COMP 5212 Machine Learning (2024 Fall)
Homework 1
Hand out: Sept 12, 2024
Due: Oct 2, 2024, 11:59 PM
Total Points: 86
Name: Tran Chun Yui
SID: 20963715

Name: Tran Chun Yui
SID: 20963715
Your solution should contain below information at the top of its first page.

1. Your name
2. Your student id number

Some Notes:

- Homeworks will not be easy, please start early.
- For this homework, please submit a single PDF file to Canvas. Make sure to prepare your solution to each problem on a separate page.
- The total points of this homework is 86, and a score of 86 already gives you full grades on this homework. There are possibly bonus questions, you can optionally work on them if you are interested and have time.
- You can choose either using LaTeX by inserting your solutions to the problem pdf, or manually write your solutions on clean white papers and scan it as a pdf file – in the case of handwriting, please write clearly and briefly, we are not responsible to extract information from unclear handwriting. **We highly recommend you use LaTeX for the sake of any misunderstandings about the handwriting.** If your submission is a scan of a handwritten solution, make sure that it is of high enough resolution to be easily read. At least 300dpi and possibly denser.
- We encourage students to work in groups for homeworks, but the students need to write down the homework solutions or the code independently. In case that you work with others on the homework, please write down the names of people with whom you've discussed the homework. You are not allowed to copy, refer to, or look at the exact solutions from previous years, online, or other resources.
- **Late Policy:** 3 free late days in total across the semester, for additional late days, 20% penalization applied for each day late. **No assignment will be accepted more than 3 days late**.
- Please refer to the Course Logistics page for more details on the honor code and logisitcs. We have zero tolerance — in the case of honor code violation for a single time, you will fail this course directly.

Linear Regression (30 pts)

Let $D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\}$ where $\mathbf{x_i} \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ be the training data that you are given. As you have to predict a continuous variable, one of the simplest possible models is linear regression, i.e. to predict y as $\mathbf{w^T x}$ for some parameter vector $\mathbf{w} \in \mathbb{R}^d$.[1] We thus suggest minimizing the following loss

$$\operatorname*{argmin}_{\mathbf{w}} \hat{R}(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} \sum_{i=1}^{n} (y_i - \mathbf{w^T x_i})^2. \tag{1}$$

Let us introduce the $n \times d$ matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ with the $\mathbf{x_i}$ as rows, and the vector $\mathbf{y} \in \mathbb{R}^n$ consisting of the scalars y_i. Then, (1) can be equivalently re-written as

$$\operatorname*{argmin}_{\mathbf{w}} \|\mathbf{Xw} - \mathbf{y}\|^2. \tag{2}$$

We refer to any $\mathbf{w}^*$ that attains the above minimum as a solution to the problem.

1. [**4 points**] Show that if $\mathbf{X^T X}$ is invertible, then there is a unique $\mathbf{w}^*$ that can be computed as $\mathbf{w}^* = (\mathbf{X^T X})^{\mathbf{-1}} \mathbf{X^T y}$.

2. [**4 points**] Please give at least two cases under which $\mathbf{X^T X}$ is not invertible, explain your answers. How many solutions does Eq. 2 have in these cases?

3. [**4 points**] Consider the case $n \geq d$. Under what assumptions on $\mathbf{X}$ does Eq. 2 admit a unique solution $\mathbf{w}^*$? Give an example with $n = 3$ and $d = 2$ where these assumptions do not hold.

The ridge regression optimization problem with parameter $\lambda > 0$ is given by

$$\operatorname*{argmin}_{\mathbf{w}} \hat{R}_{\text{ridge}}(\mathbf{w}) = \operatorname*{argmin}_{\mathbf{w}} \left[\sum_{i=1}^{n} (y_i - w^T \mathbf{x_i})^2 + \lambda \mathbf{w}^T \mathbf{w} \right]. \tag{3}$$

4. [**4 points**] Show that $\hat{R}_{\text{ridge}}(w)$ is convex with regards to $\mathbf{w}$. You can use the fact that a twice differentiable function is convex if and only if its Hessian $\mathbf{H} \in \mathbb{R}^{d \times d}$ satisfies $\mathbf{w}^T \mathbf{H} \mathbf{w} \geq 0$ for all $\mathbf{w} \in \mathbb{R}^d$ (is positive semi-definite).

5. [**4 points**] Derive the closed form solution to Eq. 3.

6. [**5 points**] Show that Eq. 3 admits the unique solution $\mathbf{w}^*_{\text{ridge}}$ for any matrix $\mathbf{X}$. Show that this even holds for the cases in (b) and (c) where Eq. 2 does not admit a unique solution w^*.

7. [**5 points**] What is the role of the term $\lambda \mathbf{w}^T \mathbf{w}$ in $\hat{R}_{\text{ridge}}(w)$? What happens to w^*_{ridge} as $\lambda \to 0$ and $\lambda \to \infty$?

[1] Without loss of generality, we assume that both $\mathbf{x_i}$ and y_i are centered, i.e. they have zero empirical mean. Hence we can neglect the otherwise necessary bias term b.

1.1 :
According to equation (2), equation (1) can be written as:

$$\underset{\mathbf{w}}{\operatorname{argmin}} \hat{R}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{Xw} - \mathbf{y}\|^2.$$

And $\|\mathbf{Xw} - \mathbf{y}\|^2$ is just $(\mathbf{Xw} - \mathbf{y})^{\mathbf{T}}(\mathbf{Xw} - \mathbf{y})$.

The gradient of the loss function is:

$$\nabla_{\mathbf{w}}\hat{R}(\mathbf{w}) = \nabla_{\mathbf{w}} \Big((\mathbf{Xw} - \mathbf{y})^{\mathbf{T}}(\mathbf{Xw} - \mathbf{y})\Big)$$

$$= \nabla_{\mathbf{w}} \Big(\mathbf{w}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{Xw} - \mathbf{y}^{\mathbf{T}}\mathbf{Xw} - \mathbf{w}^{\mathbf{T}}\mathbf{X}^{\mathbf{T}}\mathbf{y} + \mathbf{y}^{\mathbf{T}}\mathbf{y}\Big)$$

$$= 2\mathbf{X}^T\mathbf{Xw} - 2\mathbf{X}^T\mathbf{y}$$

Considering $\hat{R}(\mathbf{w}) = 0$:

$$\mathbf{X}^T\mathbf{Xw} - \mathbf{X}^T\mathbf{y} = 0$$

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

The $\mathbf{w}$, or $\mathbf{w}^*$, is the unique solution of the loss function.

1.2 :

$$\mathbf{X}^T\mathbf{X}$$

is not invertible if:

1. $\mathbf{X}$ is not full rank
2. $n < d$
3. $\mathbf{X}$ has $n - d + 1$ or more linearly dependent row vectors

Thus, consider $\mathbf{X}^T\mathbf{Xw} = \mathbf{X}^T\mathbf{y}$ as a linear transformation of vector $\mathbf{Xw}$ and $\mathbf{y}$. Since $\mathbf{X}$ is not full rank, we can find a vector $\mathbf{v}$ such that $\mathbf{X}^T\mathbf{v} = 0$. This means, $\mathbf{X}^T\mathbf{X}(\mathbf{w}+\mathbf{v}) = \mathbf{X}^T\mathbf{y}$ is also one solution.
Similarly, $\mathbf{X}^T\mathbf{X}(\mathbf{w} + k\mathbf{v}) = \mathbf{X}^T\mathbf{y}$ for any real number k is also true. There are infinitely many solutions.

1.3 :
Assumption: $\mathbf{X}$ has at least d linearly independent vectors. If $\mathbf{X} \in \mathbb{R}^{3\times 2}$, consider $\mathbf{X} = \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \mathbf{x_3} \end{bmatrix}$ where $\mathbf{x_1} = [1, 2]$, $\mathbf{x_2} = [2, 4]$, $\mathbf{x_3} = [3, 6]$, the assumption does not hold.

1.4:
Again,

$$\begin{aligned}\hat{R}_{\text{ridge}}(\mathbf{w}) &= \operatorname*{argmin}_{\mathbf{w}} \left[\sum_{i=1}^{n}(y_i - \mathbf{w}^T\mathbf{x_i})^2 + \lambda\mathbf{w}^T\mathbf{w}\right] \\ &= (\mathbf{Xw} - \mathbf{y})^T(\mathbf{Xw} - \mathbf{y}) + \lambda\mathbf{w}^T\mathbf{w}\end{aligned}$$

Taking the second derivative:

$$\nabla^2_{\mathbf{w}}\hat{R}_{\text{ridge}}(\mathbf{w}) = \nabla_{\mathbf{w}}\left(2\mathbf{X}^T\mathbf{Xw} - 2\mathbf{X}^T\mathbf{y} + 2\lambda\mathbf{w}\right)$$

Simplifying this, we have:

$$\nabla^2_{\mathbf{w}}\hat{R}_{\text{ridge}}(\mathbf{w}) = \mathbf{H} = 2\mathbf{X}^T\mathbf{X} + 2\lambda\mathbf{I}$$

Notice that,

$$\mathbf{w}^T\mathbf{X}^T\mathbf{Xw} = (\mathbf{Xw})^T(\mathbf{Xw}) = \|\mathbf{Xw}\|^2 \geq 0$$

And $\lambda\mathbf{w}^T\mathbf{w} = \lambda\|\mathbf{w}\|^2$ is also ≥ 0.

By defintion, $\mathbf{w}^T\mathbf{Hw} \geq 0$ means $\mathbf{H}$ is a positive semi-definite matrix and thus $\hat{R}_{\text{ridge}}(\mathbf{w})$ is a convex function.

1.5: From 1.4,we have calculated that

$$\nabla\hat{R}_{\text{ridge}}(\mathbf{w}) = 2\mathbf{X}^T\mathbf{Xw} - 2\mathbf{X}^T\mathbf{y} + 2\lambda\mathbf{w}$$

Let's set $\nabla\hat{R}_{\text{ridge}}(\mathbf{w}) = 0$:

$$2\mathbf{X}^T\mathbf{Xw} - 2\mathbf{X}^T\mathbf{y} + 2\lambda\mathbf{w} = 0$$

Solving for $\mathbf{w}$, we get:

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

1.6:
Consider the case in 1.2 and 1.3, $\mathbf{X}^T\mathbf{X}$ is not invertible implies that some eigenvalue of $\mathbf{X}$ is 0. From 1.4, we see that $\mathbf{X}^T\mathbf{X}$ is actually a positive definite matrix and because $\lambda > 0$, $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ should have all eigenvalues greater than 0. This means the matrix $\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}$ is invertible. Thus, $\mathbf{w}$ has a unique solution even if maxtrix $\mathbf{X}^T\mathbf{X}$ is not invertible.
1.7:
$\lambda\mathbf{w}^T\mathbf{w}$ is a regularization term to limit the complexity of $\mathbf{w}$. If $\lambda \to 0$, the whole function becomes the ordinary least squares and $\mathbf{w}^*_{\text{ridge}}$ can possibly grow unlimitedly big. If $\lambda \to \infty$, any non-zero values in $\mathbf{w}$ will result in infinity. The loss function will be minimized by setting $\mathbf{w}$ to 0. Therefore, $\mathbf{w}^*_{\text{ridge}}$ will be a zero vector.

Generalized Linear Models (25 pts)

In class, we have discussed generalized linear models (GLMs). GLMs are models for data where the response variable y follows a distribution from the exponential family, and the mean of the response variable is related to the predictors via a link function. Consider the following form of GLM:

$$p(y|\mathbf{x}, \eta) = b(y) \exp \left(y\eta - a(\eta) \right), \tag{4}$$

where p is the probability mass function or probability density function of y, $\eta := \mathbf{w}^T \mathbf{x}$ is the natural parameter, $a(\eta)$ is the log normalizer.

1. [**7 points**] Show that linear regression and logistic regression are special cases of the GLM in 4. Identify $b(y)$ and $a(\eta)$.
2. [**6 points**] Consider the Poisson regression model, which is defined by the following probability mass function:

   $$p(y|\mathbf{x}, \mathbf{w}) = \mathrm{Pois}(y|\exp(\mathbf{w}^T \mathbf{x})).$$

   Here, $\mathrm{Pois}(y|\mu) = \frac{\mu^y e^{-\mu}}{y!}$ is the Poisson distribution parameterized by μ, with support $y \in \mathbb{N}$. Show that Poisson regression belongs to the family of GLMs defined in Equation 4, and identify $a(\eta)$ for this model.
3. [**6 points**] Consider a dataset $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, where $\mathbf{x}_n \in \mathbb{R}^d$ and y_n is the response variable. Derive the negative log-likelihood (NLL) loss function for the Poisson regression model. Simplify the expression as much as possible.
4. [**6 points**] Prove that in Equation 4, $\mathbb{E}[y|\mathbf{x}] = a'(\eta)$ and $\mathrm{Var}[y|\mathbf{x}] = a''(\eta)$.

2.1: Linear regression: $y = \theta^T x + e$ where $e \sim N(0, \sigma^2)$. Assuming $\sigma^2 = 1$.
$P(\epsilon) = P(y - \theta^T x) = P(y|x;\theta)$

$$P(y|x;\theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(y - \theta^T x)^2\right)$$

$$P(y|x;\theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(y^2 - 2y\theta^T x + (\theta^T x)^2\right)\right)$$

$$P(y|x;\theta) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}y^2\right) \exp\left(y\theta^T x - \frac{1}{2}(\theta^T x)^2\right)$$

$$b(y) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}y^2\right), \quad \eta = \theta^T x, \quad T(y) = y, \quad a(\eta) = \frac{1}{2}\eta^2$$

Therefore, linear regression is a GLM with $b(y) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}y^2\right)$ and $a(\eta) = \frac{1}{2}\eta^2$.

Logistic regression: $y \in \{0, 1\}$ follows a Bernoulli distribution. Let $P(y|x;\theta) \sim$ Bernoulli(ϕ)

$$P(y|x;\theta) = \phi^y (1 - \phi)^{1-y}$$

$$P(y|x;\theta) = \exp\left(y \log(\phi) + (1 - y)\log(1 - \phi)\right)$$

$$P(y|x;\theta) = \exp\left(y \log\left(\frac{\phi}{1 - \phi}\right) + \log(1 - \phi)\right)$$

$$b(y) = 1, \quad \eta = \log\left(\frac{\phi}{1 - \phi}\right), \quad T(y) = y, \quad a(\eta) = -\log(1 - \phi)$$

Therefore, logistic regression is a GLM with $b(y) = 1$ and $a(\eta) = -\log(1 - \phi)$.

2.2: The Poisson regression:

$$P(y|\exp(w^T x)) = \frac{(\exp(w^T x))^y e^{-\exp(w^T x)}}{y!}$$

$$= \frac{\exp(y \log(\exp(w^T x)) - \exp(w^T x))}{y!}$$

$$b(y) = \frac{1}{y!}, \eta = w^T x, T(y) = y, a(\eta) = -\exp(w^T x)$$

Therefore, Poisson regression is a GLM with $b(y) = \frac{1}{y!}$ and $a(\eta) = -\exp(w^T x)$.

2.3: The NLL of $L(w)$ is:

$$L(w) = \prod_{i=1}^{n} \frac{\exp(y_i \log(\exp(w^T x_i)) - \exp(w^T x_i))}{y_i!}$$

$$l(w) = \log L(w) = \sum_{i=1}^{n} \log \left(\frac{\exp(y_i \log(\exp(w^T x_i)) - \exp(w^T x_i))}{y_i!} \right)$$

$$= \sum_{i=1}^{n} \Big(y_i \log(\exp(w^T x_i)) - \exp(w^T x_i) - \log(y_i!) \Big)$$

$$\mathrm{NLL}(w) = -l(w) = -\sum_{i=1}^{n} \Big(y_i \log(\exp(w^T x_i)) - \exp(w^T x_i) - \log(y_i!) \Big)$$

2.4:

$$\sum_y P(y|x; w) = 1$$

$$1 = \sum_y b(y) \exp(\eta y - a(\eta))$$

$$a(\eta) = \log \left(\sum_y b(y) \exp(\eta y) \right)$$

$$\nabla_\eta a(\eta) = \nabla_\eta \log \left(\sum_y b(y) \exp(\eta y) \right)$$

$$= \frac{\sum_y y b(y) \exp(\eta y)}{\sum_y b(y) \exp(\eta y)}$$

$$= E(y; \eta)$$

$$\nabla_\eta^2 a(\eta) = \nabla_\eta^2 \log \left(\sum_y b(y) \exp(\eta y) \right)$$

$$= \frac{\sum_y b(y) \exp(\eta y) \sum_y y^2 b(y) \exp(\eta y) - \Big(\sum_y y b(y) \exp(\eta y) \Big)^2}{\Big(\sum_y b(y) \exp(\eta y) \Big)^2}$$

$$= E(y^2; \eta) - E(y; \eta)^2$$

$$= \mathrm{Var}(y; \eta)$$

Support Vector Machines (31 pts)

In binary classification, we are interested in finding a hyperplane that separates two clouds of points living in, say, $\mathbb{R}^p$. The support vector machine (SVM), which we talked about in class, is a pretty popular method for doing binary classification; to this day, it's (still) used in a number of fields outside of just machine learning and statistics.

One issue arises with the standard SVM, though, when the data points are not linearly separable in $\mathbb{R}^p$, i.e., we cannot find a hyperplane which separates the two classes of points. In such cases, it is often useful to map the data points to a different space (potentially of higher dimension than $\mathbb{R}^p$) where the points become separable. Such maps are called nonlinear feature maps.

In this problem, you will develop a SVM with the RBF kernel to address the nonlinearly separable problem of the standard SVM. You will implement your own RBF-SVM in part (b) of this question, but as a starting point, we will first investigate the SVM dual problem in part (a) of this question.

Throughout, we assume that we are given n data samples, each one taking the form (x_i, y_i), where $x_i \in \mathbb{R}^p$ is a feature vector and $y_i \in \{-1, +1\}$ is a class. In order to make our notation more concise, we can transpose and stack the x_i vertically, collecting these feature vectors into the matrix $X \in \mathbb{R}^{n\times p}$; doing the same thing with the y_i lets us write $y \in \{-1, +1\}^n$.

The primal problem of SVM with slack variables is

$$\begin{array}{ll} \underset{\beta\in\mathbb{R}^p,\ \beta_0\in\mathbb{R},\ \xi\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^n \xi_i \\ \text{subject to} & \xi_i \geq 0, \quad i = 1,\dots,n, \\ & y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i, \quad i = 1,\dots,n, \end{array} \tag{5}$$

where $\beta \in \mathbb{R}^p$, $\beta_0 \in \mathbb{R}$, $\xi = (\xi_1, \dots, \xi_n) \in \mathbb{R}^n$ are our variables, and C is a positive margin coefficient chosen by the implementer.

(i, 2pts) Does strong duality (i.e. zero duality gap) hold for problem (5)? Why or why not? (Your answer to the latter question should be short.)

(ii, 8pts) Derive the Karush-Kuhn-Tucker (KKT) conditions for problem (5). Please use $\alpha \in \mathbb{R}^n$ for the dual variables (i.e., Lagrange multipliers) associated with the constraints "$y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i,\ i = 1,\dots,n$", and $\mu \in \mathbb{R}^n$ for the dual variables associated with the constraints "$\xi_i \geq 0,\ i = 1,\dots,n$".

(iii, 8pts) Show that the SVM dual problem can be written as

$$\begin{array}{ll} \underset{\alpha\in\mathbb{R}^n}{\text{maximize}} & -(1/2)\alpha^T\tilde{X}\tilde{X}^T\alpha + 1^T\alpha \\ \text{subject to} & \alpha^T y = 0, \\ & 0 \leq \alpha \leq C1, \end{array} \tag{6}$$

where $\tilde{X} \in \mathbb{R}^{n\times p} = \text{diag}(y)X$, α is the dual variable, and the 1's here are vectors (of the appropriate and possibly different sizes) containing only ones.

(iv, 5pts) Give an expression for the optimal β in terms of the optimal α variables and explain how.

Now we are going to take a glimpse of the "magic" of kernels. Let's first recall what is a kernel we learned in the lectures: Given a feature map $\phi : \mathbb{R}^d \to \mathcal{K}$, where $\mathcal{K}$ is a Hilbert space (i.e., a vector space with inner product), the kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the corresponding inner product function

$$K(x_i, x_j) \coloneqq \langle \phi(x_i), \phi(x_j) \rangle. \tag{7}$$

Here the feature map, as we mentioned earlier, is used to "embed" the original data into a higher dimensional space such that they become separable. Recall the objective of the dual SVM, and it can be rewritten as

$$\begin{aligned} & -\tfrac{1}{2}\alpha^T \tilde{X}\tilde{X}^T \alpha + 1^T \alpha && (8) \\ \Leftrightarrow & -\tfrac{1}{2}\alpha^T Y X X^T Y \alpha + 1^T \alpha && (9) \\ \Leftrightarrow & -\tfrac{1}{2}\alpha^T Y G Y \alpha + 1^T \alpha, && (10) \\ & && (11) \end{aligned}$$

where $Y = \text{diag}(y)$, and $G = XX^T$ is the so called Gram matrix (this is also called the Kernel Matrix in our lectures), $G_{ij} = \langle x_i, x_j \rangle$. One nice property of the Gram matrix of a kernel K is that

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle = G_{ij}. \tag{12}$$

Hence, the kernel builds a bridge between the feature maps and the original dual problem.

(vi, 8pts) Show that the Gram matrix of a kernel K is positive semidefinite. Let the dimension of the feature space after the feature map be p'. If $p << p'$, which one is more efficient to solve, the primal or the dual? Why?

Now we are going to probe into the infinite dimensional space. We have seen so far how to build a kernel from a given feature map, but can we do the opposite? Suppose a map K is a kernel, can we find the corresponding feature map ϕ such that $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{K}}$? In the lectures, we have learned that for $K(x_i, x_j)$ to be a valid kernel function, it is *sufficient and necessary* if the corresponding Kernel matrix (Gram Matrix) is symmetric positive semidefinite.

There is no need to go into such difficulty of finding the feature maps, however, since we have the kernel-feature map equivalence (12). We only need to compute the value of the kernel function, avoiding the complexity of computing the inner product of high dimensional feature maps.

Given this intuition, we consider the radial basis function (RBF) kernel

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle = \exp\left(-\gamma \|x_i - x_j\|^2\right). \tag{13}$$

In Eq. 13, γ controls the bandwidth of the kernel. For RBF kernel, the corresponding feature maps have infinite dimensional feature spaces. The RBF kernel is a reasonable measure of x_i and x_j's similarity, and is close to 1 when x_i and x_j are close, and near 0 when they are far apart. In the following problems, you are going to use the RBF kernel in SVM.

Part (b) (Bonus Questions)

Please submit your code as an appendix to this problem.

(i, 5pts, Bonus) Implement the dual SVM in problem (6) with the RBF kernel using a standard Quadratic Program solver (typically available as "quadprog" function in Matlab, R, or in `Mathprogbase.jl` in Julia; you may also refer `CVXOPT` in Python, `GORUBI`, or `MOSEK`). Load a small synthetic toy problem with inputs $X \in \mathbb{R}^{863\times 2}$ and labels $y \in \{-1, 1\}^{863}$ from `data.txt` and solve the dual SVM with $\gamma = \{10, 50, 100, 500\}$ and $C = \{0.01, 0.1, 0.5, 1\}$. Report the optimal objective values of the dual. **Hint for Python users:** you can use the function `cvxopt.solvers.qp` detailed in this page.

(ii, 5pts, Bonus) For each of the parameter pairs, show a scatter plot of the data and plot the decision border (where the predicted class label changes) on top. How and why does the decision boundary change with different pair of parameters?

3i: Yes. Because the primal problem satisfies Slater's Condition. By Slater's Condition, d* = p* and it is strong duality.
3ii: Using the generalized Lagrange Multiplier, the Lagrangian is given by:

$$L(\beta, \beta_0, \xi, \alpha, \mu) = \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left(y_i(x_i^T\beta + \beta_0) - 1 + \xi_i\right) - \sum_{i=1}^{n} \mu_i\xi_i$$

$$\nabla_\beta L(\beta, \beta_0, \xi, \alpha, \mu) = \beta - \sum_{i=1}^{n} \alpha_i y_i x_i = 0$$

$$\beta = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial L(\beta, \beta_0, \xi, \alpha, \mu)}{\partial \beta_0} = \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\frac{\partial L(\beta, \beta_0, \xi, \alpha, \mu)}{\partial \xi_i} = C - \alpha_i - \mu = 0$$

$$C = \alpha_i + \mu_i$$

As it is strong duality, KKT condition holds such that $\mu_i\xi_i = 0$
KKT Conditions:

$$L(\beta, \beta_0, \xi, \alpha, \mu) = \frac{1}{2}\|\beta\|_2^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i \left(y_i(x_i^T\beta + \beta_0) - 1 + \xi_i\right) - \sum_{i=1}^{n} \mu_i\xi_i$$

$$\nabla_\beta L(\beta^*, \beta_0^*, \xi^*, \alpha^*, \mu^*) = \beta^* - \sum_{i=1}^{n} \alpha_i^* y_i x_i = 0$$

$$\frac{\partial L(\beta^*, \beta_0^*, \xi^*, \alpha^*, \mu^*)}{\partial \beta_0} = \sum_{i=1}^{n} \alpha_i^* y_i = 0$$

$$\frac{\partial L(\beta^*, \beta_0^*, \xi^*, \alpha^*, \mu^*)}{\partial \xi_i} = C - \alpha_i^* - \mu^* = 0$$

$$\alpha_i^* \left(y_i(x_i^T\beta + \beta_0^*) - 1 + \xi_i^*\right) = 0, \mu_i^*\xi_i^* = 0$$

And the original constraint

$$1 - y_i(x_i^T\beta + \beta_0) - \xi_i \leq 0$$

$$\mu_i \geq 0$$

$$\alpha_i \geq 0$$

3iii: From ii), substituting back $\beta = \sum_{i=1}^{n} \alpha_i y_i x_i$, $\sum_{i=1}^{n} \alpha_i y_i = 0$, $C = \alpha_i + \mu_i$.
Lagrangian becomes:

$$\frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y_i x_i\right)^T \left(\sum_{j=1}^{n} \alpha_j y_j x_j\right) + \sum_{i=1}^{n}(\alpha_i + \mu_i)\xi_i - \sum_{i=1}^{n} \alpha_i \left(y_i(x_i^T(\sum_{j=1}^{n} \alpha_j y_j x_j) + \beta_0) - 1 + \xi_i \right)$$

$$= \frac{1}{2}(\sum_{i=1}^{n} \alpha_i y_i x_i)^T(\sum_{j=1}^{n} \alpha_j y_j x_j) + \sum_{i=1}^{n}(\alpha_i + \mu_i)\xi_i - \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i=1}^{n} \alpha_i y_i \beta_0 + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i \xi_i$$

Simplying it, we have

$$\sum_{i=1}^{n} \alpha_i + \sum_{i=1}^{n} \mu_i \xi_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

By KKT condition, $\mu_i \xi_i = 0$

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$\sum_{i=1}^{n} \alpha_i$ can be seen as the determinant of diagonal matrix with α_i in diagonal.
The second term is a inner product which can be rewritten to $(X^T I y \alpha)^T (X^T I y \alpha)$

$$= (\hat{X}^T \alpha)^T (\hat{X}^T \alpha) = \alpha^T \hat{X} \hat{X}^T \alpha \in R$$

Therefore, $L(\alpha) = -\frac{1}{2}\alpha^T \hat{X} \hat{X}^T \alpha + 1^T \alpha$

$$\min_{\alpha} L(\alpha) = \max_{\alpha} -L(\alpha) = \frac{1}{2}\alpha^T \hat{X} \hat{X}^T \alpha - 1^T \alpha$$

$$\sum_{i=1}^{n} \alpha_i y_i = \alpha^T y = 0$$

$$C = \alpha_i + \mu_i$$

$$0 \geq \alpha_i \geq C$$

$$0 \geq \alpha \geq C1$$

The dual problem is now:

$$\max_{\alpha} -\frac{1}{2}\alpha^T \hat{X} \hat{X}^T \alpha + 1^T \alpha$$

s.t.

$$\alpha^T y = 0$$

$$0 \geq \alpha \geq C1$$

3iv: The optimal $\beta^* = \sum_{i=1}^{n} \alpha_i^* y_i x_i = X^T I Y \alpha$.
Since $d^* = p^*$, by KKT condition, the original $beta = \sum_{i=1}^{n} \alpha_i y_i x_i$ is also correct for β^*, α^*

3v: $G_K = \phi(x)\phi(x)^T$
Consider

$$\alpha^T Y G_K Y \alpha = \alpha^T Y \phi(x)\phi(x)^T Y \alpha$$
$$= (\phi(x)^T Y \alpha)^T (\phi(x)^T Y \alpha) = \|\phi(x)^T Y \alpha\| \geq 0$$

This shows that G_K is positive semidefinite.
3vi: Even $p << p^*$, the efficiency of solving it is the same i.e. O(n). Because we can pre-compute all the $K(x_i, x_j)$ pairs for training and just calculate the $K(x_i, x)$ for inference.
3b:

```python
import numpy as np
import matplotlib.pyplot as plt


with open("data.txt", "r") as f:
    data = f.readlines()

X = []
y = []
for line in data:
    line = line.strip().split()
    X.append([float(line[0]), float(line[1])])
    y.append(float(line[2]))

X = np.array(X)
y = np.array(y)

#split the data into training and testing


# Load the toy dataset
data = np.loadtxt('data.txt')
X = data[:, :2]  # features (863 x 2)
y = data[:, 2]   # labels (863,)
y = y.reshape(-1, 1)  # reshape for consistency

plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o')
plt.show()

import cvxopt

def rbf_kernel(X1, X2, gamma):
    """Compute the RBF (Gaussian) kernel between two matrices X1 and X2."""
    sq_dists = np.sum(X1 ** 2, axis=1).reshape(-1, 1) + np.sum(X2 ** 2, axis=1) - 2
    return np.exp(-gamma * sq_dists)

def svm_dual_rbf(X, y, C, gamma):
    """Solves the SVM dual problem with an RBF kernel."""
    n_samples, n_features = X.shape
    # Compute the kernel matrix using the RBF kernel
    K = rbf_kernel(X, X, gamma)

    # Formulate the quadratic programming problem in cvxopt format
    P = cvxopt.matrix(np.outer(y, y) * K)  # n x n
    q = cvxopt.matrix(-np.ones((n_samples, 1)))  # n x 1
    G_std = cvxopt.matrix(np.vstack((np.eye(n_samples) * -1, np.eye(n_samples))))  #
    h_std = cvxopt.matrix(np.hstack((np.zeros(n_samples), np.ones(n_samples) * C)))
    A = cvxopt.matrix(y.T.astype(float))  # 1 x n
    b = cvxopt.matrix(np.zeros(1))  # 1

    #solve QP
    cvxopt.solvers.options['show_progress'] = False
    solution = cvxopt.solvers.qp(P, q, G_std, h_std, A, b)
    alpha = np.ravel(solution['x'])

    return alpha

# Parameters pairs
gammas = [10, 50, 100, 500]
Cs = [0.01, 0.1, 0.5, 1]

# Solve the dual SVM for different values of gamma and C
results = {}
for gamma in gammas:
    for C in Cs:
        alpha = svm_dual_rbf(X, y, C, gamma)
        objective_value = -(1/2) * np.dot(alpha, np.dot(np.outer(y, y) * rbf_kernel(
        results[(gamma, C)] = objective_value

# List the results
for (gamma, C), obj_val in results.items():
    print(f"Gamma: {gamma}, C: {C}, Objective Value: {obj_val}")
```

```
Gamma: 10, C: 0.01, Objective Value: 7.130305335448902
Gamma: 10, C: 0.1, Objective Value: 49.480753289296445
Gamma: 10, C: 0.5, Objective Value: 187.56122156215594
Gamma: 10, C: 1, Objective Value: 326.34206544232336
Gamma: 50, C: 0.01, Objective Value: 6.716281370135912
Gamma: 50, C: 0.1, Objective Value: 31.26060896926302
Gamma: 50, C: 0.5, Objective Value: 79.70705360878571
Gamma: 50, C: 1, Objective Value: 116.61153330042099
Gamma: 100, C: 0.01, Objective Value: 6.7948141489638125
Gamma: 100, C: 0.1, Objective Value: 28.374181922347265
Gamma: 100, C: 0.5, Objective Value: 60.78035518933515
Gamma: 100, C: 1, Objective Value: 83.41142489931661
Gamma: 500, C: 0.01, Objective Value: 7.306471849775369
Gamma: 500, C: 0.1, Objective Value: 42.41745966703799
Gamma: 500, C: 0.5, Objective Value: 64.36483396116256
```

```
Gamma: 500, C: 1, Objective Value: 73.96324152629589
```

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from matplotlib import cm
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_norm = scaler.fit_transform(X)
#Visualize the dataset and decision boundary original data
#Using sklearn to solve the dual SVM and plot the graph.
def plot_decision_boundary(X, y, model, gamma, C):
    plt.figure(figsize=(10, 8))

    x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
    y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200), np.linspace(y_min, y_max, 2

    grid = np.c_[xx.ravel(), yy.ravel()]

    # Compute the decision function
    Z = model.decision_function(grid)
    Z = Z.reshape(xx.shape)

    # Plot the decision boundary and margin
    plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), Z.max(), 100), cmap=cm.RdBu,
    plt.contour(xx, yy, Z, levels=[-1, 0, 1], linewidths=2, colors=['red', 'black',

    plt.contour(xx, yy, Z, levels=[0], linewidths=2, colors='black')

    # Plot the data points
    plt.scatter(X[:, 0], X[:, 1], c=y.ravel(), cmap=cm.RdBu, edgecolors='k', zorder=

    # Highlight support vectors
    plt.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1], facecolo

    plt.title(f'Decision Boundary (Gamma={gamma}, C={C})', fontsize=14)
    plt.legend()
    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)
    plt.show()

# Use sklearn's SVC with RBF kernel
for gamma in gammas:
    for C in Cs:
        model = SVC(kernel='rbf', C=C, gamma=gamma)
        model.fit(X, y)
        plot_decision_boundary(X, y, model, gamma, C)
```

```
# The blue region represents the positive class, and the red region represents the 
# As we can see, the decision boundary pattern is mainly controled by gamma while c
# When gamma and C is very large(500,1) the decision boundary is very close to the 
# When gamma and C is very small(10,0.01) the decision boundary is just a simple ci
# It is because when gamma is large, the data points have limited influence to the 
# When gamma is small, the data points have a large influence to the far region and
# To get the best model, we need to find the balance between gamma and C. In this c

import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from matplotlib import cm
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_norm = scaler.fit_transform(X)
#Visualize the dataset and decision boundary again with normalized data
#Using sklearn to solve the dual SVM and plot the graph.
def plot_decision_boundary(X, y, model, gamma, C):
    plt.figure(figsize=(10, 8))

    x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
    y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 200), np.linspace(y_min, y_max, 

    grid = np.c_[xx.ravel(), yy.ravel()]

    # Compute the decision function
    Z = model.decision_function(grid)
    Z = Z.reshape(xx.shape)

    # Plot the decision boundary and margin
    plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), Z.max(), 100), cmap=cm.RdBu
    plt.contour(xx, yy, Z, levels=[-1, 0, 1], linewidths=2, colors=['red', 'black', 

    plt.contour(xx, yy, Z, levels=[0], linewidths=2, colors='black')

    # Plot the data points
    plt.scatter(X[:, 0], X[:, 1], c=y.ravel(), cmap=cm.RdBu, edgecolors='k', zorder

    # Highlight support vectors
    plt.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1], facecol

    plt.title(f'Decision Boundary (Gamma={gamma}, C={C})', fontsize=14)
    plt.legend()
    plt.xlim(x_min, x_max)
    plt.ylim(y_min, y_max)
    plt.show()
```

```python
# Use sklearn's SVC with RBF kernel
for gamma in gammas:
    for C in Cs:
        model = SVC(kernel='rbf', C=C, gamma=gamma)
        model.fit(X_norm, y)
        plot_decision_boundary(X_norm, y, model, gamma, C)

# The normalized data shows the change of decision boundary with respect to differen
```