Name: _____ NetID: _____

S&DS 365 / 665

Intermediate Machine Learning

Midterm Exam

Wednesday, March 16, 2022

Complete all of the problems. You have 75 minutes to complete the exam.

The exam is closed book, computer, phone, etc. You are allowed one double-sided $8\frac{1}{2} \times 11$ sheet of paper with hand-written notes.

The following facts may (or may not) be helpful:

- If (X_1, X_2) are jointly Gaussian with distribution

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} A & C \\ C^T & B \end{pmatrix} \right)$$

  then the conditional distributions are also Gaussian and given by

$$\begin{aligned} X_1 \,|\, x_2 &\sim N\left(\mu_1 + CB^{-1}(x_2 - \mu_2),\ A - CB^{-1}C^T\right) \\ X_2 \,|\, x_1 &\sim N\left(\mu_2 + C^T A^{-1}(x_1 - \mu_1),\ B - C^T A^{-1}C\right) \end{aligned}$$

- If $p(x, y)$ is a joint distribution, then $p(x)$ is called the marginal over x. One can sample from the marginal by first sampling y from $p(y)$ and then sampling x from the conditional $p(x \,|\, y)$.
- For Dirichlet processes, one can sample X from the marginal by first sampling $F \sim DP(\alpha, F_0)$ and then sampling X from F. The Chinese restaurant process enables sampling from the marginal without sampling F.
- The function `np.linalg.inv` computes the inverse of a matrix.
- The function `numpy.random.choice(a, size, p)` generates a random sample of size `size` by sampling elements of a given array `a`, with weights `p`.
- The function `scipy.stats.norm.rvs(loc, scale, size)` generates a random sample of size `size` from a normal with mean `loc` and standard deviation `scale`.

1. ***Multinomial choice*** (10 points)

   For each of the following questions, circle the *single best* answer.

   1.1. To carry out ridge regression, suppose that $\mathbb{X}$ is the $n \times p$ design matrix and Y is the n-vector of responses. The estimator with regularization parameter λ has a closed form, and the fitted values $\widehat{Y}$ are given by

   (a) $\mathbb{X}^T(\mathbb{X}\mathbb{X}^T + \lambda I)^{-1}\mathbb{X}Y$

   (b) $(\mathbb{X}^T\mathbb{X} + \lambda I)^{-1}\mathbb{X}^TY$

   [(c)] $\mathbb{X}(\mathbb{X}^T\mathbb{X} + \lambda I)^{-1}\mathbb{X}^TY$

   (d) $(\mathbb{X}\mathbb{X}^T + \lambda I)^{-1}\mathbb{X}Y$

   1.2. The main purpose of the lasso is to

   (a) estimate coefficients in a linear model

   (b) discard variables that are not predictive of the response

   [(c)] find the best sparse linear regression model for a given dataset

   (d) perform penalized linear regression

   1.3. Suppose that we have a kernel regression technique in one dimension with bandwidth parameter h for which the squared bias scales as $O(h^2)$ and the variance scales as $O\left(\frac{1}{nh}\right)$ as $h \to 0$ with $nh \to \infty$, for a sample of size n, under certain assumptions. What is the fastest rate at which the risk (expected squared error) can decrease with sample size for this technique?

   (a) $O(n^{-1/3})$

   [(b)] $O(n^{-2/3})$

   (c) $O(n^{-1/2})$

   (d) $O(n^{-2/5})$

   1.4. Variational methods are used to make

   (a) numerical approximations to a Gibbs sampling algorithm

   (b) random approximations to a frequentist model

   [(c)] deterministic approximations to a Bayesian posterior distribution

   (d) the ELBO as small as possible

1.5. Consider the model $p(z_1, z_2, z_3) \propto \exp(z_1z_2 + z_2z_3 + z_1z_3)$ where each $z_i \in \{0, 1\}$ is a binary random variable. You are running the Gibbs sampling algorithm for this model. Suppose that you currently have $z_1 = 1$, $z_2 = 1$, and $z_3 = 1$. Randomly sampling z_1 while holding $z_2 = 1$ and $z_3 = 1$ fixed, what is the probability that z_1 is changed to $z_1 = 0$?

(a) $1/(1 + e^2)$

(b) $1/(1 + e^{-2})$

(c) $1/3$

(d) $1/2$

To explain some of these answers:

1.1 The key is to check the dimensions of the matrices; (c) is the only one that makes sense since $\widehat{Y}$ has to be an n-vector.

1.2 The lasso looks for a good sparse linear model. It is not (b) because this would allow for non-linear effects.

1.3 Balancing squared bias and variance we have $h^2 \approx \frac{1}{nh}$ which implies $h \approx \frac{1}{n^{1/3}}$. Thus the risk is $O(h^2) = O(n^{-2/3})$.

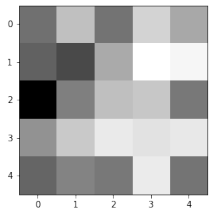
1.5 The probability of $(z_1, z_2, z_2) = (1, 1, 1)$ is proportional to e^3. The probability of $(z_1, z_2, z_2) = (0, 1, 1)$ is proportional to e^1. So the conditional probability of the latter is $e/(e + e^3) = 1/(1 + e^2)$.

2. ***Convolutional neural networks*** (10 points)

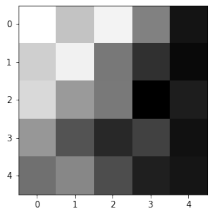
Convolutional neural networks (CNNs) work by learning a set of kernel functions or "filters" that are swept across an image to create a "feature map." Some of the filters can be difficult to interpret; others are more interpretable.

(a) The figure below shows six 5×5 filters learned by training a CNN.

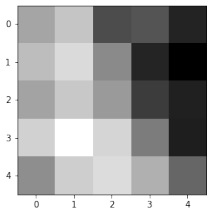
Filter A



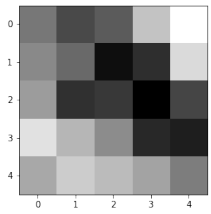
Filter B



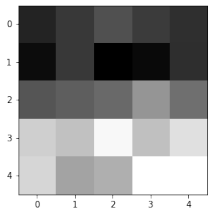
Filter C



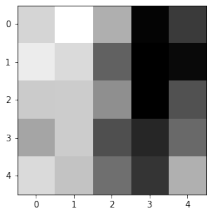
Filter D



Filter E

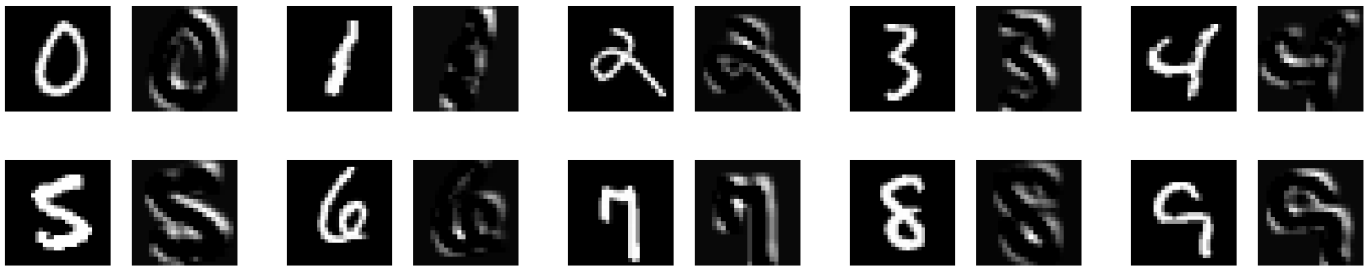


Filter F



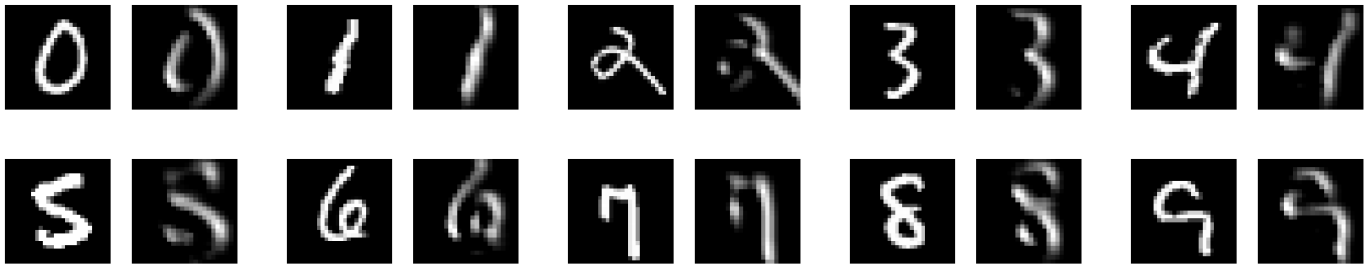
On the following two pages, a set of images is shown, together with the feature map generated by applying one of these filters. Write the name of the filter that generated each group of maps. Each filter is applied to only one group of images.

Feature map 1. Filter used: __D_____



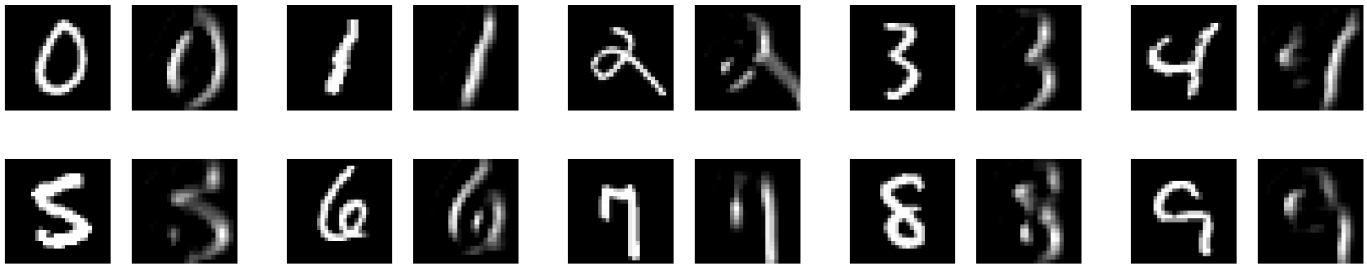
This causes double lines, mainly on a diagonal.

Feature map 2. Filter used: __C_____



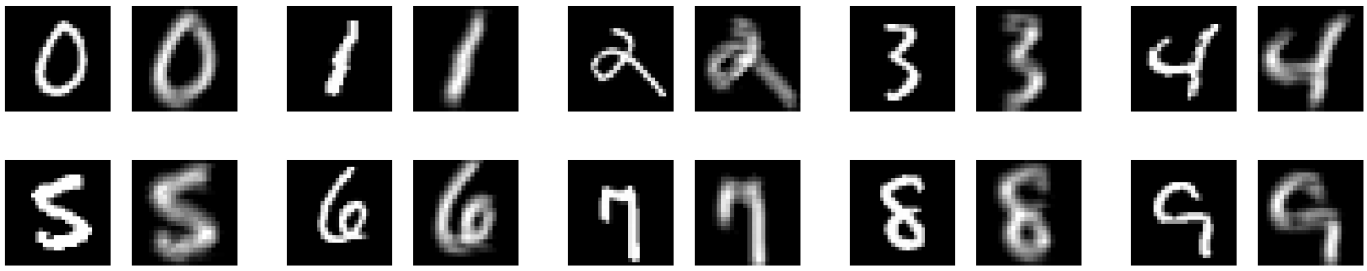
This is a diagonal filter; look at the tail of the 2.

Feature map 3. Filter used: __F_____



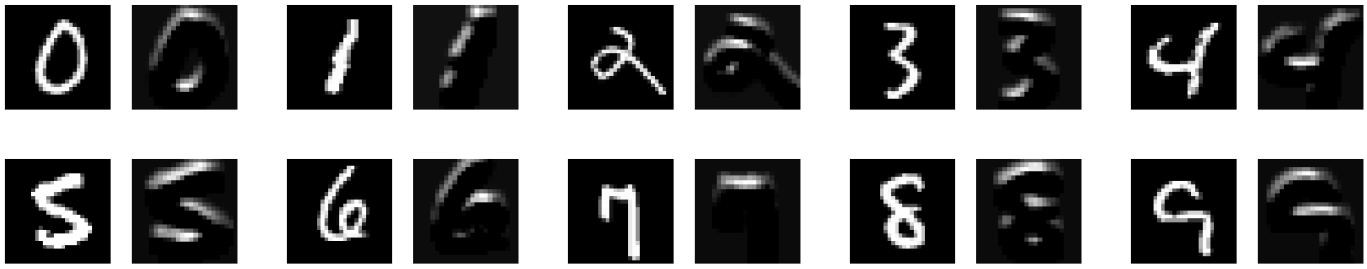
This is mainly a vertical filter.

Feature map 4. Filter used: A



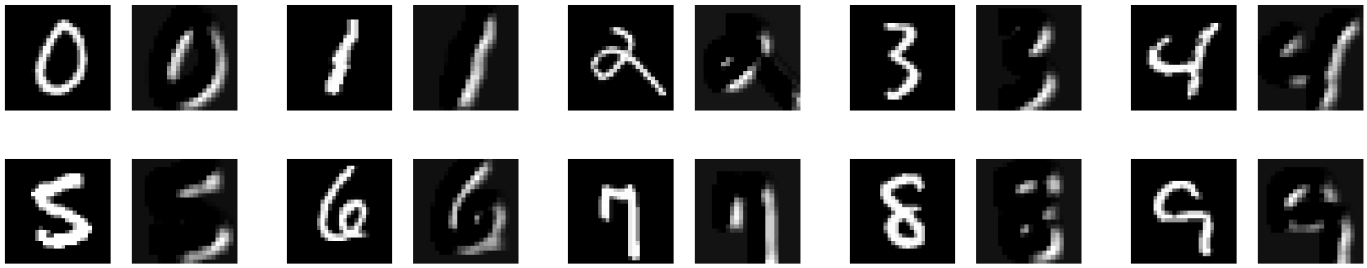
This acts a blurring filter, like a Gaussian.

Feature map 5. Filter used: E



This is a horizontal filter.

Feature map 6. Filter used: B

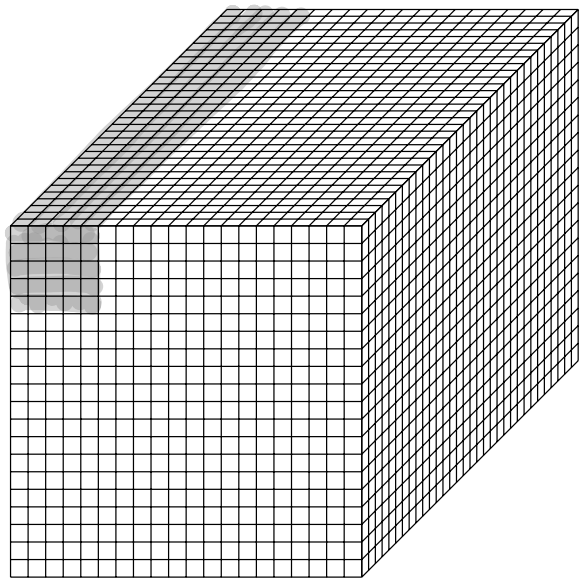


This is a diagonal filter, from lower left to upper right.

(b) Suppose we were to have a convolutional layer constructed as

```
model.add(layers.Conv2D(32, (5, 5), input_shape=(20, 20, 32)))
```

that applies 32 kernels to an input tensor of shape $(20, 20, 32)$. In the diagram below of an input tensor of this shape, shade in the voxels that would be used to compute the (0,0) entry of the feature map for one of the kernels.



(c) What is the shape of the output tensor for this layer? `(16, 16, 32)`

3. ***Short answer*** (10 points)

The following two subproblems ask you to explain the important concepts associated with two topics that have been central to the first part of the course.

(a) ***Mercer kernels***. Describe the role of Mercer kernels in machine learning, answering each of the following questions: (1) How are Mercer kernels defined? (2) What is the RKHS associated with a Mercer kernel? (3) What are two examples of Mercer kernels? (4) What are two ways that Mercer kernels are used?

(1) Mercer kernels are defined by the property that $K(x, x') \geq 0$ is symmetric and the matrix $\mathbb{K} = [K(x_i, x_j)]$ is positive semidefinite for any $x_1, \ldots x_n$.

(2) The reproducing kernel Hilbert space (RKHS) associated with the Mercer kernel K is spanned by functions of the form $\sum_i \alpha_i K(\cdot, x_i)$ with inner product $\langle f, g \rangle = \sum_{ij} \alpha_i \beta_j K(x_i, z_j)$ if $f(\cdot) = \sum_i \alpha_i K(x_i, \cdot)$ and $g(\cdot) = \sum_j \beta_j K(z_j, \cdot)$.

(3) The Gaussian kernel $K(x, x') = \exp\left(-\frac{1}{2}\|x - x'\|^2\right)$ is a Mercer kernel as is $K(x, x') = x^T x'$ or $K(x, x') = \phi(x)^T \phi(x')$ for any feature map $x \to \phi(x)$

(4) Gaussian processes and Mercer kernel regression are two uses we've discussed extensively.

(b) ***Gaussian processes***. Describe the role of Gaussian processes in machine learning, answering each of the following questions: (1) What is the definition of a Gaussian process? (2) What is one way that Gaussian processes are used? (3) Describe how Gaussian processes form a nonparametric Bayesian method by describing the Gaussian process prior, the data likelihood, and how the posterior distribution is calculated.

(1) A Gaussian process is a stochastic process m with the property that for any set of points $x_1, \ldots, x_n$, the vector $(m(x_1), \ldots, m(x_n))$ is Gaussian with distrubtion $N(\mu(x), K(x))$ where $K(x) = \mathbb{K}$ is a Mercer kernel.

(2) Gaussian processes are used for Bayesian regression.

(3) The three ingredients for Bayesian regression are:
   a) The prior is m is a Gaussian process with (typically) mean zero and covariance given by a kernel K.
   b) The likelihood is typically of the form $Y_i \mid m, x_i \sim N(m(x_i), \sigma^2)$
   c) Posterior inference is done using the formulas for conditional Gaussians given on the first page of this exam

4. ***Implementation and Dirichlet processes*** (10 points)

Consider the following partial code for sampling data $X_1, \ldots, X_{\text{size}}$ from the marginal of the posterior of a Dirichlet process with prior $DP(\alpha, F_0)$ with $F_0 = N(\mu_0, \tau_0^2)$.

```
import numpy as np
from scipy.stats import norm

def sample_X_from_posterior_DP(X=[], size=10, alpha=1, mu0=0, tau0=1):
    X_sample = []
    for i in np.arange(size):
        m = i + len(X)
        z = np.random.choice([0,1], p=[alpha/(alpha+m), m/(alpha+m)])
        # your code begins here
        ...
        # your code ends here
        X_sample.append(x)

    return(X_sample)
```

(a) Complete the code, to compute a list of samples `X_sample` of length `(size)` drawn from the posterior marginal. Write your code below.

Hint: See the descriptions of the functions `np.random.choice` and `norm.rvs` from the cover page.

```
if z==0:
   x = norm.rvs(loc=mu0, scale=tau0)
else:
   x = np.random.choice(X + X_sample)
```

(b) We ran following code six times, with different values of `alpha`, `size`, and `n`

```
X = norm.rvs(size=n, loc=3)
sample = sample_X_from_posterior_DP(X=X, alpha=alpha, size=size)
plot_empirical_cdf(sample)
```

Match each plot with the values of `size`, `alpha`, and `n` that were *most likely* used.

(a) `size=10, alpha=1, n=100`
(b) `size=10, alpha=100, n=100`
(c) `size=10, alpha=100, n=1`
(d) `size=50, alpha=1, n=1`
(e) `size=50, alpha=1, n=100`
(f) `size=50, alpha=100, n=100`

Here is one line of reasoning for the solution on the following page, referring to the plots by (row, col) as $(1,1)$, $(1,2)$, . . ., $(3,1)$, $(3,2)$.

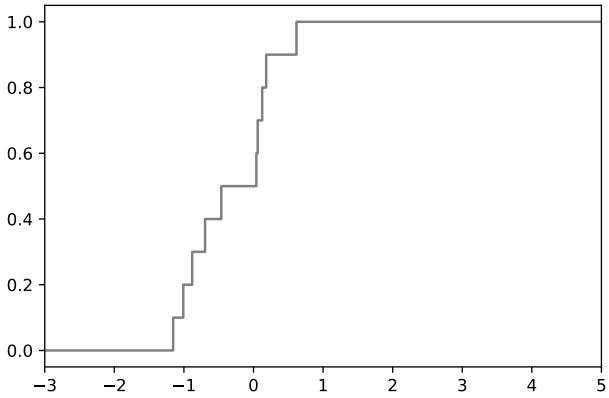
- Plots (1,2) and (2,2) have about 50 steps, so they have `size`=50 and large α. (1,2) is shifted far to the right, so that's (e). (1,2) has a mix of $N(0,1)$ and $N(3,1)$ so that's (b).

- Plots (1,1) and (2,1) have 10 steps. So those have `size`=10 and large α. Plot (1,1) is all $N(0,1)$ so that's (c), and $(2,1)$ is a mix, so that's (b).

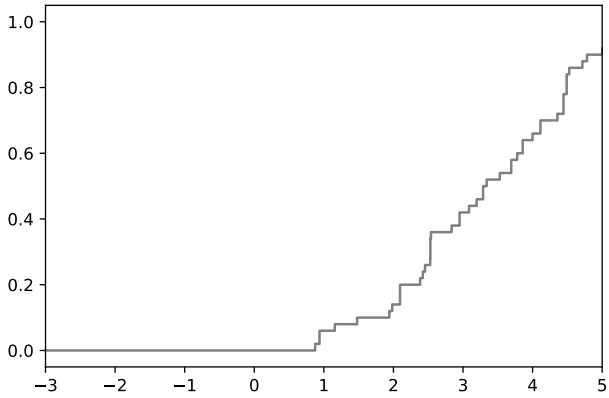
- This leaves us with the last row. We have to choose between (d) and (a). Since plot (3,1) is pulled much farther to the right, that's (a).

Mark a, b, c, d, e, or f in the box above each plot, according to the key on the previous page.

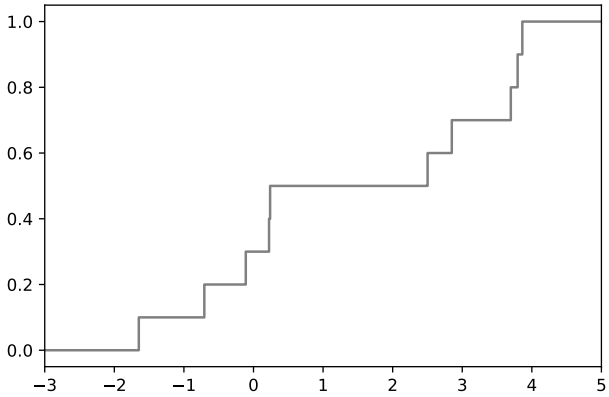
(c)



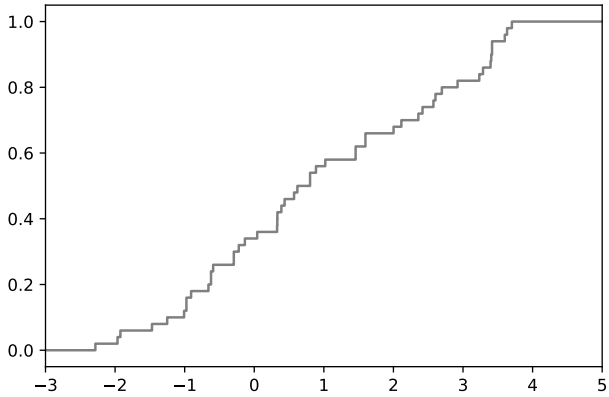
(e)



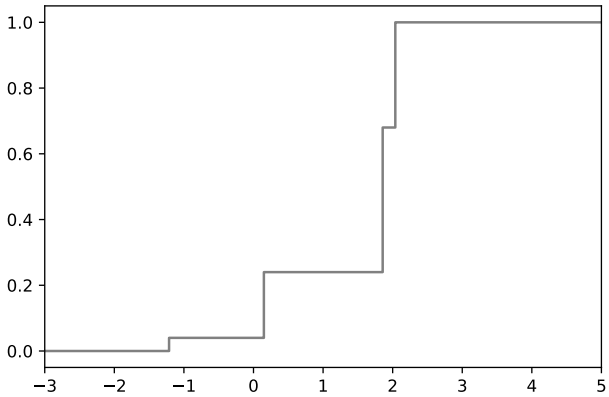
(b)



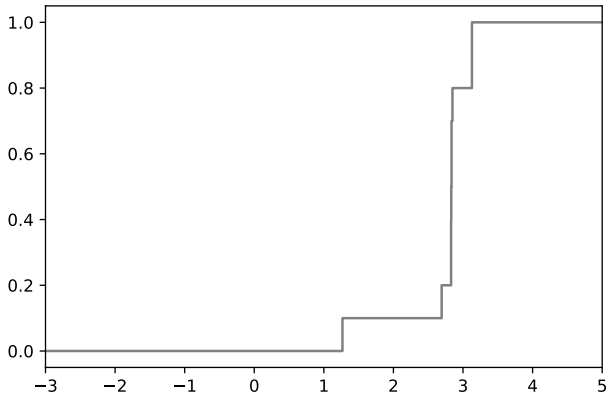
(f)



(d)



(a)



5. ***Extra credit: Sparse Mercer kernel regression*** (2 points)

Mercer kernel regression estimates a weight α_i for each data point X_i. If we were to look for a sparse estimate, zeroing out the weight for many data points, the estimator could combine the ℓ_1 and RKHS penalties:

$$\widehat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \frac{1}{2n}\|Y - \mathbb{K}\alpha\|^2 + \rho\lambda\|\alpha\|_1 + (1-\rho)\frac{\lambda}{2}\alpha^T\mathbb{K}\alpha \tag{1}$$

where $0 \leq \rho \leq 1$ controls how much emphasis is put on the ℓ_1 norm $\|\alpha\|_1 = \sum_{j=1}^{p}\|\alpha_j|$ relative to the RKHS norm; for $\rho = 1$ we get the lasso, for $\rho = 0$ we get Mercer kernel regression, and for $0 < \rho < 1$ we use a combination of the two penalties.

(a) Consider the special case

$$\widehat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \frac{1}{2}(Y-\alpha)^2 + \rho\lambda|\alpha| + (1-\rho)\frac{\lambda}{2}K\alpha^2$$

where Y is a single (random) number, and α is a single parameter, and K is a kernel value. Give a closed-form solution for $\widehat{\alpha}$.

$$\widehat{\alpha} = \frac{\text{Soft}_{\rho\lambda}(Y)}{1 + (1-\rho)\lambda K}$$

(b) Give an algorithm for solving the general optimization (1).

The algorithm is coordinate descent over each α_j, holding the others fixed.
This involves an elastic net type of optimization

$$\min_{\alpha_j} \frac{1}{2n} \sum_i (r_{i\setminus j} - K(x_i, x_j)\alpha_j)^2 + \rho\lambda|\alpha_j| + (1-\rho)\frac{\lambda}{2}K(x_j, x_j)\alpha_j^2 + w_j\alpha_j$$

where $w_j = (1-\rho)\lambda \sum_{l\neq j} K(x_l, x_j)\alpha_l$ and $r_{i\setminus j} = Y_i - \sum_{l\neq j} K(x_i, x_l)\alpha_l$ is the residual. The solution to this has a closed form, similar to the solution for part (a) (and the practice midterm):

$$\alpha_j = \frac{\text{Soft}_{\rho\lambda}\left(\frac{1}{n}\sum_i K(x_i, x_j) r_{i\setminus j} - w_j\right)}{\frac{1}{n}\sum_i K(x_i, x_j)^2 + (1-\rho)\lambda K(x_j, x_j)}$$