COM S 573: Machine Learning
Homework #2

1. (20 points) Consider the toy data set $\{([0,0],-1),([2,2],-1),([2,0],+1)\}$. Set up the dual problem for the toy data set. Then, solve the dual problem and compute α^*, the optimal Lagrange multipliers. (Note that there will be three weights $\boldsymbol{w} = [w_0, w_1, w_2]$ by considering the bias.)

   Solution: After including bias $\mathbf{x}_1 = [1\ 0\ 0]^T$, $\mathbf{x}_2 = [1\ 2\ 2]^T$, $\mathbf{x}_3 = [1\ 2\ 0]^T$ and $y_1 = -1$, $y_2 = -1$, $y_3 = +1$.

$$\begin{aligned}\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} &= \sum_{i=1}^{3} \alpha_i y_i \mathbf{x}_i \\ &= \alpha_1 \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} -1 \\ -2 \\ -2 \end{bmatrix} + \alpha_3 \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} -\alpha_1 - \alpha_2 + \alpha_3 \\ -2\alpha_2 + 2\alpha_3 \\ -2\alpha_2 \end{bmatrix} \end{aligned} \tag{1}$$

   We have

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{3} \alpha_i (1 - y_i(\mathbf{w}^T \mathbf{x}_i)) \\ &= \frac{1}{2}\|\mathbf{w}\|_2^2 + \alpha_1(1 + \mathbf{w}^T\mathbf{x}_1) + \alpha_2(1 + \mathbf{w}^T\mathbf{x}_2) + \alpha_3(1 + \mathbf{w}^T\mathbf{x}_3)\end{aligned} \tag{2}$$

   If we substitute 1 in 2, we get

$$\begin{aligned}\mathcal{L} =& \frac{1}{2}\left((-\alpha_1 - \alpha_2 + \alpha_3)^2 + (-2\alpha_2 + 2\alpha_3)^2 + (-2\alpha_2)^2\right) \\ &+ \alpha_1(1 - \alpha_1 - \alpha_2 + \alpha_3) + \alpha_2(1 - \alpha_1 - \alpha_2 + \alpha_3 + 2(-2\alpha_2 + 2\alpha_3) + 2(-2\alpha_2)) \\ &+ \alpha_3(1 + \alpha_1 + \alpha_2 - \alpha_3 - 2(-2\alpha_2 + 2\alpha_3))\end{aligned}$$

   After simplification, we get

$$\mathcal{L} = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}\alpha_1^2 - \frac{9}{2}\alpha_2^2 - \frac{5}{2}\alpha_3^2 - \alpha_1\alpha_2 + 5\alpha_2\alpha_3 + \alpha_1\alpha_3 \tag{3}$$

   Differentiate 3 with respect to α_1, α_2, and α_3 and equate it to zero. That gives us to solve the following system of linear equations:

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 9 & -5 \\ -1 & -5 & 5 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{4}$$

   The solution is $\alpha_1 = \frac{3}{2}, \alpha_2 = \frac{1}{2}$, and $\alpha_3 = 1$. By substituting these values in 1 and we get $\mathbf{w} = [-1\ 1\ -1]^T$.

2. (15 points) In a separable case, when a multiplier $\alpha_i > 0$, its corresponding data point $(\boldsymbol{x}_i, y_i)$ is on the boundary of the optimal separating hyperplane with $y_i(\boldsymbol{w}^T\boldsymbol{x}_i) = 1$.

Show that the inverse is not True. Namely, it is possible that $\alpha_i = 0$ and $(\boldsymbol{x}_i, y_i)$ is on the boundary satisfying $y_i(\boldsymbol{w}^T \boldsymbol{x}_i) = 1$.

[Hint: Consider a toy data set with two positive examples at ([0,0], +1) and ([1, 0], +1), and one negative example at ([0, 1], -1).] (Note that there will be three weights $\boldsymbol{w} = [w_0, w_1, w_2]$ by considering the bias.)

Solution: After including bias $\mathbf{x}_1 = [1\ 0\ 0]^T$, $\mathbf{x}_2 = [1\ 1\ 0]^T$, $\mathbf{x}_3 = [1\ 0\ 1]^T$ and $y_1 = +1$, $y_2 = +1$, $y_3 = -1$.

$$\begin{aligned}\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} &= \sum_{i=1}^{3} \alpha_i y_i \mathbf{x}_i \\ &= \alpha_1 \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \alpha_3 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \alpha_1 + \alpha_2 - \alpha_3 \\ \alpha_2 \\ -\alpha_3 \end{bmatrix} \end{aligned} \tag{5}$$

We have

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{3} \alpha_i (1 - y_i(\mathbf{w}^T \mathbf{x}_i)) \\ &= \frac{1}{2}\|\mathbf{w}\|_2^2 + \alpha_1(1 + \mathbf{w}^T \mathbf{x}_1) + \alpha_2(1 + \mathbf{w}^T \mathbf{x}_2) + \alpha_3(1 + \mathbf{w}^T \mathbf{x}_3)\end{aligned} \tag{6}$$

If we substitute 5 in 6, we get

$$\begin{aligned}\mathcal{L} =& \frac{1}{2}\left((\alpha_1 + \alpha_2 - \alpha_3)^2 + (\alpha_2)^2 + (\alpha_3)^2\right) \\ &+ \alpha_1(1 - \alpha_1 - \alpha_2 + \alpha_3) + \alpha_2(1 - \alpha_1 - 2\alpha_2 + \alpha_3) \\ &+ \alpha_3(1 + \alpha_1 + \alpha_2 - 2\alpha_3)\end{aligned}$$

After simplification, we get

$$\mathcal{L} = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}\alpha_1^2 - \alpha_2^2 - \alpha_3^2 - \alpha_1\alpha_2 + \alpha_2\alpha_3 + \alpha_1\alpha_3 \tag{7}$$

Differentiate 7 with respect to α_1, α_2, and α_3 and equate it to zero. That gives us to solve the following system of linear equations:

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \tag{8}$$

The solution is $\alpha_1 = 3, \alpha_2 = 0$, and $\alpha_3 = 2$. By substituting these values in 5 and we get $\mathbf{w} = [1\ 0\ -2]^T$. The value of $y_2(\mathbf{w}^T \mathbf{x}_2) = 1$ and the corresponding Lagrange multiplier $\alpha_2 = 0$. From this, we conclude that the Lagrange multiplier α_i can be zero for the corresponding data point $(\mathbf{x}_i, y_i)$ while satisfying $y_i(\mathbf{w}^T \mathbf{x}_i) = 1$.

3. (15 points) **Non-separable Case SVM:** In Lecture 8 (page 18), we compare the hard-margin SVM and soft-margin SVM. Prove that the dual problem of soft-margin SVM is almost identical

to the hard-margin SVM, except that α_is are now bounded by C (tradeoff parameter).

Solution: The objective function for the hard-margin SVM is

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N} \alpha_i(1 - y_i(\mathbf{w}^T\mathbf{x}_i)) \text{ subject to } \alpha_i \geq 0 \tag{9}$$

The objective function for the soft-margin SVM is

$$\begin{aligned}\mathcal{L} &= \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\zeta_i + \sum_{i=1}^{N}\alpha_i(1 - \zeta_i - y_i(\mathbf{w}^T\mathbf{x}_i)) - \sum_{i=1}^{N}\mu_i\zeta_i \text{ s.t. } \alpha_i \geq 0,\ \mu_i \geq 0 \\ &= \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N}\alpha_i(1 - y_i(\mathbf{w}^T\mathbf{x}_i)) + \sum_{i=1}^{N}(C - \alpha_i - \mu_i)\zeta_i \text{ s.t. } \alpha_i \geq 0,\ \mu_i \geq 0 \end{aligned} \tag{10}$$

If we differentiate 10 with ζ_i and equate it to zero. We get $C - \alpha_i - \mu_i = 0 \implies C = \alpha_i + \mu_i$. In addition, we have constraints $\alpha_i \geq 0$ and $\mu_i \geq 0$. Knowing that $\alpha_i \geq 0$, $\mu_i \geq 0$ and to satisfy $C = \alpha_i + \mu_i$ we need $\alpha_i \leq C$. Therefore, if we substitute the new constraints $C = \alpha_i + \mu_i$ and $0 \leq \alpha_i \leq C$ in 10. We get

$$\mathcal{L} = \frac{1}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{N} \alpha_i(1 - y_i(\mathbf{w}^T\mathbf{x}_i)) \text{ subject to } 0 \leq \alpha_i \leq C \tag{11}$$

In equations 9,10, and 11 $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$. By observing 9 and 11 we conclude that soft-margin SVM is identical to hard-margin SVM, except that α_i's are bounded by C.

4. (20 points) **Kernel Function:** A function K computes $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = -\boldsymbol{x_i}^T\boldsymbol{x}_j$. Is this function a valid kernel function for SVM? Prove or disprove it.

Solution: If we have n data points such as $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$ The kernel function matrix with respect to $K(\mathbf{x}_i, \mathbf{x}_j) = -\mathbf{x}_i^T\mathbf{x}_j$ is

$$K = \begin{bmatrix} -\mathbf{x}_1^T\mathbf{x}_1 & -\mathbf{x}_1^T\mathbf{x}_2 & \cdots & -\mathbf{x}_1^T\mathbf{x}_n \\ -\mathbf{x}_2^T\mathbf{x}_1 & -\mathbf{x}_2^T\mathbf{x}_2 & \cdots & -\mathbf{x}_2^T\mathbf{x}_n \\ \vdots & \vdots & \ddots & \vdots \\ -\mathbf{x}_n^T\mathbf{x}_1 & -\mathbf{x}_n^T\mathbf{x}_2 & \cdots & -\mathbf{x}_n^T\mathbf{x}_n \end{bmatrix} \tag{12}$$

Since we have $K(\mathbf{x}_i, \mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)^T$, the kernel matrix K (12) is symmetric. To verify positive semidefiniteness we have to show $\mathbf{z}^T K\mathbf{z} \geq 0\ \forall\ \mathbf{z} \in \mathbb{R}^n$. For example, if we consider $\mathbf{z} = (1, 0, 0 \ldots, 0)$ then $\mathbf{z}^T K\mathbf{z} = -\mathbf{x}_1^T\mathbf{x}_1 \leq 0 \implies K$ is not positive semidefinite. Therefore, $K(\mathbf{x}_i, \mathbf{x}_j) = -\mathbf{x}_i^T\mathbf{x}_j$ is not a kernel function.

5. (15 points) **Support Vectors:** In the **linearly separable** case, prove that we need at most $d + 1$ support vectors for a naive SVM, where d is the number of features. (Naive SVM means we use a hard-margin SVM with "linear" kernel.)

Solution: We have optimal $\mathbf{w}$ is a linear combination of all data points with a multiplier α_i for each data point i.e. $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$ with $\mathbf{x}_i \in \mathbb{R}^{d+1}$ (even though true $\mathbf{x}_i \in \mathbb{R}^d$). Because we add a bias as one feature to each data point $\mathbf{x}_i$. As a result, we get at most d+1 system of linear equations with variables as d+1 Lagrange multipliers. $\implies$ we get at most d+1 NON-ZERO Lagrange multiplier values, which assign to the at most d+1 data points on the margin lines ($y_i(\mathbf{w}^T\mathbf{x}_i) = 1$). Since we

know that the d+1 data points which lie on $y_i(\mathbf{w}^T\mathbf{x}_i) = 1$ are called support vectors. Therefore, we have at most d+1 support vectors with at most d+1 nonzero Lagrange multipliers. In non-separable cases, we have outlier examples that can lie on the other class margin-line ($y_i(\mathbf{w}^T\mathbf{x}_i) = 1$). So, we can have at most d+1 support vectors only in the linearly separable case.

6. (15 points) **Support Vector Machine for Handwritten Digits Recognition**: You need to use the software package scikit-learn https://scikit-learn.org/stable/modules/svm.html to finish this assignment. We will use "svm.SVC()" to create a svm model. The handwritten digits files are in the "data" folder: train.txt and test.txt. The starting code is in the "code" folder. In the data file, each row is a data example. The first entry is the digit label ("1" or "5"), and the next 256 are grayscale values between -1 and 1. The 256 pixels correspond to a 16×16 image. You are expected to implement your solution based on the given codes. The only file you need to modify is the "solution.py" file. You can test your solution by running "main.py" file. Note that code is provided to compute a two-dimensional feature (symmetry and average intensity) from each digit image; that is, each digit image is represented by a two-dimensional vector. These features along with the corresponding labels should serve as inputs to your solution functions.

   (a) (5 points) Complete the **svm_with_diff_c()** function. In this function, you are asked to try different values of cost parameter c.

   (b) (5 points) Complete the **svm_with_diff_kernel()** function. In this function, you are asked to try different kernels (linear, polynomial and radial basis function kernels).

   (c) (5 points) Summarize your observations from (a) and (b) into a short report. In your report, please report the accuracy result and total support vector number of each model. A briefly analysis based on the results is also needed. For example, how the number of support vectors changes as parameter value changes and why.2

   Solution:

```
For c = 0.010000, No. of support vectors = 540,540, and Train_Accuracy = 92.056374, Test_Accuracy = 89.386792
For c = 0.100000, No. of support vectors = 207,207, and Train_Accuracy = 98.014094, Test_Accuracy = 96.226415
For c = 1.000000, No. of support vectors = 81,81, and Train_Accuracy = 98.078155, Test_Accuracy = 95.990566
For c = 2.000000, No. of support vectors = 64,65, and Train_Accuracy = 98.142217, Test_Accuracy = 95.990566
For c = 3.000000, No. of support vectors = 58,59, and Train_Accuracy = 98.206278, Test_Accuracy = 96.226415
For c = 5.000000, No. of support vectors = 52,52, and Train_Accuracy = 98.206278, Test_Accuracy = 96.226415
For linear kernel, Number of support vectors = 81,81, and Train_Accuracy = 98.078155, Test_Accuracy = 95.990566
For poly kernel, Number of support vectors = 37,38, and Train_Accuracy = 98.270340, Test_Accuracy = 95.754717
For rbf kernel, Number of support vectors = 46,44, and Train_Accuracy = 98.270340, Test_Accuracy = 96.226415
```

Figure 1: Classification Accuracy for different c values and different kernel functions

More c means less regularization and small c means more regularization. From the figure 1, we can observe that as c increases from 0.01 to 5 both train and test accuracy are improved from 92.05 to 98.20 and 89.38 to 96.22 while the number of support vectors are reduced from 1080 to 104. This means if we focus more on regularization the accuracy is not so good. However, if we keep the regularization appropriately we can see the improvement in accuracy and decrease in number of support vectors. For the 2nd part, the rbf kernel is giving high test accuracy as 96.22 with 90 support vectors. The poly kernel has least number of support vectors as 75.