

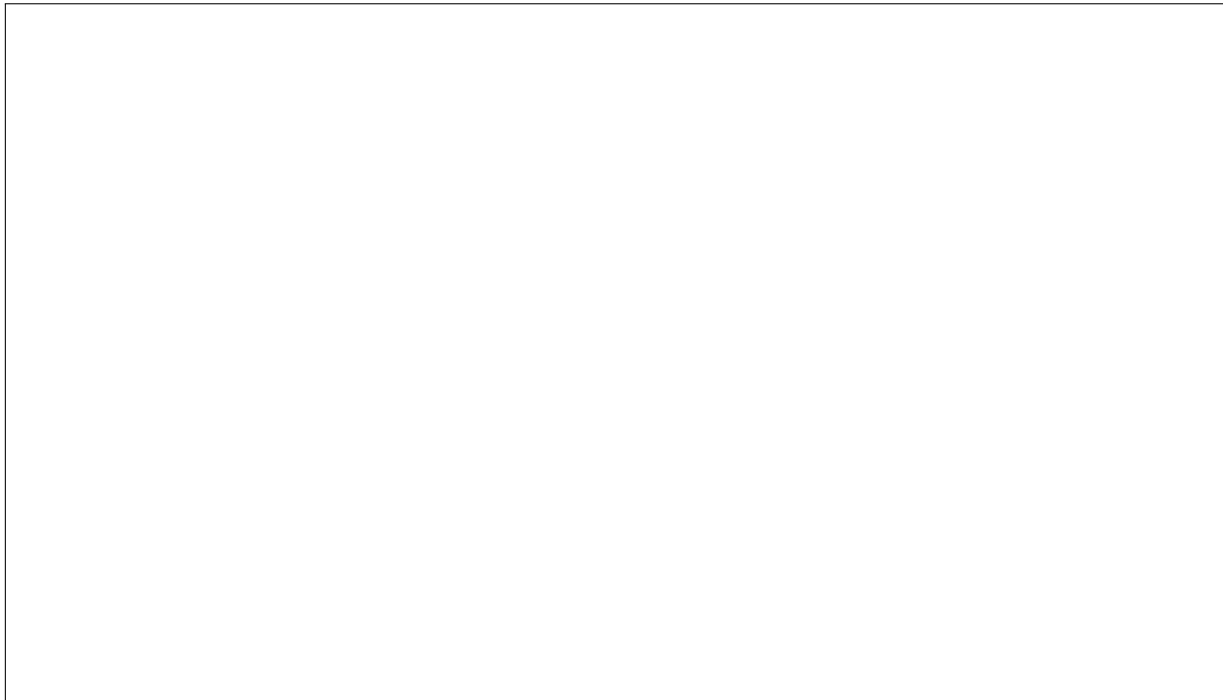
INSTRUCTIONS

- **Due:** Wednesday, 6 April 2022 at 11:59 PM EDT.
- **Format:** Complete this pdf with your work and answers. Whether you edit the latex source, use a pdf annotator, or hand write / scan, make sure that your answers (tex'ed, typed, or handwritten) are within the dedicated regions for each question/part. If you do not follow this format, we may deduct points.
- **How to submit:** Submit a pdf with your answers on Gradescope. Log in and click on our class 10-315, click on the appropriate *Written* assignment, and upload your pdf containing your answers. Don't forget to submit the associated *Programming* component on Gradescope if there is any programming required.
- **Policy:** See the course website for homework policies and Academic Integrity.

Name	
Andrew ID	
Hours to complete (both written and programming)?	

Q1. [20pts] Linear and Kernel SVMs

- (a) [12pts] We are given two sets of data points $A = \{(0, 0), (1, 1), (2, 2)\}$ and $B = \{(0, 1), (1, 2), P\}$, where $P = (u, v)$ is an unknown point on the 2D Cartesian coordinate plane. Assume these points can be perfectly classified, such that the points in A are labeled $+1$ and the points in B are labeled -1 , using a decision boundary of the form $y = wx + b$ for $w, b \in \mathbb{R}$. (That is, the boundary is the set of (x, y) points that satisfy this equation.) If the best possible decision boundary for this split is **not** $y = x + \frac{1}{2}$, what are all of the possible points that P can be? (Here, the “best” possible decision boundary means the maximum margin decision boundary.)



- (b) [8pts] You are given a training dataset, as shown in Fig 2. Note that the training data comes from sensors which can be error-prone, so you should avoid trusting any specific point too much. For this problem, assume that we are training an SVM with a quadratic kernel.

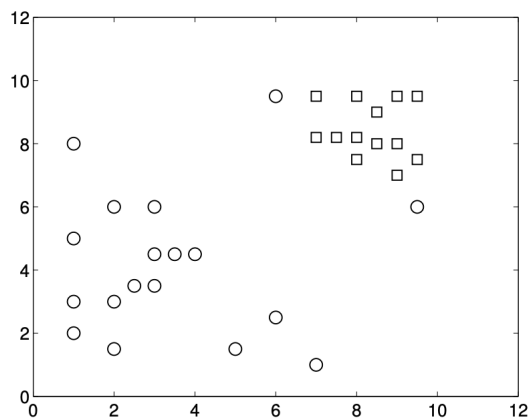
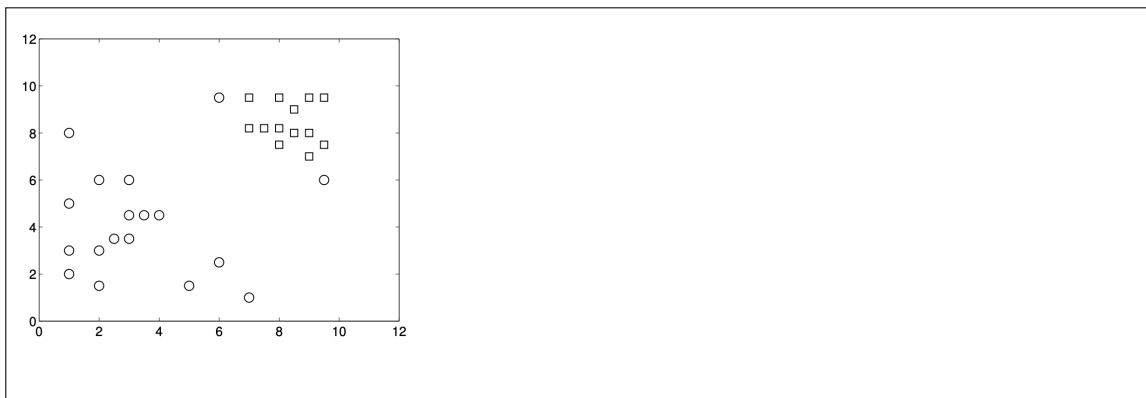
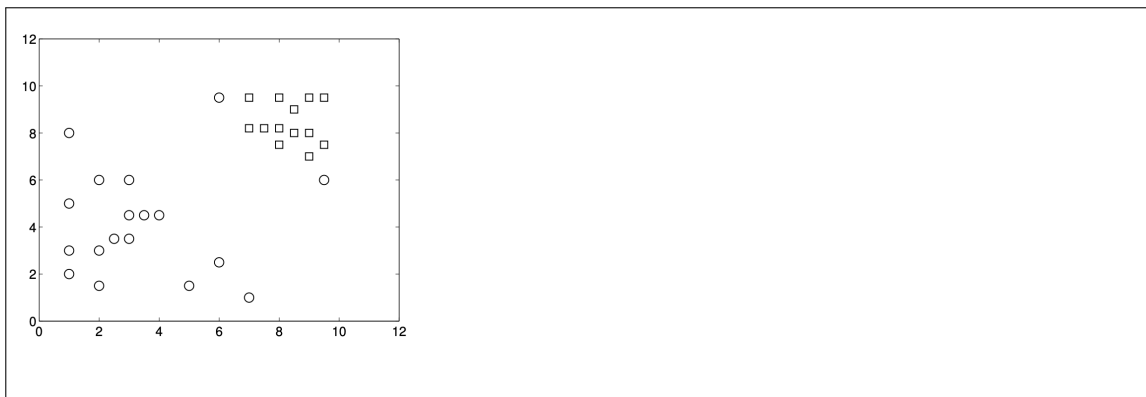


Figure 2: Training dataset

- (i) [3pts] Where would the decision boundary be for very large values of C (i.e., $C \rightarrow \infty$)? Draw on the figure below and justify your answer.



- (ii) [3pts] For C close to 0, indicate in the figure below where you would expect the decision boundary to be? Justify your answer.



- (iii) [2pts] Which of the two cases above would you expect to work better in the classification task? Why?

Q2. [25pts] Kernels

(a) Kernel Computation Cost

- (i) [4pts] Suppose we have a two-dimensional input space such that the input vector is $\mathbf{x} = [x_1, x_2]^T$. Define the feature mapping $\phi(\mathbf{x}) = [x_1^2, \sqrt{2}x_1x_2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1]^T$. What is the corresponding kernel function, i.e. $k(\mathbf{x}, \mathbf{z})$? Do not leave $\phi(\cdot)$ in your final answer. Simplify your answer to write it using input vectors \mathbf{x}, \mathbf{z} . In order to receive full credit, you must show your work.

- (ii) [4pts] Suppose we want to compute the value of the kernel function $k(\mathbf{x}, \mathbf{z})$ from the previous question, on two vectors $\mathbf{x}, \mathbf{z} \in \mathbb{R}^2$. How many operations (additions, multiplications, powers) are needed if you map the input vector to the feature space and then perform the dot product on the mapped features? Show your work.

Num:

Work:

- (iii) [2pts] How many operations (additions, multiplications, powers) are needed if you compute through the kernel function you derived in question 1? Show your work.

Num:

Work:

(b) [5pts] Sum of Kernels

Assume $k_1(\cdot, \cdot)$ is a kernel with corresponding feature mapping $\phi_1 : \mathbb{R}^M \rightarrow \mathbb{R}^{M_1}$, and $k_2(\cdot, \cdot)$ is a kernel with corresponding feature mapping $\phi_2 : \mathbb{R}^M \rightarrow \mathbb{R}^{M_2}$, both acting on the same space. Prove that, $k'(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$ is also a valid kernel by constructing its corresponding feature mapping $\phi'(\cdot)$.

(c) [10pts] Which of the following are valid kernels? Explain your reasoning for each option. Data points x, z are scalars and \mathbf{x}, \mathbf{z} are vectors.

(i) [2pts] $K(x, z) = -xz$

(ii) [2pts] $K(\mathbf{x}, \mathbf{z}) = 10\mathbf{x} \cdot \mathbf{z} + (\mathbf{x} \cdot \mathbf{z} + 1)^8$

(iii) [2pts] $K(x, z) = x^2 z$

(iv) [2pts] $K(\mathbf{x}, \mathbf{z}) = -\exp(\|\mathbf{x} - \mathbf{z}\|^2)$

(v) [2pts] $K(\mathbf{x}, \mathbf{z}) = \exp(-(\|\mathbf{x}\|^2 + \|\mathbf{z}\|^2))$

Q3. [35pts] Programming - Nonparametric kernel regression

This assignment includes an autograder for you to grade your answers on your machine. This can be run with the command:

```
python3.6 autograder.py
```

The code for this assignment consists of several Python files, some of which you will need to read and understand in order to complete the assignment, and some of which you can ignore.

Files you will edit:

- `kernel_regression.py`: Your code to implement nonparametric kernel regression tasks.
- `additional_code.py`: You shouldn't need this file for this assignment, but it is provided just in case you have additional code that doesn't fit into `kernel_regression.py` for some reason. If you do submit this file the code should be runnable by calling `python3.6 additional_code.py`, but there are no requirements on the format and it will not be executed by the autograder.

Files you might want to look at:

- `util.py`: Convenience methods to generate various plots that will be needed in this assignment.
- `test_cases/Q*/*.py`: These are the unit tests that the autograder runs. Ideally, you would be writing these unit tests yourself, but we are saving you a bit of time and allowing the autograder to check these things. You should definitely be looking at these to see what is and is not being tested. These test cases also generate plots related to the task; the plots are saved in a directory named `figures`. The autograder on Gradescope may run a different version of these unit tests.

Files you can safely ignore:

- `autograder.py`: Autograder infrastructure code.

Files to Edit and Submit:

You will fill in portions of `kernel_regression.py` (and `additional_code.py`, if necessary) during the assignment. You should submit this file containing your code and comments to the Programming component on Gradescope. Please do not change the other files in this distribution or submit any of our original files other than these files. Please do not change the names of any provided functions or classes within the code, or you will wreak havoc on the autograder.

(a) [20pts] Kernel Functions

In `kernel_regression.py`, implement the following kernels in their respective functions. See function docstring for details.

Kernel	Function	Equation
RBF	<code>kernel_rbf(x, z, h)</code>	$e^{-\frac{1}{h^2} \ x-z\ _2^2}$
Boxcar	<code>kernel_boxcar(x, z, h)</code>	$\mathbb{1}[\ x-z\ \leq \frac{h}{2}]$

You may run the following command to run a quick unit test on your Q1 implementation:

```
python3.6 autograder.py -q Q1
```

We encourage you to write your own code to test out your implementation as you work through the assignment. For example, you may want to use some of the functions in `util.py` to plot the functions you just implemented.

The autograder will also generate plots of your kernel functions for a fixed 2D point \mathbf{z} and various hyperparameter settings. It will save these plots as png files in a new directory named **figures**. You are required to include some of these figures as part of the written component of this assignment. See the written component for details about which plots to include.

(b) [15pts] Kernel Regression

In the `predict_kernel_regression` function in `kernel_regression.py`, implement nonparametric kernel regression as defined in lecture and use it to predict the output values for a set of input points, \mathbf{X} . See function docstring for details.

You may run the following command to run a quick unit test on your Q2 implementation:

```
python3.6 autograder.py -q Q2
```

The autograder will also generate plots of your regression prediction for different kernels, different numbers of training points, and different hyperparameter settings. You are required to include some of these figures as part of the written component of this assignment. See the written component for details about which plots to include.

Submission

Complete all questions as specified in the above instructions. Then upload `kernel_regression.py` (and `additional_code.py` if you used it) to Gradescope. Your submission should finish running within 10 minutes, after which it will time out on Gradescope.

Don't forget to include any request results in the PDF of the Written component, which is to be submitted on Gradescope as well.

You may submit to Gradescope as many times as you like. You may also run the autograder on your own machine to speed up the development process. Just note that the autograder on Gradescope will be slightly different than the local autograder. The autograder can be invoked on your own machine using the command:

```
python3.6 autograder.py
```

Note that running the autograder locally will **not** register your grades with us. Remember to submit your code when you want to register your grades for this assignment.

The autograder on Gradescope might take a while but don't worry: **so long as you submit before the deadline, it's not late.**

Q4. [20pts] Programming - Continued

The following questions should be completed after you work through the programming portion of this assignment.

(a) [4pts] Kernel Functions

Include surface plots for the boxcar kernel and the RBF kernel with $h = 0.5$ and $\mathbf{z} = (2, -3)$. Your surface plot should represent the values the function takes on around \mathbf{z} with the given hyperparameter settings. Use <https://matplotlib.org/stable/gallery/mplot3d/surface3d.html> for suggestions on how to get the surface plots up and running.

Plot boxcar:

Plot RBF:

(b) Kernel Regression

- (i)** [4pts] Include surface plots for the kernel regression using the linear dataset with the boxcar kernel and the RBF kernel with $h = 0.5$. These will be automatically generated for you by the autograder when you run `autograder.py`.

Plot boxcar:

Plot RBF:

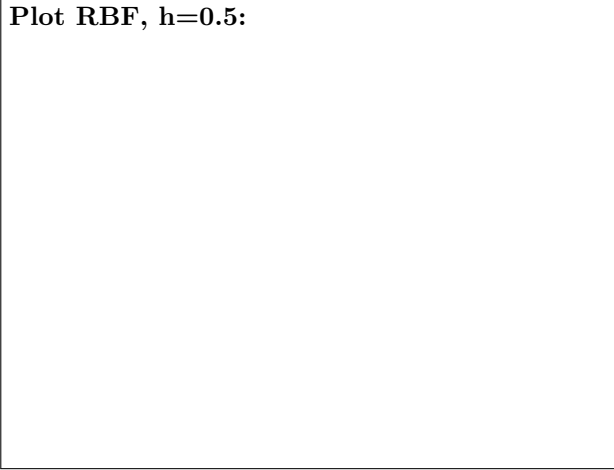
- (ii) [4pts] Include surface plots for the kernel regression with the nonlinear dataset with the boxcar kernel with $h = 1$ and 2.

Plot boxcar, $h=1$:

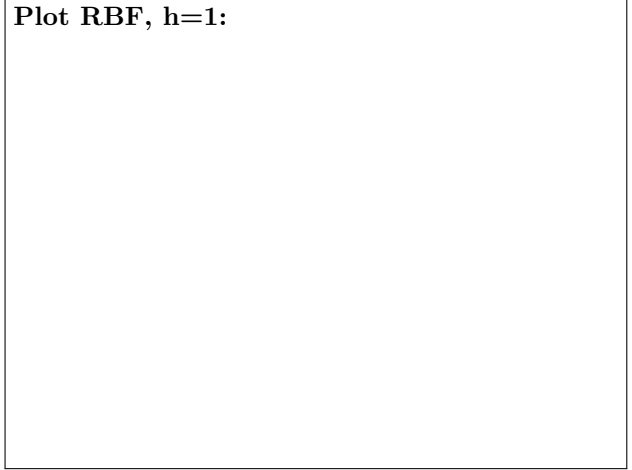
Plot boxcar, $h=2$:

- (iii) [5pts] Include surface plots for the kernel regression with the nonlinear dataset with the RBF kernel with $h = 0.5, 1$, and 2 .

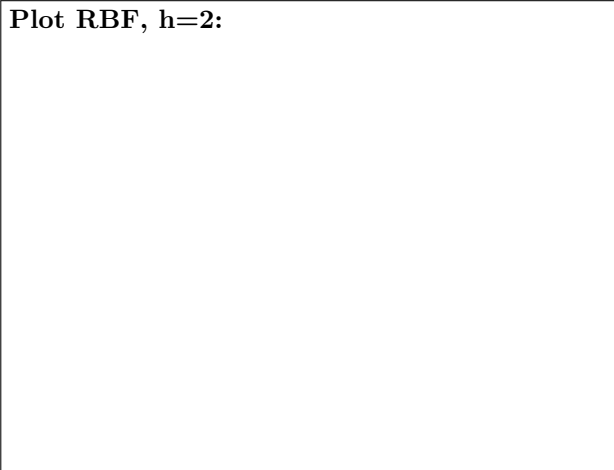
Plot RBF, $h=0.5$:



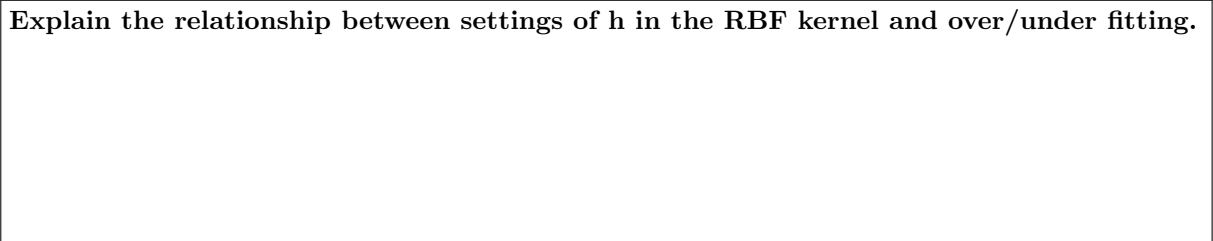
Plot RBF, $h=1$:



Plot RBF, $h=2$:



Explain the relationship between settings of h in the RBF kernel and over/under fitting.



- (iv) [3pts] Among all of the kernels and hyperparameter settings that the autograder test cases ran through, which kernel and hyperparameter combination should you choose? Why?

Answer:

