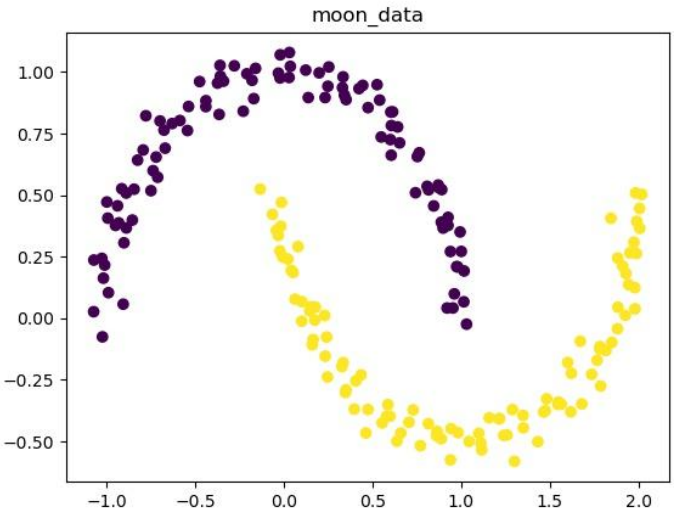
Fourier Transform and Random Fourier Features

This is an exercise problem of the digital signal processing (DSP) course at School of Artificial Intelligence at the Nanjing University (NJU), teaching by Han-Jia Ye. The course homepage is at DSP. This exercise is written by Jia-Qi Yang. Please feel free to contact me by mailing yangjq@lamda.nju.edu.cn if you have any questions.

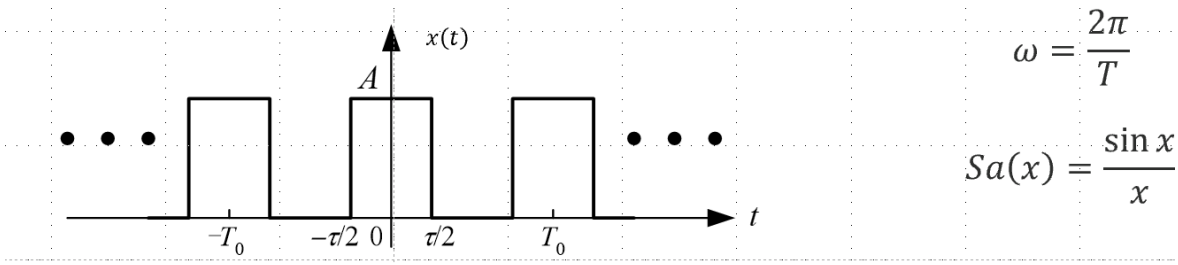
Please read before you dig into the problems:

1. Do not distribute your answer on the internet.
2. The default dataset is "moon" in data.py, the number of points may be tuned.
3. You are supposed not to use any additional packages other than those used in the demo code to implement functions. However, you may visualize your results using any tool you like.
4. You should submit a zip file containing your source code and a report.



Problem 1: Fourier Transform (35pt)

The periodic square wave signal is defined in the following figure:



$$\omega = \frac{2\pi}{T}$$

$$Sa(x) = \frac{\sin x}{x}$$

We can infer from the figure that:

1. When τ approaches T_0, the signal approaches a direct current signal with amplitude A.

1.1: What else can you infer from the definition ? (5pt)

Hint: consider influence of τ and T_0 on time domain and spectrum domain.

1.2: Calculate the fourier transform of this signal. (10pt)

1.3: Plot the spectrum using the results from 1.2, verify your answer in 1.1 visually. (20pt)

Hint: how will the spectrum change while varying τ and T_0?

Problem 2: Random Fourier Features (65pt)

You need to implement non-linear SVM using Random Fourier Features (RFF) in this problem.

2.1: RFF kernel (20pt)

Implement the *RFF_kernel_fn* function in practice.py. This kernel function should approximate the RBF kernel defined in *RBF_kernel_fn*. Please refer to *linear_kernel_fn* and *RBF_kernel_fn* function in examples.py to see how it works.

Answer following questions:

1. How the dimension (D in the paper) of random fourier features affect the precision of approximation (complete *test_RFF_kernel_fn*)? Investigate this question by computational experiments, theoretical analysis is **not** required. (10pt)
2. How x_dim and D affect the speed of RBF kernel and RFF kernel? Investigate this question by computational experiments, **or** analyze the computational complexity of using RFF kernel and RBF kernel to compute the gram matrix. (10pt)

2.2: RFF kernel in SVM (25pt)

1. Run the *demo_plot_data demo_linear_svm demo_rbf_svm* functions in practice.py. How does the decision boundary learned by linear kernel and RBF kernel look like? How do they perform on the moon data? (5pt)

2. Implement the *test_RFF_kernel_svm* function using *RFF_kernel_fn* defined in problem 2.1. Plot and compare the decision boundary of RFF kernel and RBF kernel on the moon dataset. Please refer to *demo_rbf_svm* for example. (10pt)

3. Play with different number of features (the value of D), and answer following questions:

    1. How does D affect the decision boundary of RFF kernel? (5pt)
    2. Why is this approach very slow with large D or large dataset? (5pt)

Hints:

1. Set boundary=False in plot_decision_boundary to avoid the overhead of plotting when comparing speed.
2. Try to adjust the number of data points in data.py, the default n_samples=200 may be too small to compare speed.

2.3: Fast Approximation of RBF-SVM Using RFF (20pt)

It is known that SVM with linear kernel can be much faster than SVMs with non-linear kernel. However, linear kernel will not work well on problems that are not linearly separable. How can we achieve

performance similar with non-linear kernels (such as RBF kernel) as well as fast speed comparable with linear kernel? RFF is designed to enable training with large-scale kernel machines.

Implement a fast version of SVM with RBF kernel approximated by RFF, and answer following questions:

1. Plot the decision boundary of RFF-SVM and compare with RBF-SVM. (5pt)
2. Report and compare the testing accuracies between the RBF-SVM and RFF-SVM (on the moon dataset), do they perform similar? (5pt)
3. Report and compare the training and inference time between the RBF-SVM and RFF-SVM, and try to analyze the reasons. (10pt)

Hint:

1. The sklearn.svm.SVC(kernel="linear") is slow, use sklearn.svm.LinearSVC to implement linear SVM.

Further Reading

1. Another popular approach to accelerate kernel machines is the Nystroem method, which is implemented in the sklearn package. It randomly samples a subset of training examples and computes a kernel matrix $\hat{K}$ for the random samples. It then represents each data point by by a vector based on its kernel similarity to the random samples and the sampled kernel matrix $\hat{K}$.
2. An important difference between RFF and Nystroem method is that Nystroem method approximates a **data dependent** kernel, while RFF is **data independent**. Thus Nystroem method may perform better than RFF in some case. Please refer to Yang. et al. for more details.
3. A kernel of RFF is determined by the sampling distribution $p(\omega)$, can we learn this distribution? There has been a paper about this idea.

Bonus !

If you

1. Found any mistake or typo.
2. Come up with some nice ideas to improve this exercise.

Please mail to yangjq@lamda.nju.edu.cn, we will give you some additional points (total points of this exercise will not exceed 100pt) if your suggestion is accepted.