CS589: Machine Learning - Spring 2020

Homework 3: Kernels

Assigned: March 27^{th}, 2020 Due: 5:00 pm, Apr 10^{th}, 2020

In this homework you will experiment with kernel methods for regression and classification problems using kernel ridge regression and support vector machine (SVM).

Getting Started: In this assignment, you will train and evaluate different kernels for both classification and regression on three datasets. Please install Python 3.7 via Anaconda on your personal machine. For this homework you will only be using numpy, scipy, sklearn and matplotlib packages. Download the homework file HW03.zip (available on Moodle). Unzipping this folder will create the directory structure shown below,

```
HW03
--- HW03.pdf
--- Data
    |--Synthetic
    |--Housing
    |--Corona
--- Submission
    |--Code
    |--Predictions
       |--Housing
       |--Corona
```

The data files for each data set are in 'Data' directory respectively. You will write your code under the Submission/Code directory. Make sure to put the deliverables (explained below) into the respective directories.

Deliverables: This assignment has three types of deliverables: a report, code files, and Kaggle submissions.

- **Report:** The solution report will give your answers to the homework questions (listed below). Try to keep the maximum length of the report to 6 pages, including all figures and tables. Reports longer than six pages will only be graded up until the first six pages. You can use any software to create your report, but your report must be submitted in PDF format.
- **Code:** The second deliverable is the code that you wrote to answer the questions, which will involve implementing a regression models. Your code must be `Python 3.7` (no iPython notebooks or other formats). You may create any additional source files to structure your code.
- **Kaggle Submissions:** We will use `Kaggle`, a machine learning competition service, to evaluate the performance of your models. You will need to **register** on `Kaggle` using a `umass.edu` email address to submit to Kaggle (you can use any user name you like). You will generate test prediction files, save them in Kaggle format (helper code provided called `Code/kaggle.py`) and upload them to Kaggle for scoring.

Submitting Deliverables: When you complete the assignment, you will upload your report and your code using the `Gradescope.com` service. Here are the steps:

1. Place your final code in `Submission/Code`, and the Kaggle prediction files for your best-performing submission only for each data set in `Submission/Predictions/<Dataset>/best.csv`
2. Create a **zip** file of your submission directory, `Submission.zip` (No rar, tar or other formats).
3. Upload this single zip file on Gradescope as your solution to the `HW03-Kernel-Programming` assignment. Gradescope will run checks to determine if your submission contains the required files in the correct locations.
4. Upload your pdf report to the `HW03-Kernel-Report` assignment. When you upload your report please make sure to select the correct pages for each question respectively. Failure to select the correct pages will result in point deductions.
5. The submission time for your assignment is considered to be the latest of the submission timestamps of your code, report and Kaggle submissions.

Academic Honesty Policy: You are required to list the names of anyone you discuss problems with on the first page of your solutions. This includes teaching assistants or instructors. Copying any solution materials from external sources (books, web pages, etc.) or other students is considered cheating. To emphasize: no detectable copying is acceptable, even, e.g., copying a single sentence from an outside source. Sharing your code or solutions with other students is also considered cheating. Any detected cheating will result in a grade of -100% on the assignment for all students involved (negative credit), and potentially a grade of F in the course.

Data: You will work with three datasets,

- **Synthetic:** Use this dataset to check equivalence between basis expansion and the use of kernels in regression settings. You are provided four files: `data_train.txt, label_train.txt, data_test.txt and label_test.txt`.
- **Housing:** This dataset has 12 attributes and one output, all real numbers. The attributes correspond to different features of a house, and the output measures the price of the house. You are provided three files: `train_x.npy`, `train_y.npy` and `test_x.npy`.
- **Corona:** This dataset has 24 attributes and a binary output. We treat this as a classification problem. The attributes are different measurements of characteristics of a patient, which are useful in predicting whether the virus affects the patient adversely. You are provided three files: `train_x.npy, train_y.npy` and `test_x.npy`.

Code to load datasets is provided in `run_me.py`. Below is a summary of the datasets, allowed python functions per dataset and performance metric to report in your HW03 pdf report which matches with Kaggle's performance reporting.

Dataset	Python functions	Use	Metric
Synthetic	`sklearn.kernel_ridge.*`	Not allowed	Mean squared error
	Inbuilt Basis expansions	Not allowed	(regression)
	`sklearn.linear_model.Ridge`	Allowed	
Housing	`sklearn.kernel_ridge.*`	Allowed	Mean squared error
	`sklearn.model_selection.*`	Allowed	(regression)
Corona	`sklearn.svm.*`	Allowed	Accuracy
	`sklearn.model_selection.*`	Allowed	(classification)

1. **Kernel Ridge Regression:** This is a kernel regression method. Let $(x^{(1)}, y^{(1)}), \cdots, (x^{(N)}, y^{(N)})$ be the set of inputs, the problem consists on finding the w that minimizes

$$\sum_{n=1}^{N} (x^{(n)} \cdot w - y^{(n)})^2 + \lambda \|w\|_2^2, \tag{1}$$

where the first term corresponds to the squared loss in the estimation and the second is a regularization term. As seen during the lectures, one can apply a transformation to the samples (which increases their dimensionality) and perform the linear regression in this new higher dimensional space, which will correspond to a non-linear regression in the original space. An extremely important note regarding this is that, in order to estimate the value corresponding to a new sample x^{new}, only inner products like $x_{new} \cdot x^{(n)}$ are necessary. Thus, the kernel trick is applicable.

2. **SVM:** This is a classification method that can assign classes to new samples using only the inner product between the new samples and the samples in the training data. This allows us to use several different kernels, which makes SVM an extremely powerful classification method.

Questions:

1. (*0 points*) **Collaboration statement**:
Please list the names of anyone you discussed the assignment with, including TAs or instructors.

2. (*70 points*) **Kernel Ridge Regression:** For parts a,b and c, assume that $x^{(n)}, \phi(x^{(n)})$ are column vectors

(11) a. We begin with regular ridge regression. The goal is to find w that minimizes

$$\sum_{i} (w \cdot x^{(n)} - y^{(n)})^2 + \lambda \|w\|_2^2. \tag{2}$$

Show that the optimal w is

$$w^* = \big(\sum_{n=1}^{N} x^{(n)} \, x^{(n)^\top} + \lambda \, I \big)^{-1} \sum_{n=1}^{N} x^{(n)} \, y^{(n)}. \tag{3}$$

(4) b. Now, we consider the case where a basis expansion is used. So, each input x is transformed into $\phi(x)$. The goal is thus to instead minimize

$$\sum_{i} (w \cdot \phi(x^{(n)}) - y^{(n)})^2 + \lambda \|w\|_2^2. \tag{4}$$

What now is the optimal w^*? Argue why your answer is correct.

(5) c. Once the optimal w^*, is found, it would be used to estimate the value of a new point as y^{new} from its corresponding input as $y^{\text{pred}} = w^* \cdot \phi(x^{\text{new}})$.

However, for this type of regression the kernel trick is applicable. Formally, this means that it is not necessary to know the transformation ϕ; having an expression for $k(x^{(1)}, x^{(2)}) = \phi(x^{(1)}) \cdot \phi(x^{(2)})$ (the inner product of two samples in the new space) suffices. Given a new sample x^{new}, derive an expression for the predicted value y^{pred} that depends only on inner products between samples.

(30) d. In this exercise you will perform Kernel Ridge Regression using the synthetic dataset with inputs $x^{(n)} \in \mathbb{R}$ and outputs $y^{(n)} \in \mathbb{R}$. The training data is in `data_train.txt, label_train.txt`, and the testing data is in `data_test.txt, label_test.txt`. The training data is shown in Fig. 1.

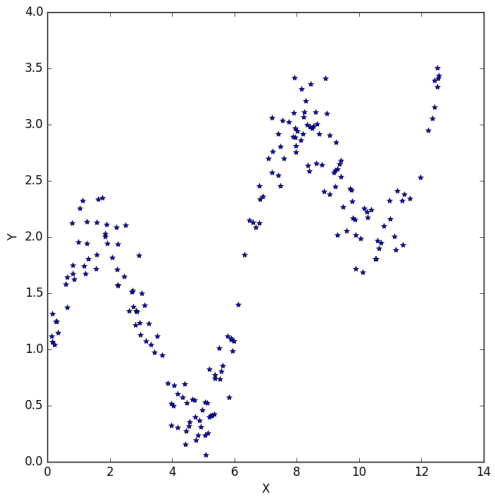

Figure 1: Data

For this question, we would like you to implement your **own handwritten** kernel ridge regression and basis expansions. However, you are allowed to sklearn's implementation of ridge regression. (**linear** ridge regression, not kernel ridge regression.) Again, you are **not** allowed to use `sklearn.kernel_ridge.KernelRidge` or any of its variants in this problem.

We would like you to compare the performance of the two approaches on the synthetic dataset to better understand kernel methods. As mentioned in the class, the implementation using a kernel $k(x^{(1)}, x^{(2)}) = \phi(x^{(1)}) \cdot \phi(x^{(2)})$ should return the same results as the implementation using the basis expansion Φ followed by ridge regression. You will use the following parameters for KRRS/BERRs (with $\lambda = 0.1$):

Polynomial order i: (for $i \in \{1, 2, 4, 6\}$)

1. Kernel ridge regression scratch (KRRS):

$$k(x^{(1)}, x^{(2)}) = (1 + x^{(1)} \cdot x^{(2)})^i \tag{5}$$

2. Basis expansion + ridge regression (BERR):

$$\Phi(x) = \left[w_0\ 1,\ w_1\ x^1,\ w_2\ x^2,\ ...,\ w_i\ x^i\right], w_k = \sqrt{\binom{i}{k}} \tag{6}$$

Trigonometric order i: (for $i \in \{3, 5, 10\}$ and $\delta = 0.5$)

1. Kernel ridge regression scratch (KRRS):

$$k(x^{(1)}, x^{(2)}) = 1 + \sum_{k=1}^{i} (\sin(k\,\delta\,x^{(1)})\sin(k\,\delta\,x^{(2)}) + \cos(k\,\delta\,x^{(1)}) \cdot \cos(k\,\delta\,x^{(2)})) \tag{7}$$

2. Basis expansion + ridge regression (BERR):

$$\Phi(x) = [1, \sin(\delta\,x), \cos(\delta\,x), \sin(2\delta\,x), \cos(2\delta\,x), ..., \sin(i\,\delta\,x), \cos(i\,\delta\,x)] \tag{8}$$

1. Train your models using the train data and predict the outputs for the test data. Visually inspect the quality of predictions by plotting the predicted test labels on top of the original test labels as a function of test data. Make these plots corresponding to polynomial kernels of degree 2 and 6, and trigonometric kernels of degree 5 and 10 only. A sample plot is provided in Figure 2. In each plot include the test samples as blue '*' and plot the predictions made on the test set as red circles. Make sure to label the axis, and include a title for each graph with implementation, kernel, degree and lambda used. Arrange your plots as a 4×2 grid (i.e. four rows and 2 columns; see `matplotlib.pyplot.subplot`) where the four rows correspond to the kernels and the two columns correspond to the two implementations. Write in at most three sentences what your observations are about these eight plots.

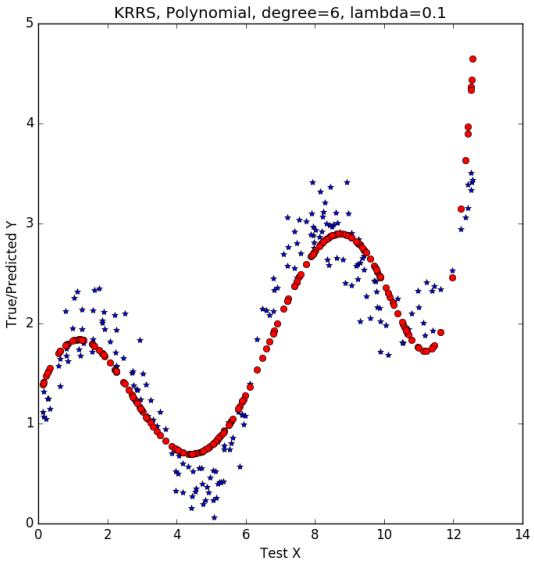

Figure 2: Test samples are plotted in blue '*' and predicted test labels are plotted in red 'o'

2. For each of the kernels/basis expansions mentioned above, show the mean squared error over the **test set** using a table like the one below (please make the table exactly as shown below to make grading more efficient),

	Kernel $k(x^{(1)}, x^{(2)})$	Basis Expansion $\Phi(x)$
Polynomial degree 1		
Polynomial degree 2		
Polynomial degree 4		
Polynomial degree 6		
Trignometric degree 3		
Trignometric degree 5		
Trignometric degree 10		

where in each empty box you need to write the mean squared error obtained via KRRS or BERR.

(20) e. In this exercise you will perform Kernel Ridge Regression using the housing dataset. For this question you are allowed to (and encouraged) use sklearn's implementation of Kernel Ridge Regression. You will train several kernels (radial basis functions, polynomial degree 3, linear) with parameters - $\alpha \in \{1, 1e-1, 1e-2, 1e-3, 5e-3\}$ and $\gamma \in \{1e-3, 1e-2, 1e-1, 1\}$ (you should keep in mind the γ parameter does not have any effect on linear kernels). For each kernel, pick your choice of model selection method to estimate the out of sample mean squared error. Report the results in a table format as shown below (please make the table format exactly as shown below to make grading more efficient),

	RBF	Poly. degree 3	Linear
$\alpha = 1, \gamma = 1e-3$			
$\alpha = 1, \gamma = 1e-2$			—
$\alpha = 1, \gamma = 1e-1$			—
$\ldots, \ldots$			—
$\alpha = 5e-3, \gamma = 1e-2$			—
$\alpha = 5e-3, \gamma = 1e-1$			—
$\alpha = 5e-3, \gamma = 1$			—

Choose the model with lowest estimated out of sample error, train it using the full training set, make predictions on the test set, kaggleize your outputs (helper code given in `Code/kaggle.py`) and finally upload your predictions to Kaggle. The Kaggle link to this question is https://www.kaggle.com/t/75695499abab4b40ab556778547d150e.
Report back the mean squared error that you obtained on the public leader board as well as your Kaggle display name so that we can match your results with your rank on the leaderboard. Make sure to clearly indicate which model provided the best results as well as your choice of model selection method. Finally, consider the results that you obtained on the train set (via model selection) as well as your test results. Compare the two and draw some conclusions on your chosen kernel, specific parameter settings and its relation to the dataset itself.

3. (*25 points*) SVM:

(25) a. For this question you will use perform classification using a SVM on the corona dataset. For this question you are allowed to (and encouraged) use sklearn's implementation of SVM. You will train an SVM classifier using the kernels shown in the below table, and complete the table using estimates of the out of sample accuracy (NOT error but classification accuracy) obtained using your choice of model selection method.

	RBF	Poly. degree 3	Poly. degree 5	Linear
$C = 0.5, \gamma = 1$				
$C = 0.5, \gamma = 0.01$				—
$C = 0.5, \gamma = 0.001$				—
$C = 0.05, \gamma = 1$				
$C = 0.05, \gamma = 0.01$				—
$C = 0.05, \gamma = 0.001$				—
$C = 0.0005, \gamma = 1$				
$C = 0.0005, \gamma = 0.01$				—
$C = 0.0005, \gamma = 0.001$				—

Again here the γ parameter does not have any effect on linear kernels.

Choose the model with highest estimated out of sample accuracy, train it using the full training set, make predictions on the test set, kaggleize your outputs (helper code given in `Code/kaggle.py`) and finally upload your predictions to Kaggle. The Kaggle link to this question is https://www.kaggle.com/t/d40af1477fdf4b988db745b02a2c7a42.

Report back the classification accuracy that you obtained on the public leader board as well as your Kaggle display name so that we can match your results with your rank on the leaderboard. Make sure to clearly indicate which model provided the best results and your choice of model selection method. Finally, consider the results that you obtained on the train set (via model selection) as well as your test results. Compare the two and draw some conclusions on your chosen kernel, specific parameter settings and its relation to the dataset itself.

Extra Credit: These questions are deliberately open-ended, leaving you more space for creativity. As a result, you will need to carefully describe exactly what you did for each question. Also, note that these questions carry small point values. To maximize your score with limited time, you should make sure the above questions are done thoroughly and ignore these. We will be very stingy in giving credit for these questions – do them only for the glory, and only at your own risk!

4. (*5 points*) **Extra credit:** For the housing dataset perform regression with your choice of kernels (you are free to even create your own kernels). Experiment with different parameter ranges and/or model selection methods with a single goal of improving performance on the held out test set. For this extra credit question the kaggle URL to submit your predictions is
`https://www.kaggle.com/t/bbdec9031e194dd99e02ca749e6d80f7`.
Make sure to clearly describe your approach and list your performance on the public leaderboard. To avoid any confusion make sure to write the outputs to Predictions/Housing/best_extra_credit.csv. Note that your grades will be dependent on your ranking in the private leaderboard.

5. (*5 points*) **Extra credit:** For the corona dataset perform classification with your choice of kernels. Experiment with different parameter ranges and/or model selection methods with a single goal of improving performance on the held out test set. For this extra credit question the kaggle URL to submit your predictions is
`https://www.kaggle.com/t/30cad0133e66430194a6da7afac05ea3`.
Make sure to clearly describe your approach and list your performance on the public leaderboard. To avoid any confusion make sure to write the outputs to Predictions/Corona/best_extra_credit.csv. Note that your grades will be dependent on your ranking in the private leaderboard.

6. (*5 points*) **Code Quality:**

(5) Your code should be sufficiently documented and commented that someone else (in particular, the TAs and graders) can easily understand what each method is doing. Adherence to a particular Python style guide is not required, but if you need a refresher on what well-structured Python should look like, see the Google Python Style Guide: `https://google.github.io/styleguide/pyguide.html`. You will be scored on how well documented and structured your code is.