Assignment 4

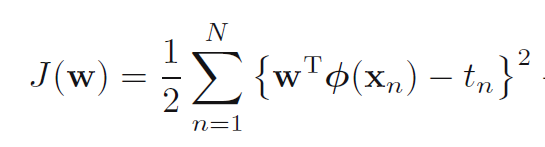
## **Task1**. Dual Representation

### Background

In this assignment, we want to validate the dual representation in regularized linear regression. Specifically, we will build a linear regression as a non-parametric model.

### Requirement

1. Use the same functions(create\_toy\_data() and f()) from assignment 1, task one to generate 10 data for training,and 100 samples for testing.
2. Use polynomial basis function(PolynomialFeatures() from sklearn) to convert your training and testing data into matrix Phi. Set polynomial order = 9
3. Now we want to study the dual representation for the following error function:



(Hint you can use LinearRegression() from sklearn to train this model)

Take a careful look at this loss, if you want to use LinearRegression(), you might want to change some of its default settings.

Train the model using your Phi matrix, save the predictions on the test data. and report the **test MSE**.

1. Now use the dual model(the kernel representation) to make the prediction on the test data. You need to first compute the Gram matrix K, and the vector k(x). Your prediction rule should be similar, but not identical, to eq 6.9 from the textbook. Save the predictions on the test data. and report the **test MSE**.
2. Compare the predictions from 4 and 5, are they identical? Why or why not?

## **Task2**. Gaussian Processes for Regression

### Background

As shown in the previous task, the kernel method is easy to ‘train’. But recall that we have to explicitly use the results of the basis function(phi(x)) to derive the Gram matrix K. This does not grant us much freedom because designing the basis function itself is a difficult task. In this task, we will unleash the true power of the kernel machine. Pay attention to how we manage to use a different way to define the gram matrix without knowing what basis function generates it.

1. First, we need to define the gram matrix using the kernel function k(x,x’). We will try the classical RBF(Radias Basis Function) kernel. You can find the function here.

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.rbf_kernel.html>

Set parameter gamma to 5.

(HInt you should use the raw data to generate Gram matrix. That is the whole point of using kernel: avoid defining basis explicitly)

From the definition of the kernel, you can see that every kernel function has its corresponding basis function (phi(x)).You might wonder what basis function (phi(x)) the RBF kernel corresponds to. The answer is quite intriguing You can find the answer in exercise 6.11.

1. Next, we need to compute the C matrix, which is the covariance of the target values. Recall how GP is derived in 6.4.2 from our textbook. Set beta in eq 6.58 to 10 and compute the C matrix according to eq 6.62.

It is usually a good habit to check whether the covariance is invertible. Check that in your code.

1. Compute the predictive mean for all the test samples. Note eq 6.66 only tells you how to compute the mean for one test data point. But you should be able to extend them to predict multiple test samples at once. (Hint, you need to extend the k vector)

You can not use a ‘for loop’ to predict the mean for all the test samples.

Print the test MSE.

(optional)You can try different gamma values and see how they affect the model performance.

(optimal) You can compute the predictive variance in a similar way.

## **Task3**. SVMs for Classification

1. We will practice a simple multi-classification task on the iris dataset using SVM.

Carefully read the API for the dataset, and the model

<https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html>

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC>

Read the iris dataset. And use train\_test\_split() from sklearn to split the dataset into training and testing. Set the testing\_size to 0.3, random\_state to 5.

1. Create a SVC model and train it using the training data. You do not have to specify any parameters when creating SVC. Just use the default parameters.

Answer the following questions in your code:

2.1: In our class, SVM is introduced as a binary classifier. But iris has 3 classes. How does SVC handle this issue?

2.2: How many support vectors do we learn from the data? validate your answer with some codes.

2.3: Is the 18th training sample in the training dataset a support vector? validate your answer with some codes.

2.4: How many support vectors are in class 2?(note the class index starts from 0)validate your answer with some codes.

1. Report the classification test accuracy.

------Below are open research bonus assignments; you are expected to do some extra reading/research to complete the task.-----

## **Task4**.[Bonus] Bayesian Optimization

-----You need to complete this task in a separate .py file.----

### Background:

In Bayesian Optimization, we have a black box function f(x), and we want to find x that minimizes f. Of course, you can try all the available x and see which one gives you the smallest f(x). But unfortunately, there is a cost associated with calling f(). Imagine x is the experimental setting for a large project, and f(x) is the critical outcome from actually conducting the experiment( eg. launching a rock). We can not afford randomly test many x, we want to be more efficient. The key is to use a surrogate model of f()(e.g. A GP) to help us select the next x through some acquisition function. Study how Bayesian Optimization can be done with a Gaussian Processes model and an iterative learning approach.

1. Read the x.npy and y.npy using numpy.load(). Each row of x is a data instance, and each row of y is f(x). Our task is to find the x that minimizes f(x) using Bayesian optimization.
2. Randomly select 10 samples from x, and assume we know their f(x) already, and treat all the rest samples as candidate samples that you can select and reveal their target values. Start the Bayesian optimization with these 10 samples. Use GP with the mean and variance function m(x), var(x) as the surrogate model and use a(x) = m(x)/var(x) as the acquisition function. You can use the sklearn implementation of GP <https://scikit-learn.org/stable/modules/gaussian_process.html>

You need to use the Bayesian optimization to select 500 samples.(Hint: do not forget to re-train your GP after each sample has been selected).

1. The first result you need to show is a curve plot where the x-axis indicates the iteration number of the Bayesian optimization and the y-axis represents the f(x) value for x you selected in that iteration. You need to draw two curves, one is the Bayesian optimization curve in which x is selected according to the acquisition function we give you. The other is the random curve(which acts as a baseline) in which x is selected randomly from the candidate pool.
2. The second result you need to show is the so-called ‘cumulative epsilon-optimal sample curve.’ Suppose y\_min is the global minimal value in y. The epsilon optimal samples are defined as {x: |f(x)-y\_min|/y\_min<epsilon}. Set epsilon = 0.5 in this case. The x-axis in your plot is the same as the previous task. The y-axis should indicate how many epsilon optimal samples you have already found at the corresponding iteration. You also need to draw two curves, one for Bayesian optimization, and one for random sampling.

Grade Sheet:

| Task | Grade |
| --- | --- |
| 1 | \_\_\_/12% |
| 2 | \_\_\_/10% |
| 3 | \_\_\_/10% |
| 4[Bonus] | \_\_\_/68% |