Name: ______________________________ NetID: ___________

S&DS 365 / 665

Intermediate Machine Learning

Midterm Exam (Sample Solution)

October 16, 2023

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. No calculators—one problem requires some multiplication and addition that you can do on paper.

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

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

- 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 `numpy.linalg.inv(A)` computes the inverse of a matrix `A`

- The identity function is $\text{id}(u) = u$. The rectified linear unit activation function is defined by $\texttt{relu}(u) = \max(u, 0)$ and the hyperbolic tangent activation function is $\tanh(u) = \dfrac{e^u - e^{-u}}{e^u + e^{-u}} = 2\sigma(2u) - 1$ where $\sigma(u) = \dfrac{1}{1 + e^{-u}}$ is the sigmoid.

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

For each of the following questions, circle the *single best* answer, unless the question allows for multiple answers.

1.1. Consider the toy lasso problem

$$\widehat{\beta} = \arg\min_{\beta} \left\{ (Y - 2\beta)^2 + |\beta| \right\}$$

where Y is a random variable and β is a scalar. If $Y = -1$ the solution is

(a) $\widehat{\beta} = 0$

(b) $\widehat{\beta} = -\frac{1}{2}$

[✓ (c)] $\widehat{\beta} = -\frac{3}{8}$

(d) $\widehat{\beta} = \frac{1}{4}$

1.2. 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 $O\left(\frac{1}{nh^2}\right)$ as $h \to 0$ with $nh^2 \to \infty$, for a sample of size n, under certain assumptions. What is the fastest rate at which the risk (expected squared error) will decrease with sample size for this technique?

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

(b) $O(n^{-1/4})$

[✓ (c)] $O(n^{-1/2})$

(d) $O(n^{-1})$

1.3. When neural networks are used for classification, an activation function is used for each layer of hidden neurons, and the last layer is just linear. Which of the following activation functions lead to piecewise linear decision boundaries? Circle all that apply.

(a) tanh (hyperbolic tangent)

[✓ (b)] relu (rectified linear unit)

[✓ (c)] id (identity function)

(d) sigmoid (sigmoid function)

(e) None of the above

1.4. Which of the following are Mercer kernels over inputs $x \in \mathbb{R}^d$, meaning that $\mathbb{K} = [K(x_i, x_j)]$ is a positive semi-definite matrix for any collection of data points $x_1, \ldots, x_n$? Circle all that apply.

[✓ (a)] $K(x_1, x_2) = \phi(x_1)^T \phi(x_2)$ for a linear feature map $\phi(x) = Wx$

[✓ (b)] $K(x_1, x_2) = \phi(x_1)^T \phi(x_2)$ for a possibly nonlinear feature map $\phi : \mathbb{R}^d \to \mathbb{R}^k$

[✓ (c)] $K(x_1, x_2) = c \exp\left(-\|x_1 - x_2\|^2\right)$ with $c > 0$

[✓ (d)] $K(x_1, x_2) = \frac{1}{2}K_1(x_1, x_2) + \frac{1}{2}K_2(x_1, x_2)$ where K_1 and K_2 are Mercer kernels

(e) None of the above

1.5. Consider the following derivation of a bound on $p(x)$ for a Bayesian model, leading to the ELBO (evidence lower bound):

$$\begin{aligned}
\log\ p(x) &= \int q(\theta) \log p(x)\, d\theta && (1)\\
&= \int q(\theta) \log\left(\frac{p(x,\theta)}{q(\theta)}\right) d\theta + \int q(\theta) \log\left(\frac{q(\theta)}{p(\theta \,|\, x)}\right) d\theta && (2)\\
&\geq \int q(\theta) \log\left(\frac{p(x,\theta)}{q(\theta)}\right) d\theta && (3)\\
&= H(q) + \mathbb{E}_q\big(\log p(x,\theta)\big) && (4)
\end{aligned}$$

Which (if any) of the above steps in the derivation is *not* correct?

(a) (1) ⇒ (2)

(b) (2) ⇒ (3)

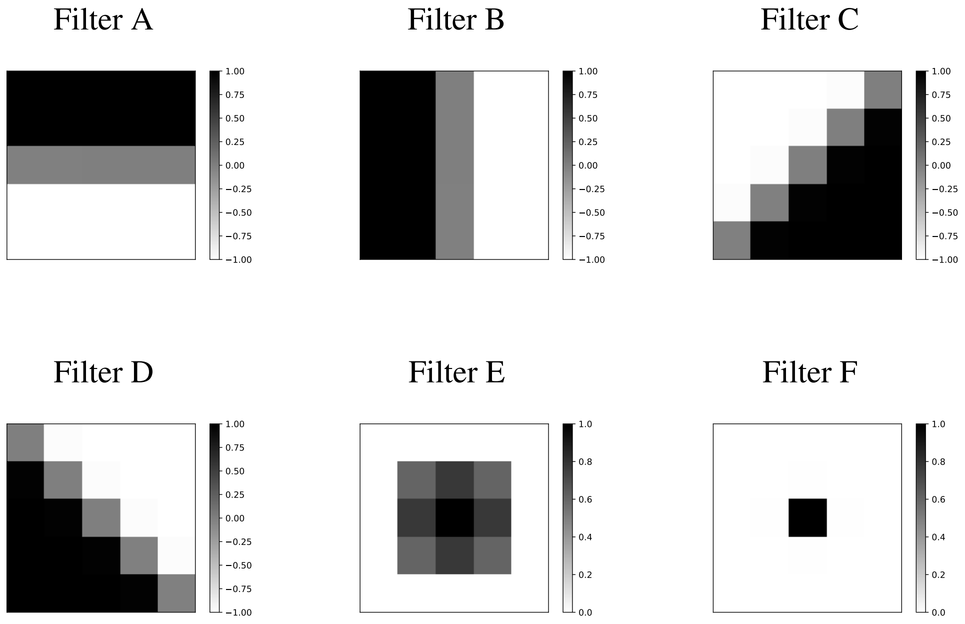
(c) (3) ⇒ (4)

[✓ (d)] They are all correct

2. ***(Not) A convoluted question*** (16 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.

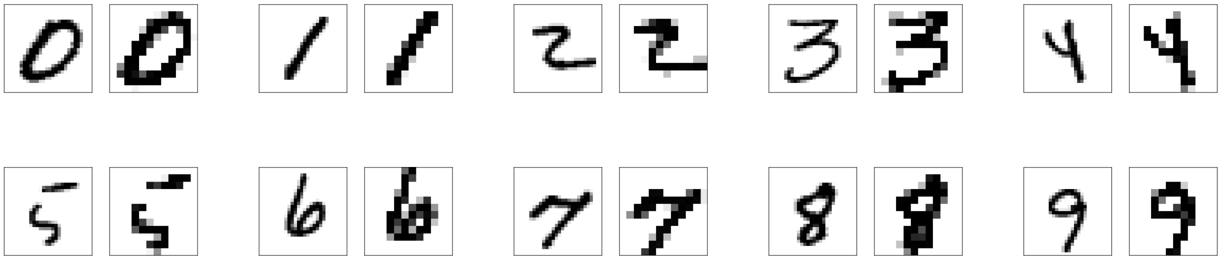
(a) The figure below shows six 5×5 filters:



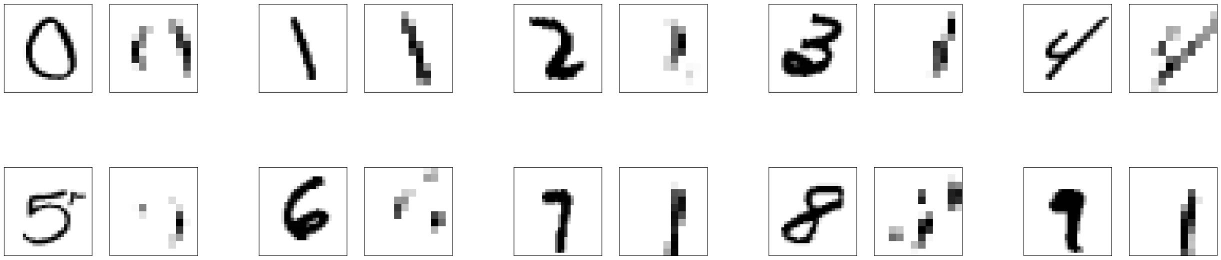
On the following two pages, a set of images is shown, together with the feature map generated by applying one of these filters. Each filter is applied using *either* the `relu` activation function or the `tanh` activation function, each with a particular value of an intercept b. Then, the max pooling is applied over 2×2 regions. The resulting feature map is displayed.

On the following pages, write the name of the filter and the activation function that generated each group of maps. *Each filter is applied to only one group of images.*

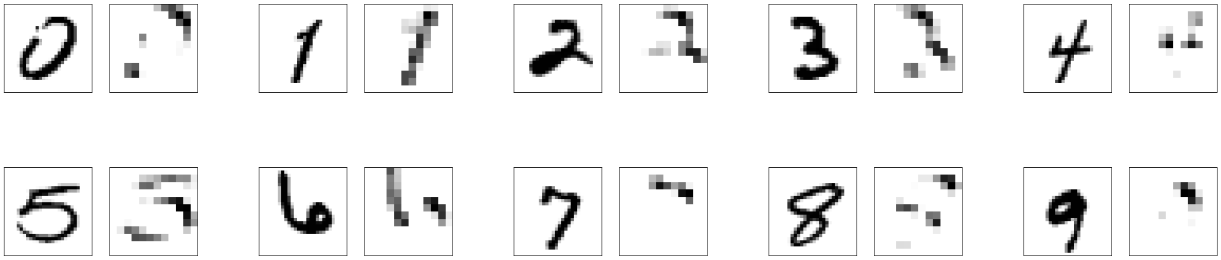
Feature map 1: Filter: F Activation function: relu



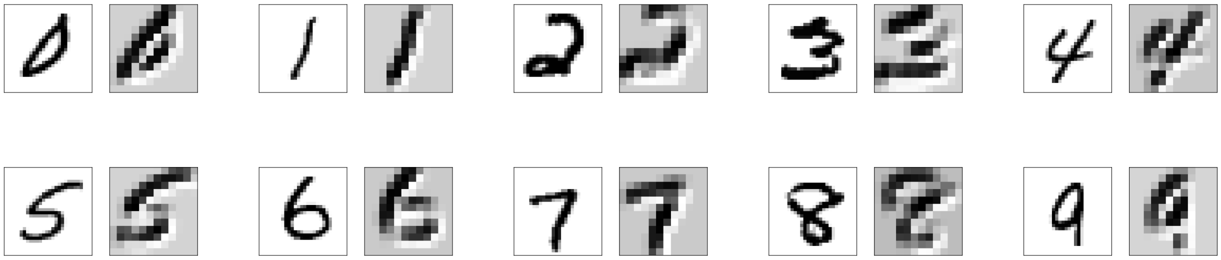
Feature map 2: Filter: B Activation function: relu



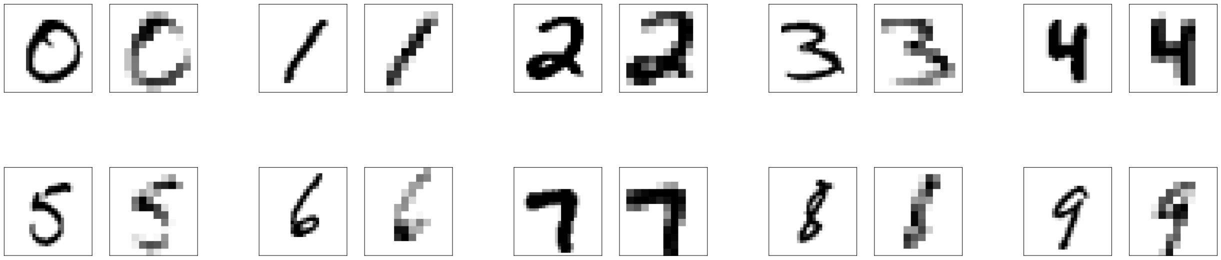
Feature map 3: Filter: D Activation function: relu



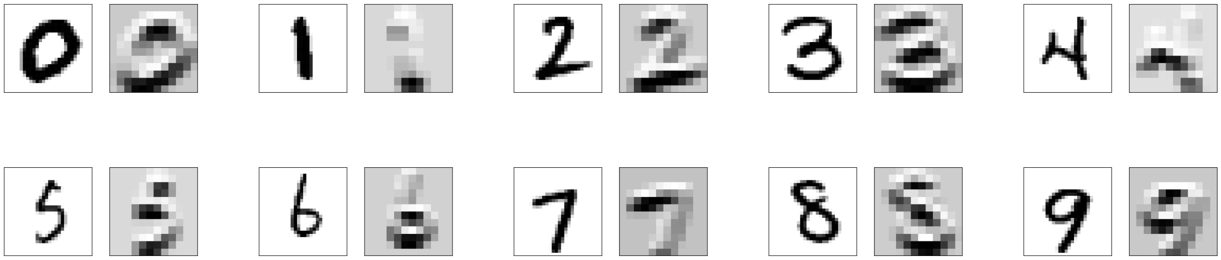
Feature map 4: Filter: C Activation function: tanh



Feature map 5: Filter: E Activation function: relu



Feature map 6: Filter: A Activation function: tanh



(b) The following TensorFlow code constructs a CNN to classify 28×28 grayscale images, such those in the MNIST dataset, using a single convolutional layer with max pooling and dropout, followed by a dense layer.

```
from tensorflow.keras import layers, models
model = models.Sequential()

model.add(layers.Conv2D(10, (3, 3), input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((4, 4)))
model.add(layers.Dropout(rate=.5))
model.add(layers.Flatten())
model.add(layers.Dense(10))
```

Indicate the shape of the output tensor for each layer, together with the number of trainable parameters, by filling in the two missing fields for each of the rows below. For partial credit if an answer is wrong, you may show your work below the table. No calculators.

Either of the following solutions:

```
__________________________________________________________________
 Layer (type)                 Output Shape              Param #
==================================================================
 conv2d (Conv2D)             (None, 26, 26, 10)       100

 max_pooling2d (MaxPooling)  (None, 6, 6, 10)         0

 dropout (Dropout)           (None, 6, 6, 10)         0

 flatten (Flatten)           (None, 360)              0

 dense (Dense)               (None, 10)               3610

==================================================================
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)            (None, 26, 26, 10)       100

 max_pooling2d (MaxPooling) (None, 7, 7, 10)         0

 dropout (Dropout)          (None, 7, 7, 10)         0

 flatten (Flatten)          (None, 490)              0

 dense (Dense)              (None, 10)               4910

=================================================================
```

3. ***Short but sweet*** (12 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.

Answer both of the following two questions.

(a) ***RKHS***. Describe the role of the Reproducing Kernel Hilbert Space (RKHS) in machine learning when using Mercer kernels, answering each of the following questions: (1) Describe how a RKHS is defined, by giving examples of functions in the RKHS. (2) How is the inner product in an RKHS defined? (3) What are two different machine learning models that use an RKHS?

(1). A RKHS is a space spanned by the functions of the form $f(x) = \sum_i \alpha_i K(x_i, x)$ where $K(\cdot, \cdot) : X \times X \to \mathbb{R}$ is Mercer kernel.

(2). The inner product in an RKHS is defined as:
$f(x) = \sum \alpha_i K(x_i, x), g(x) = \sum \beta_j K(y_j, x)$
$\langle f, g \rangle = \sum \sum \alpha_i \beta_j K(x_i, y_j) = \alpha^T K \beta$

(3). Mercer Kernel Regression, Gaussian Process

(b) ***Double descent***. Describe the "double descent" phenomenon in machine learning, answering each of the following questions. (1) What does double descent refer to, as a function of $\gamma = p/n$ for a linear model? (2) How is a random features neural network defined, and why is it equivalent to a linear model? (3) What is the significance of double descent for understanding the behavior of deep neural networks?

(1). Double descent is a phenomenon on the risk profile of machine learning models as a function of model complexity $\gamma = p/n$.

When $\gamma < 1$, the curve of risk is traditional U-shaped and there is a bias-variance trade-off. As γ keeps increasing, it will reach the interpolation threshold $\gamma = 1$. When $\gamma > 1$, the risk descends again and the variance also goes down.

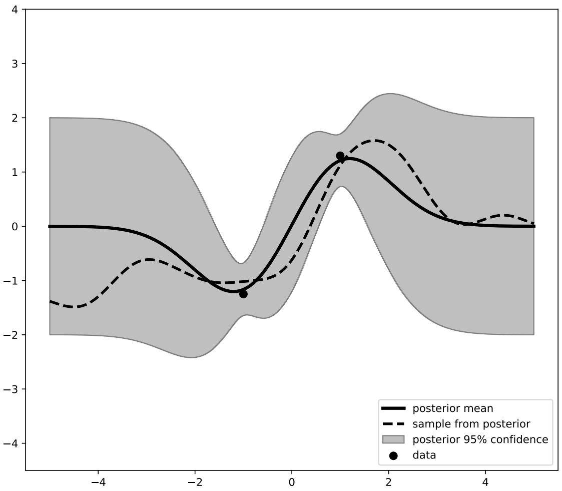
(2). The input layer is a vetor x, and the random features layer is $\phi(wx+b)$. The output layer is then $y = \beta^T \phi(wx + b)$, which is equivalent to a linear model.

(3). Double descent offers insights into why deep learning models improve generalization and overcome the traditional U-shaped bias-variance trade-off.

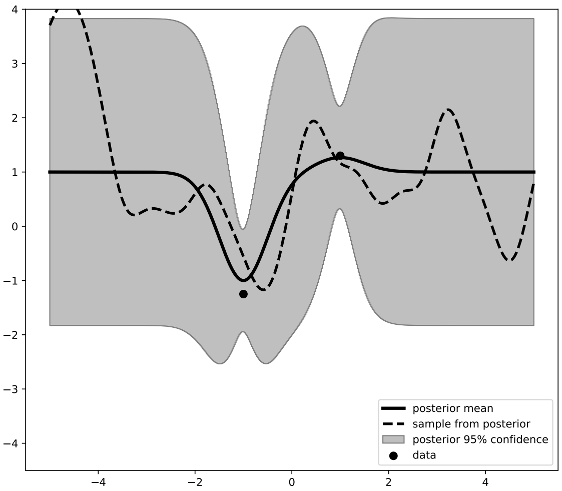
4. ***Due process*** (16 points)

The following four plots show posterior inference for a Gaussian process with different settings of the (Gaussian) kernel bandwidth h, a constant factor c scaling the kernel, the prior mean μ, and the measurement noise level σ. Please see the code below for how exactly these parameters were used.

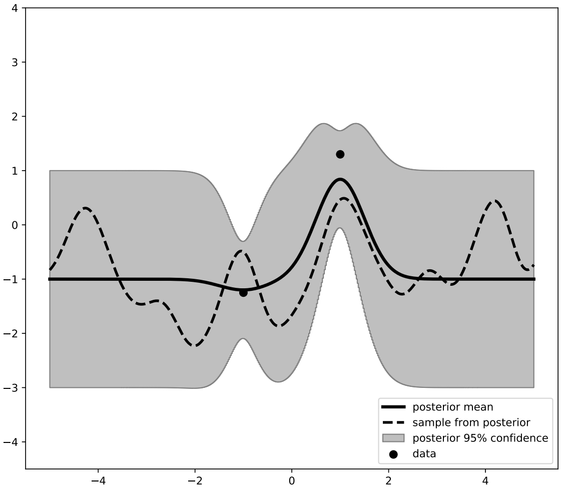
Plot A



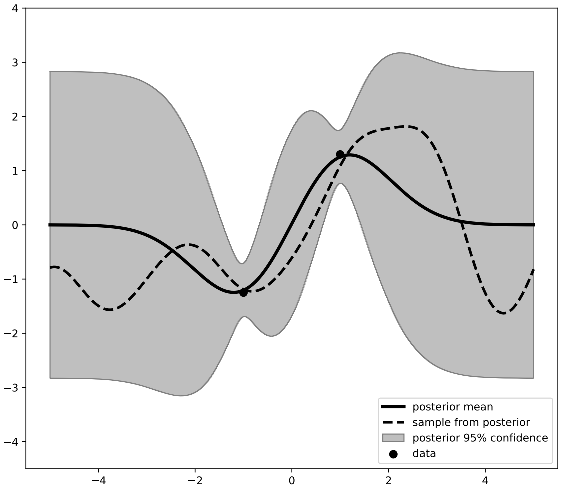
Plot B



Plot C



Plot D



Note: The posterior mean, sample, and covariance are computed for the posterior distribution $\mathbb{P}(m(x)\,|\,y_1, y_2)$ *of the regression function. In the demo presented in class, the posterior shown was for* $\mathbb{P}(Y = m(x) + \sigma\epsilon\,|\,y_1, y_2)$. *This is why the posterior samples above are smooth rather than "wiggly" as in the class demo and Olympics examples.*

(a) Each plot was generated with a choice of h, c, μ, σ, chosen from a few possibilities:

$$h \in \left\{\tfrac{1}{2}, 1\right\} \quad c \in \{1, 2\} \quad \mu \in \{-1, 0, 1\} \quad \sigma \in \left\{\tfrac{1}{4}, \tfrac{1}{2}\right\}$$

Indicate what the parameters were for each plot:

Plot A: $h = 1 \quad c = 1 \quad \mu = 0 \quad \sigma = \frac{1}{4}$

Plot B: $h = \frac{1}{2} \quad c = 2 \quad \mu = 1 \quad \sigma = \frac{1}{2}$

Plot C: $h = \frac{1}{2} \quad c = 1 \quad \mu = -1 \quad \sigma = \frac{1}{2}$

Plot D: $h = 1 \quad c = 2 \quad \mu = 0 \quad \sigma = \frac{1}{4}$

The following code partially implements a Gaussian process to generate these examples. Your job is to complete the implementation by providing four additional lines of code.

```
import numpy as np


bandwidth = 0.5
const = 2
prior_mean = 1
sigma = 0.25

def prior_mean_fun(x):
    return prior_mean * np.ones(len(x))

def kernel_fun(x, z):
    K = np.zeros((len(x),len(z)))
    for j in np.arange(len(z)):
        K[:,j] = const * gaussian_kernel(x-z[j], bandwidth)
    return K

x_train = np.array([-1, 1])
x_test = np.linspace(-5, 5, 500)
n = len(x_train)
m = len(x_test)
y_train = np.array([-1, 1]) + sigma*np.random.normal(size=n)

K_train = kernel_fun(x_train, x_train)
K_train_test = kernel_fun(x_train, x_test)
K_test_test = kernel_fun(x_test, x_test)

prior_mean_train = prior_mean_fun(x_train)
prior_mean_test = prior_mean_fun(x_test)


###### Complete the following four lines.
posterior_mean = ...
posterior_cov = ...
posterior_upper_band = ...
posterior_lower_band = ...
#######
```

(b) Complete the implementation, by writing the four missing lines below. To make your code more readable, you may want to define extra variables.

```
K_inv = np.linalg.inv(K_train + sigma**2 * np.eye(n))
posterior_mean = prior_mean_test +
       K_train_test.T @ K_inv @ (y_train - prior_mean_train)
posterior_cov = K_test_test - K_train_test.T @ K_inv @ K_train_test
posterior_upper_band = posterior_mean + 2*np.sqrt(np.diag(posterior_cov))
posterior_lower_band = posterior_mean - 2*np.sqrt(np.diag(posterior_cov))
```

5. ***Class act*** (10 points)

In class and on the assignments, we used Gaussian processes and Mercer kernels mainly for regression. But they can also be used for classification.

For binary classification, a natural approach to using a Mercer kernel K is based on the likelihood model

$$\mathbb{P}(y = 1 \,|\, m, x) = \frac{e^{m(x)}}{1 + e^{m(x)}} = \text{sigmoid}(m(x)).$$

The regularized log-loss over a training set $\{(x_i, y_i)\}$ with $y_i \in \{0, 1\}$ is then

$$\sum_{i=1}^{n} \left\{ \log\bigl(1 + e^{m(x_i)}\bigr) - y_i m(x_i) \right\} + \lambda \|m\|_K^2$$

where $\|\cdot\|_K$ is the norm in the RKHS for the kernel K.

(a) Give an algorithm for estimating a function m that minimizes this loss function. For full credit, give the algorithm in an explicit form that could be implemented.

Hint: Recall (or derive) the SGD algorithm for standard parametric logistic regression.

The model is

$$p(x) = \mathbb{P}(y = 1 \,|\, m, x) = \text{sigmoid}\left(\sum_{i=1}^{n} \alpha_i K(x, x_i) \right)$$

which can be expressed as a parametric logistic regression model. The SGD update is for data point (x_i, y_i) is to increment all the parameters α_j as

$$\alpha_j \leftarrow \alpha_j + \eta(y_i - p(x_i) K(x_i, x_j) - 2\eta\lambda \sum_{i=1}^{n} K(x_i, x_j)\alpha_j$$

which follows from the parametric logistic regression update.

(b) When taking a nonparametric Bayesian approach, we use a Gaussian process prior $m \sim GP(0, K)$ with mean zero and covariance given by the kernel K. As for regression, the Bayesian approach leads to a posterior confidence interval on $m(x)$ for an arbitrary point x, given data $\{(x_i, y_i)\}$ and assuming the same likelihood as in part (a). Explain why this is more challenging to compute in the case of classification, compared with regression. Describe an approach to approximating the posterior.

Because of the binomial (Bernoulli) likelihood, the posterior distribution over m is not Gaussian, and does not have a closed form — unlike for Gaussian processes. So, we need to approximate the posterior, using either variational inference or Gibbs sampling.

Extra work space