***4**: 10 points of extra credit. See question 2.*
The overall grade will be divided by 10 and inserted into the grade book. Therefore, you can earn 11 out of 10.

Please provide proper steps to show how you get the answers.

1 Polynomial Regression (Programming)

There are many datasets where a standard linear regression model is not sufficient to fit the data (i.e. the data is not linear). Thus, we need a higher order model to better fit the data. For this function, recall from the class lectures that the problem of learning the parameters is still linear although we are learning a nonlinear function. There are **TWO** portions to this section.

First, you must generate 300 samples using "DataGenerationPoly.py". Then you must fill out the "PolyRegressionTemplate.py" and turn it in as "PolyRegression.py".

Second, you must complete a written portion and turn it in as part of the pdf with the rest of the assignment. You must explain (and include) the three plots(see details below) and whether the outputted theta makes sense given the true underlying distribution.

1.1 Data Generation

- First, we will generate our data. In machine learning, we assume that there in an underlying data distribution. There should be some hidden function that maps the inputs to the outputs we are trying to predict plus some noise. The noise comes from not having all the relevant information and some of the inputs being wrong. We try to figure out what the hidden function is so that we can make good guesses of the output given the input.
- We have provided "DataGenerationPoly.py" that has the underlying data distribution that we are trying to find. We get samples by running the script with a command line argument specifying the number of samples we want. The data is saved as "dataPoly.txt". We will use this data for the next steps.

1.2 Polynomial Regression Model Fitting

- ATT: Please re-use the gradient-descent regression solver you implemented in HW1 in the function: $solve\_regression(x, y)$. If you did not have a good implementation, please contact instructors-19f-cs-6316@collab.its.virginia.edu.
- **Task 1 hyperparameter tuning**: We will plot polynomial order versus training and *validation* loss. We will use all 300 samples for this task. For this plot, use 60% of the total data as training data and 20% as validation data. Leave the last 20% for testing later. As discussed in class, validation loss is used to tune hyperparameters because training loss is minimized by the highest variance model. Additionally, if you use test loss to tune hyperparameters then test loss is no longer a good estimation of the true error of your model which is useful to know after producing said model.

  Refer to $get\_loss\_per\_poly\_order(x, y, degrees)$ in the template.

  This function should explore multiple different orders of fitting polynomial regressions, like $d \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ as discussed in class. In this plot, x-axis shows the value of d and the y-axis represents the MSE loss on the training set and MSE loss on the validation set(as two different curves in the same plot).
- **Task 2**: In part 2, we will use the best hyperparameter d from part 2 to train our final model. We use both the validation and training data to train our model. This is because now that we have tuned our hyperparameter we no longer need the validation set but the extra training data is useful. Then we use the test error as a good estimation of the true error. Similar to HW1, you will plot the best fit curve: showing the data samples and also draw the best-fit curve learned on the same graph. Please include the best fit curve, final test MSE error and the best θ as part of your write up. Also discuss how you got the best θ. Now, please take a look at the data generation script and note your observations as part of the write up if the learnt θ makes sense in relation to the true underlying distribution.

  Att: When you try to draw the best-fit curve, please do not directly use the x from the training set or validation set. Instead, you can (1) just sort the x and y values by x when plotting the curve; OR (2) you can use the function $linspace(start, end, NumSamples)$ **from numpy to get a set of x (uniformly distributed) for plotting the best-fit-curve nicely.**

- **Task 3**: Similar to the epoch vs. training losses we require you to generate in HW1, now please generate a figure including two curves: 1. the GD epoch vs. training loss, and 2. the GD epoch vs. loss on the testing data. This is a great figure to visually check how the GD optimize to reduce the training loss and the gap between the training loss and test loss along the gradient descent optimization path.

- **Task 4**: Last, you are required to plot training and testing loss versus dataset size. In real life collecting more data can be expensive. This plot can help tell if collecting more data would even help the model increase accuracy. Additionally, this graph indicates whether your model has overfit/underfit.

  Refer to $get\_loss\_per\_num\_examples(x, y, example\_num, train\_proportion)$ in the coding template.

  Polynomial regression using a degree of 8 will be used for this question. The code should generate a figure with x-axis representing the value of n, the number of examples used for the model and the y-axis showing the training MSE loss and the test MSE loss(as two different curves). For this plot, you will vary n using $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. $example\_num$ is a list of n. $train\_proportion$ is the proportion of n to be used for training, and the rest for testing. For this question, you will use $train\_proportion = 0.5$. In a 'real' setting, we would use more of the data for training. Please include the plot in the written submission.

We should be able to run "python3 PolyRegression.py" and it should work!

2 Ridge Regression (programming and QA)

There are **THREE** portions to this section.

First, you have questions to answer that act as a primer for the coding question. Include the answers in the written portion.

Second, you must generate 200 samples using "DataGenerationRidge.py". Then you must fill out the "RidgeRegressionTemplate.py" and turn it in as "RidgeRegression.py".

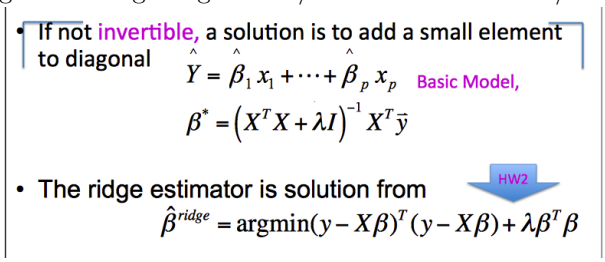
Third, you must explain the plot, the L2 norms, and the testing loss i.e. why are some parts relatively high and other parts low.

Extra Credit: code up gradient descent for ridge regression and show that it matches modified normal equation for the first 5 values of beta.

2.1 QA

- Here we assume $X_{n\times p}$ represents a data sample matrix which has p features and n samples. $Y_{n\times 1}$ includes target variable's value of n samples. We use β to represent the coefficient. (Just a different notation. We had used θ for representing coefficient before.)
- 1.1 Please provide the math derivation procedure for ridge regression (shown in Figure 1)

Figure 1: Ridge Regression / Solution Derivation / 1.1



(Hint1: provide a procedure similar to how linear regression gets the normal equation through minimizing its loss function.)
(Hint2: $\lambda|\beta|_2 = \lambda\beta^T\beta = \lambda\beta^T I\beta = \beta^T(\lambda I)\beta$)
(Hint3: Linear Algebra Handout Page 24, first two equations after the line "To recap,")

- 1.2 Suppose $X = \begin{bmatrix} 1 & 2 \\ 3 & 6 \\ 5 & 10 \end{bmatrix}$ and $Y = [1, 2, 3]^T$, could this problem be solved through linear regression?

Please provide your reasons.
(Hint: just use the normal equation to explain the reason)

- 1.3 If you have the prior knowledge that the coefficient β should be **sparse**, which regularized linear regression method should you choose? (Hint: sparse vector)

2.2 Programming

- Similar to the previous section, we will first generate data using the provided script "DataGenerationRidge.py". We generate 200 samples in this part. The data is saved as "dataRidge.txt". We will use this data for the next steps.
- **Task 1**: We will use cross-validation for this question. You are required to plot training loss and validation loss (as two curves in the same plot) as a function of hyperparameter λ. In this plot, the x-axis is λ and y-axis is the training loss and validation loss. You will use 4-fold cross validation for this question. Refer to the function $cross\_validation(x\_train, y\_train, lambdas)$ in the template. Here, the validation loss is the average validation loss across the 4 folds during cross-validation. You are also required to reimplement *normal_equation* for ridge regression. The values of λ are specified in the template. Please include some discussion about your observations from the plot i.e. discuss the trends based on the increasing or decreasing λ values.
- **Task 2**: Please write down the best λ in the previous step in the written submission. Finally, your code should print out L2 norms and the test loss of the learnt β_λ parameters for the best λ as well as other values of λ (specified in the template). Please include these values as part of the written submission. Also, include your observations about the norms and the test loss, discussing general trends of the values.
- **Task 3:** In this part, you are required to submit a bar graph showing the learnt values of β vector from the best λ. If β is a $p \times 1$ vector, in this plot, the x-axis is i where $i \in \{1, \ldots, p\}$ and the y-axis denotes the β_i. Now take a look at the data generation file and discuss your observations if the learnt β makes sense in relation to the true underlying distribution.

Explanation of Ridge Regression:

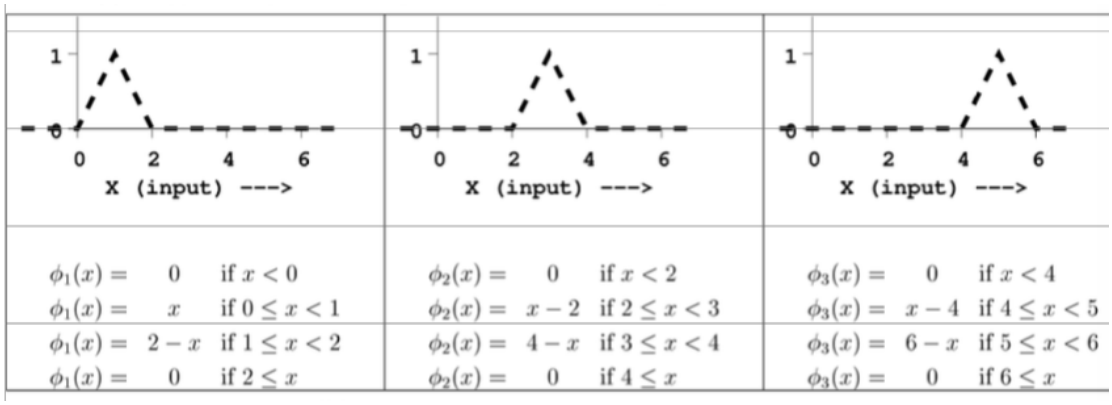
- As you can see from the data generation file, most of the features are useless. The output depends on the bias (via the pseudofeature) and x1. The rest of the xs are noise that has no influence on y.
- However, straight linear regression will use those values to predict the output exactly. Aka the model learns the noise.
- Ridge regression penalized the l2 vector norm of beta. The model's predictions get worse less quickly when it lowers weights associated with unimportant features than with important features. As a result, the model learns less noise.

Explanation of k-fold cross validation:

- If you do not have a lot of data, then using a training set, validation set, and testing set doesn't work very well. For example, if your validation set is too small you won't be able to tune hyperparameters well because the validation loss will be too noisy.
- One solution is k-fold cross validation. It is more complicated (so takes longer to code) and runs slower than using training, validation, and testing sets. However, It allows you to better hyperparameter tune.
- We take the training set and split it into "k" folds. Then we combine all the folds execept one and use that as the training set with the leftout fold as validation set. We do this k times using each left out fold as a validation set. Then we average the training losses and validation losses and say that is our training and validation loss.
- In this way all the training data gets some time being part of the validation set, so the validation loss is less noisy and we can pick better hyperparameters.

3 Sample Questions:

Question 1. Basis functions for regression



$\phi_1(x) =$	0	if $x < 0$
$\phi_1(x) =$	x	if $0 \leq x < 1$
$\phi_1(x) =$	$2 - x$	if $1 \leq x < 2$
$\phi_1(x) =$	0	if $2 \leq x$

$\phi_2(x) =$	0	if $x < 2$
$\phi_2(x) =$	$x - 2$	if $2 \leq x < 3$
$\phi_2(x) =$	$4 - x$	if $3 \leq x < 4$
$\phi_2(x) =$	0	if $4 \leq x$

$\phi_3(x) =$	0	if $x < 4$
$\phi_3(x) =$	$x - 4$	if $4 \leq x < 5$
$\phi_3(x) =$	$6 - x$	if $5 \leq x < 6$
$\phi_3(x) =$	0	if $6 \leq x$

Figure 2: Basis functions for regression (c) with one real-valued input (x as horizontal axis) and one real-valued output (y as vertical axis).

We plan to run regression with the basis functions shown as above, i.e., $y = \beta_1\phi_1(x) + \beta_2\phi_2(x) + \beta_3\phi_3(x)$. Assume all of our data points and future points are within $1 \leq x \leq 5$. Is this a generally useful set of basis functions to use ? If "yes", explain their prime advantage. If "no", explain their biggest drawback. (1 to 2 sentences of explanation are expected.)

Question 2. Polynomial Regression

Suppose you are given a labeled dataset (with one real-valued input and one real-valued output) including points as shown in Figure 3 :

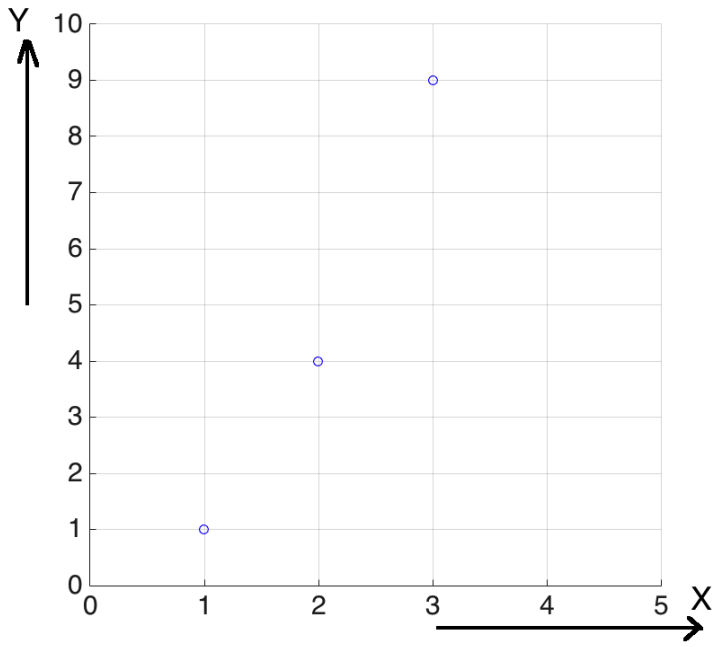


Figure 3: A reference dataset for regression with one real-valued input (x as horizontal axis) and one real-valued output (y as vertical axis).

(a) Assuming there is no bias term in our regression model and we fit a quadratic polynomial regression (i.e. the model is $y = \beta_1 x + \beta_2 x^2$) on the data, what is the mean squared LOOCV (leave one out cross validation) error?