DDA3020 Homework 1 Solution

Due date: Oct 14, 2024

Instructions

- The **deadline** is 23:59, Oct 14, 2024.
- The weight of this assignment in the final grade is 20%.
- **Electronic submission**: Turn in solutions electronically via Blackboard. Be sure to submit your homework as a single file. Please name your solution file as $DDA3020HW1\_studentID\_name$
- Note that **late submissions** will result in discounted scores: 0-24 hours $\rightarrow$ 80%, 24-120 hours $\rightarrow$ 50%, 120 or more hours $\rightarrow$ 0%.
- Answer the questions in English. Otherwise, you'll lose half of the points.
- Collaboration policy: You need to solve all questions independently and collaboration between students is **NOT** allowed.

1 Written Problems (50 points)

1.1. (Learning of Linear Regression, 25 points) Suppose we have training data:

$$\{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\},$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $\boldsymbol{y}_i \in \mathbb{R}^k$, $i = 1, 2, \ldots, N$.

1) (9 pts) Find the closed-form solution of the following problem.

$$\min_{\boldsymbol{W}, \boldsymbol{b}} \sum_{i=1}^{N} \|\boldsymbol{y}_i - \boldsymbol{W}\boldsymbol{x}_i - \boldsymbol{b}\|_2^2,$$

2) (8 pts) Show how to use gradient descent to solve the problem. (Please state at least one possible Stopping Criterion)

3) (8 pts) We further suppose that $x_1, x_2, \ldots, x_N$ are drawn from $\mathcal{N}(\mu, \sigma^2)$. Show that the maximum likelihood estimation (MLE) of σ^2 is $\hat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{MLE})^2$.

1) Firstly, we have

$$\begin{aligned} J(\boldsymbol{W}) &= J(\boldsymbol{w}, \boldsymbol{b}) \\ &= \sum_{i=1}^{N} \|\boldsymbol{y}_i - \boldsymbol{W}\boldsymbol{x}_i - \boldsymbol{b}\|_2^2 \\ &= tr[(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W})^T(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{W})] \\ &= tr[\boldsymbol{Y}^T\boldsymbol{Y} - \boldsymbol{W}^T\boldsymbol{X}^T\boldsymbol{Y} - \boldsymbol{Y}^T\boldsymbol{X}\boldsymbol{W} + \boldsymbol{W}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{W}] \end{aligned} \quad \text{(3 pts)}$$

Then, we can calculate the derivative of J with respect to $\boldsymbol{W}$ as

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{W}} &= -\boldsymbol{X}^T\boldsymbol{Y} - \boldsymbol{X}^T\boldsymbol{Y} + (\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{X}^T\boldsymbol{X})\boldsymbol{W} \\ &= -2\boldsymbol{X}^T\boldsymbol{Y} + 2\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{W} \end{aligned} \quad \text{(3 pts)}$$

Let $\frac{\partial J}{\partial \boldsymbol{W}} = 0$, we have

$$\boldsymbol{W}^* = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y} \quad \text{(3 pts)}$$

2) (Each steps worth 2 points) The gradient descent procedure is

(a) Initialize: Choose learning rate β and iteration T, and initialize $t = 0$ and $\boldsymbol{W}_0$.

(b) Update parameters $\boldsymbol{W}$ with gradient descent

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \beta \frac{\partial J}{\partial \boldsymbol{W}}$$

(c) Repeat the last step for T-times iterations until convergence;

(d) Stopping criterion: a) maximum number of iterations reached; b) step size is smaller than tolerance.

3) The MLE of σ^2:

- (2 pts) Step 1: Write down the p.d.f. of Normal Distribution as

$$\begin{aligned} L(\mu, \sigma | x_1, \cdots, x_N) &= \prod_{i=1}^{N} f(x_i) = \prod_{i=1}^{N} \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{-(x_i - \mu)^2}{2\sigma^2}) \right) \\ &= \frac{1}{\sqrt{(2\pi\sigma^2)^N}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - \mu)^2 \right) \end{aligned}$$

- (2 pts) Step 2: Take the log of both sides, we have

$$\log L(\mu, \sigma | x_1, \cdots, x_N) = -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - \mu)^2$$

- (2 pts) Step 3: Setting the derivatives equal to 0, we have

$$\begin{cases} \frac{\partial}{\partial \mu} \log L(\mu, \sigma | x_1, \cdots, x_N) = \frac{1}{\sigma^2} \sum_{i=1}^{N} (x_i - \mu) = 0 \\ \frac{\partial}{\partial \sigma^2} \log L(\mu, \sigma | x_1, \cdots, x_N) = -\frac{N}{2\sigma^2} + \frac{1}{2(\sigma^2)^2} \sum_{i=1}^{N} (x_i - \mu)^2 = 0 \end{cases} \Rightarrow \begin{cases} \mu = \frac{1}{N} \sum_{i=1}^{N} x_i = \bar{x} \\ \sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2 \end{cases}$$

- (2 pts) Step 4: We have

$$\sigma_{mle}^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_{mle})^2, \text{ where } \mu_{mle} = \bar{x}$$

1.2. (Support Vector Machine, 25 points) Given two positive samples $x_1 = (3, 3)^T$, $x_2 = (4, 3)^T$, and one negative sample $x_3 = (1, 1)^T$, find the maximum-margin separating hyperplane and support vectors. Solution steps:

i) Formulating the Optimization Problem (5 pts)

ii) Constructing the Lagrangian (5 pts)

iii) Using KKT Conditions (5 pts)

iv) Solving the Equations (5 pts)

v) Determining the Hyperplane Equation and Support Vectors (5 pts)

1) (Formulating the Optimization Problem 5 pts)

The general form of the hyperplane is:

$$w^T x + b = 0$$

Constraints: For positive samples x_1 and x_2:

$$w^T x_1 + b \geq 1$$
$$w^T x_2 + b \geq 1$$

For the negative sample x_3:

$$w^T x_3 + b \leq -1$$

Optimization Goal: Maximize the margin, which is equivalent to minimizing $\frac{1}{2}\|w\|^2$.

Mathematical Expression of the Optimization Problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 \\ \text{subject to} \quad & w^T x_1 + b \geq 1 \\ & w^T x_2 + b \geq 1 \\ & w^T x_3 + b \leq -1 \end{aligned}$$

2) (Constructing the Lagrangian 5 pts)

To introduce Lagrange multipliers, we transform the inequalities into equalities and introduce multipliers $\alpha_i \geq 0$ for each constraint.

The Lagrangian function $L(w, b, \alpha)$ is:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \alpha_1(w^T x_1 + b - 1) - \alpha_2(w^T x_2 + b - 1) - \alpha_3(-w^T x_3 - b - 1)$$

where: α_1, α_2 correspond to the positive samples' constraints, α_3 corresponds to the negative sample's constraint

3) (Using KKT Conditions 5 pts)

The KKT (Karush-Kuhn-Tucker) conditions include:

(a) **Primal Constraints:**

$$\begin{cases} w^T x_1 + b \geq 1 \\ w^T x_2 + b \geq 1 \\ w^T x_3 + b \leq -1 \\ \alpha_1, \alpha_2, \alpha_3 \geq 0 \end{cases}$$

(b) **Dual Feasibility:**

$$\alpha_i(\text{constraint}) = 0 \quad \forall i$$

(c) **Gradient Conditions:**

$$\frac{\partial L}{\partial w} = w - \alpha_1 x_1 - \alpha_2 x_2 + \alpha_3 x_3 = 0$$

$$\frac{\partial L}{\partial b} = -\alpha_1 - \alpha_2 + \alpha_3 = 0$$

4) (Solving the Equations 5 pts)

Gradient Condition:

$$w = \alpha_1 x_1 + \alpha_2 x_2 - \alpha_3 x_3$$

Support Vector Conditions (Equality Constraints):

For support vectors, we have:

$$w^T x_i + b = \text{label} \times 1$$

To determine the support vectors, we need to consider different cases. We know at least two of the samples must be support vectors. We will explore three scenarios:

Case 1: Assuming all three samples are support vectors(1 pts)

We need to solve the following set of equations:

Assuming all are support vectors (to verify), we have:

$$\begin{cases} (w^T x_1) + b = 1 \\ (w^T x_2) + b = 1 \\ (w^T x_3) + b = -1 \end{cases}$$

Substituting w into the above equations gives us:

For $x_1 = (3,3)^T$:

$(\alpha_1 x_1 + \alpha_2 x_2 - \alpha_3 x_3)^T x_1 + b = 1$

$\alpha_1(3 \times 3 + 3 \times 3) + \alpha_2(4 \times 3 + 3 \times 3) - \alpha_3(1 \times 3 + 1 \times 3) + b = 1$

$18\alpha_1 + 21\alpha_2 - 6\alpha_3 + b = 1$

For $x_2 = (4,3)^T$:

$(\alpha_1 x_1 + \alpha_2 x_2 - \alpha_3 x_3)^T x_2 + b = 1$

$\alpha_1(3 \times 4 + 3 \times 3) + \alpha_2(4 \times 4 + 3 \times 3) - \alpha_3(1 \times 4 + 1 \times 3) + b = 1$

$21\alpha_1 + 25\alpha_2 - 7\alpha_3 + b = 1$

For $x_3 = (1,1)^T$:

$(\alpha_1 x_1 + \alpha_2 x_2 - \alpha_3 x_3)^T x_3 + b = -1$

$\alpha_1(3 \times 1 + 3 \times 1) + \alpha_2(4 \times 1 + 3 \times 1) - \alpha_3(1 \times 1 + 1 \times 1) + b = -1$

$6\alpha_1 + 7\alpha_2 - 2\alpha_3 + b = -1$

Combining $\alpha_1 + \alpha_2 = \alpha_3$, we have three equations and three unknowns:

$$\begin{cases} 18\alpha_1 + 21\alpha_2 - 6\alpha_3 + b = 1 \\ 21\alpha_1 + 25\alpha_2 - 7\alpha_3 + b = 1 \\ 6\alpha_1 + 7\alpha_2 - 2\alpha_3 + b = -1 \end{cases}$$

Simplification Steps:

From the equations, we can simplify as follows:

$$18\alpha_1 + 21\alpha_2 - 6\alpha_3 + b = 1$$

$$21\alpha_1 + 25\alpha_2 - 7\alpha_3 + b = 1$$

$$6\alpha_1 + 7\alpha_2 - 2\alpha_3 + b = -1$$

Further simplification leads to:

$$\begin{cases} 12\alpha_1 + 15\alpha_2 + b = 1 & (1) \\ 14\alpha_1 + 18\alpha_2 + b = 1 & (2) \\ 4\alpha_1 + 5\alpha_2 + b = -1 & (3) \end{cases}$$

Subtracting (1) from (2):

$$(2) - (1) : 2\alpha_1 + 3\alpha_2 = 0 \quad (4)$$

Subtracting (1) from (3):

$$(1) - (3) : 8\alpha_1 + 10\alpha_2 = 2 \quad (5)$$

From equation (4), we have:

$$2\alpha_1 + 3\alpha_2 = 0 \implies \alpha_1 = -\frac{3}{2}\alpha_2$$

Given $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$, the only solution is $\alpha_1 = \alpha_2 = 0$.

However, from (5):

$$8\alpha_1 + 10\alpha_2 = 2$$

This means we have a contradiction, implying that not all samples can be support vectors. In fact, only some samples can be support vectors.

Case 2: Assuming x_2 and x_3 Are Support Vectors(1 pts)

From the previous calculations, we find:

$$\alpha_2 = \alpha_3 = \frac{2}{13}.$$

The weight vector components are:

$$w_1 = \frac{6}{13}, \quad w_2 = \frac{4}{13}.$$

The bias term is:

$$b = -\frac{23}{13}.$$

Now, we check the KKT condition for x_1:

$$w^T x_1 + b = \frac{7}{13}.$$

Since $\frac{7}{13} < 1$, the KKT condition $w^T x_1 + b \geq 1$ is not satisfied.

Therefore, the assumption that x_2 and x_3 are the only support vectors is invalid.

Case 3: Assuming x_1 and x_3 Are Support Vectors(3 pts)

From the calculations, we find:

$$\alpha_1 = \alpha_3 = \frac{1}{4}.$$

The weight vector components are:

$$w_1 = w_2 = \frac{1}{2}.$$

The bias term is:

$$b = -2.$$

Now, we check the KKT condition for x_2:

$$w^T x_2 + b = \frac{3}{2}.$$

Since $\frac{3}{2} > 1$, the KKT condition $w^T x_2 + b \geq 1$ is satisfied.

Therefore, the assumption that x_1 and x_3 can be valid support vectors holds true.

5) (Determining the Hyperplane Equation and Support Vectors 5 pts)

Hyperplane equation (3 pts)The maximum-margin separating hyperplane equation is:

$$\frac{1}{2}x_1 + \frac{1}{2}x_2 - 2 = 0$$

Support vectors (2 pts)The support vectors are $x_1 = (3, 3)^T$ and $x_3 = (1, 1)^T$.

2 Programming (50 points)

2.1. (Linear regression, 25 points) We have a labeled dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_n, y_n)\}$, with $\mathbf{x}_i \in \mathbb{R}^d$ being the d-dimensional feature vector of the i-th sample, and $y_i \in \mathbb{R}$ being real valued target (label).

A linear regression model is give by

$$f_{w_0,\ldots,w_d}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d, \tag{1}$$

where w_0 is often called bias and $w_1, w_2, \ldots, w_d$ are often called coefficients.

Now, we want to utilize the dataset $\mathcal{D}$ to build a linear model based on linear regression. We provide a training set $\mathcal{D}_{\text{train}}$ that includes 2024 labeled samples with 11 features (See linear_regression_train.txt) to fit model, and a test set $\mathcal{D}_{\text{test}}$ that includes 10 unlabeled samples with 11 features (see linear_regression_test.txt) to estimate model.

1. Using the LinearRegression class from Sklearn package to get the bias w_0 and the coefficients $w_1, w_2, \ldots, w_{11}$, then computing the $\hat{y} = f(\mathbf{x})$ of test set $\mathcal{D}_{\text{test}}$ by the model trained well. (Put the estimation of $w_0, w_1, \ldots, w_{11}$ and these $\hat{y}$ in your answers.)

2. Implementing the linear regression by yourself to obtain the bias w_0 and the coefficients $w_1, w_2, \ldots, w_{11}$, then computing the $\hat{y} = f(\mathbf{x})$ of test set $\mathcal{D}_{\text{test}}$. (Put the estimation of $w_0, w_1, \ldots, w_{11}$ and these $\hat{y}$ in your answers. It is allowed to compute the inverse of a matrix using the existing python package.)

   (Hint: Note that for linear_regression_train.txt, there are 2024 rows with 12 columns where the first 11 columns are features $\mathbf{x}$ and the last column is target y and linear_regression_test.txt only contains 10 rows with 11 columns (features). Both of two tasks require the submission of code and results. Put all the code in a "HW1_yourID_Q1.ipynb" Jupyter notebook. file.(".py" file is also acceptable))

According to this problem, we need to build a linear regression model based on given dataset $\mathcal{D}$. Depending on the linear regression model, we know the closed form solution (the unique solution) for $\mathbf{W}^*$ as

$$\mathbf{W}^* = [w_0, w_1, \cdots, w_d]^T = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y},$$

where $\mathbf{X} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n]^T$ and $\tilde{\mathbf{x}}_i = [1, x_1, \ldots, x_d]^T$.

Thus, for Question 1 and 2, we should obtain the same solution for $\mathbf{W}^*$. The solution of $\mathbf{W}^*$ is as follows:

$$\begin{aligned} w_0 &= 3.613646 & w_1 &= 0.015325 \\ w_2 &= 0.000252 & w_3 &= 0.000720 \\ w_4 &= 0.999163 & w_5 &= 0.999740 \\ w_6 &= 1.000623 & w_7 &= 0.998832 \\ w_8 &= 1.000001 & w_9 &= 1.000224 \\ w_{10} &= 0.999039 & w_{11} &= 0.999344 \end{aligned}$$

Also, for Question 1 and 2, we should obtain the same predictions for $\mathcal{D}_{\text{test}}$. The predictions orderly of $\mathcal{D}_{\text{test}}$ are as follows:

$$\begin{aligned} \hat{y}_1 &= -56.111296 & \hat{y}_2 &= -173.516519 \\ \hat{y}_3 &= -6.770877 & \hat{y}_4 &= 209.517090 \\ \hat{y}_5 &= 116.890297 & \hat{y}_6 &= -100.290845 \\ \hat{y}_7 &= -310.127839 & \hat{y}_8 &= 501.386301 \\ \hat{y}_9 &= 244.114767 & \hat{y}_{10} &= 18.566393 \end{aligned}$$

An example code is provided in "HW1_Q1.py".

Scoring of Rubric

1. (10 pts)
   - (10 pts): The code runs correctly and gives the correct results for $\mathbf{W}^*$ and the predictions of $\mathcal{D}_{\text{test}}$.
   - (0 pts): The results are from report and output of code totally different.
   - (-1 pts): Deducting one point for one less result, up to a maximum of 10 points.
2. (15 points)
   - (15 pts): The code runs correctly and gives the correct results for $\mathbf{W}^*$ and the predictions of $\mathcal{D}_{\text{test}}$.
   - (5 pts): Correctly using the closed form solution in programming.
   - (-1 pts): Deducting one point for one less result, up to a maximum of 10 points.

2.2. (SVM, 25 points)

Task Description You are asked to write a program that constructs support vector machine models with different kernel functions and slack variables.

Datasets You are provided with the iris dataset. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. There are four features: 1. sepal length in cm; 2. sepal width in cm; 3. petal length in cm; 4. petal width in cm. You need to use these features to classify each iris plant as one of the three possible types.

What you should do You should use the SVM function from python sklearn package, which provides various forms of SVM functions. For multiclass SVM you should use the one vs rest strategy. You are recommended to use sklearn.svm.svc() function. You can use numpy for vector manipulation. For technical report, you should report the results required as mentioned below (e.g. training error, testing error, and so on).

Notably, there exists some ambiguity in the question statements. All reasonable interpret are acceptable in this homework:

- **About one v.s. rest strategy and its implementation**: sklearn.svm.SVC supports to specify decision_function_shape='ovr'. However, this function is actually a one-vs-one method and just aggregates results of clf.decision_function(X) into the same shape as that using one-vs-rest strategy. As the question statement doesn't distinguish this difference, this function

parameter (i.e. decision_function_shape='ovr') is really misleading. This implementation is also acceptable in this homework. Finally, there are three types of implementations: (1) train three binary classifiers by calling sklearn.svm.SVC three times manually (2) use OneVsRestClassifier (3) call sklearn.svm.SVC one time (and specify decision_function_shape='ovr')

- **About train/test error per class**: It could be interpreted as (1) train/test error for each class (i.e. only consider samples in the target class) based on the multi-class prediction, or (2) train/test error of each class-specific binary classifier (i.e. consider all samples). Both are acceptable.
- **About support vector indices and slack variables per class**: It could be interpreted as (1) all support vector indices and slack variables learned by each class-specific binary classifier (i.e. support vectors might belong to different classes), (2) class-specific support vector indices and slack variables learned by each class-specific binary classifier (i.e. must output support vectors belonging to the target class).

All of these implementations and reference solutions could see 'HW1_SVM_code.ipynb'.

1. **(2 points) Split training set and test set**. Split the data into a training set and a test set. The training set should contain 70% of the samples, while the test set should include 30%. The number of samples from each category in both the training and test sets should reflect this 70-30 split; for each category, the first 70% of the samples will form the training set, and the remaining 30% will form the test set. Ensure that the split maintains the original order of the data. You should report instance ids in the split training set and test set. The output format is as follows:

```
Q2.2.1 Split training set and test set:
Training set:  xx

Test set:  xx
```

You should fill up xx in the template. You should write ids for each set in the same line with comma separated, e.g. `Training set:[1, 4, 19]`.

1 point for correct splitting results and 1 point for executable code. If only the training or test set is split correctly, deduce half.

```
Q2.2.1 Split training set and test set:
Training set:  [1,2,3,...,33,34,35,51,52,53,...,83,84,85,101,102,103,...,133,134,135]

Test set:  [36,37,38,...48,49,50,86,87,88,...,99,100,136,137,138,...,148,149,150]
```

2. **(10 points) Calculation using Standard SVM Model (Linear Kernel)**. Employ the standard SVM model with a linear kernel. Train your SVM on the split training dataset and

validate it on the testing dataset. Calculate the classification error for both the training and testing datasets, output the weight vector $\mathbf{w}$, the bias b, and the indices of support vectors (start with 0). Note that the scikit-learn package does not offer a function with hard margin, so we will simulate this using $C = 1e5$. You should first print out the total training error and testing error, where the error is $\frac{\text{wrong prediction}}{\text{number of data}}$. Then, print out the results for each class separately (note that you should calculate errors for each class separately in this part). You should also mention in your report which classes are linear separable with SVM without slack. The output format is as follows:

```
Q2.2.2 Calculation using Standard SVM Model:
total training error: xx, total testing error: xx,

class setosa:
training error: xx, testing error: xx,
w: xx, b: xx,
support vector indices: xx,

class versicolor:
training error: xx, testing error: xx,
w: xx, b: xx,
support vector indices: xx,

class virginica:
training error: xx, testing error: xx,
w: xx, b: xx,
support vector indices: xx,

Linear separable classes: xx
```

If we view the one vs all strategy as combining the multiple different SVM, each one being a separating hyperplane for one class and the rest of the points, then the w, b and support vector indices for that class is the corresponding parameters for the SVM separating this class and the rest of the points. If a variable is of vector form, say $a = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$, then you should write each entry in the same line with comma separated e.g. `[1,2,3]`.

Scoring scheme:

- 2 point for "total training error" and "total testing error" calculation. Half for each.
- 2 point for "querying results by category".
- 1 point for "training error per class" and "testing error per class" calculation. Half for each.
- 1 point for "$\mathbf{w}$ and b". Half for each.

- 2 point for "support vector indices".
- 2 point for "linear separable classes".

Specifically,

- "Total training error" and "total testing error": 2 points will be awarded if both errors are reported correctly. If only one is accurate, 1 point will be deducted.
- "Querying results by category": 2 points will be awarded if all class-specific results are correctly matched. If results are misassigned (e.g., results for setosa reported as versicolor) but can be correctly matched by reordering (e.g. switch results between class setosa and class versicolor), 1 point will be awarded. If reordering does not resolve the misalignment, 0 points will be awarded.
- "Training error per class" and "testing error per class": 0.5 points will be awarded for each. Switching results between classes is acceptable, and no repeated deductions will be made for misclassification (e.g. results for setosa reported as versicolor).
- "$\mathbf{w}$ and b": 0.5 points will be awarded for each. Switching results between classes is allowed.
- "Support vector indices": 2 points will be awarded if it is correct. Switching results between classes is allowed.
- "Linear separable classes": 2 points will be awarded if it is correct. The answer must be fully correct. No points will be awarded for responses with either additional or missing answers.
- "Executable code": If the code for any requirement is not executable, half of the points awarded for that part will be deducted. Scores should be rounded down, e.g., 0.25 should be rounded to 0.2.

3. **(6 points) Calculation using SVM with Slack Variables (Linear Kernel)**. For each $C = 0.25 \times t$, where $t = 1, 2, \ldots, 4$, train your SVM on the training dataset, and subsequently validate it on the testing dataset. Calculate the classification error for both the training and testing datasets, the weight vector $\mathbf{w}$, the bias b, and the indices of support vectors, and the slack variable ζ of support vectors (you may compute it as $max(0, 1 - y \cdot f(X))$. The output format is as follows:

```
Q2.2.3 Calculation using SVM with Slack Variables (C = 0.25 × t, where t = 1,...,4):
-----------------------------------------
C=0.25,
total training error:  xx, total testing error:  xx,

class setosa:
training error:  xx, testing error:  xx,
```

```
w:  xx, b:  xx,
support vector indices:  xx,
slack variable:  xx,


class versicolor:
training error:  xx, testing error:  xx,
w:  xx, b:  xx,
support vector indices:  xx,
slack variable:  xx,


class virginica:
training error:  xx, testing error:  xx,
w:  xx, b:  xx,
support vector indices:  xx,
slack variable:  xx,
-----------------------------------------
C=0.5,
<...  results for (C=0.5) ...>
-----------------------------------------
C=0.75,
<...  results for (C=0.75) ...>
-----------------------------------------
C=1,
<...  results for (C=1) ...>
```

Compared to the previous question, this one introduces "SVM with slack variables", "iteration across various C" and "reporting slack variables". Scoring scheme:

- "Basic implementation of SVM with slack variables": 4 point will be awarded if "total training error", "total testing error", "querying results by category", "training error per class", "testing error per class", "$\mathbf{w}$ and b", "support vector indices" are all reported correctly. If there exists any mistake similar to that in the previous question, while at least one of them is correct, 1 point will be awarded. If all of them are incorrect, 0 points will be awarded. For simplicity, in this part, we only refer to the results of "C=0.25" (code logic across various C should be the same).
- "Iteration across various C": 1 point will be awarded if the correctness of results from different C demonstrates consistent patterns (e.g. all of them are incorrect in reporting $\mathbf{w}$ and b). Otherwise 0.
- "Reporting slack variables": 1 points will be awarded if it is correct. Switching results between classes is allowed.

- "Executable code": If the code for any requirement is not executable, half of the points awarded for that part will be deducted. Scores should be rounded down, e.g., 0.25 should be rounded to 0.2.

4. **(7 points) Calculation using SVM with Kernel Functions**. Conduct experiments with different kernel functions for SVM without slack variable. Calculate the classification error for both the training and testing datasets, and the indices of support vectors for each kernel type:

   (a) 2nd-order Polynomial Kernel

   (b) 3nd-order Polynomial Kernel

   (c) Radial Basis Function Kernel with $\sigma = 1$

   (d) Sigmoidal Kernel with $\sigma = 1$

   The output format is as follows:

```
Q2.2.4 Calculation using SVM with Kernel Functions:
-------------------------------------------
(a) 2nd-order Polynomial Kernel,
total training error:  xx, total testing error:  xx,

class setosa:
training error:  xx, testing error:  xx,
w:  xx, b:  xx,
support vector indices:  xx,

class versicolor:
training error:  xx, testing error:  xx,
w:  xx, b:  xx,
support vector indices:  xx,


class virginica:
training error:  xx, testing error:  xx,
w:  xx, b:  xx,
support vector indices:  xx,
-------------------------------------------
(b) 3nd-order Polynomial Kernel,
<...  results for (b) ...>
-------------------------------------------
(c) Radial Basis Function Kernel with σ = 1,
<...  results for (c) ...>
-------------------------------------------
(d) Sigmoidal Kernel with σ = 1,
```

```
<...  results for (d) ...>
```

Compared to the previous question, this one introduces "SVM with kernel functions", "different kernel functions". Scoring scheme:

- "Basic implementation of SVM with kernel functions": 4 point will be awarded if "total training error", "total testing error", "querying results by category", "training error per class", "testing error per class", "$\mathbf{w}$ and b", "support vector indices" are all reported correctly. If there exists any mistake similar to that in the previous question, while at least one of them is correct, 1 point will be awarded. If all of them are incorrect, 0 points will be awarded. In this part, we only refer to the results of "(a) 2nd-order Polynomial Kernal". Notably, we only care about the correctness of applying kernel functions and a different C is acceptable.
- "Different kernel functions": For each kernel function among (b), (c), (d), 1 point will be awarded if the correctness of results demonstrates consistent patterns (e.g. all of them are incorrect in reporting $\mathbf{w}$ and b) to that of (a). There are 3 points in total.
- "Executable code": If the code for any requirement is not executable, half of the points awarded for that part will be deducted. Scores should be rounded down, e.g., 0.25 should be rounded to 0.2.

Submission Submit your technical report as "A1_yourID_Q2.2.pdf" and your executable code in a "A1_yourID_Q2.2.ipynb" Jupyter notebook. Indicate the corresponding question number in the comment for each cell, and ensure that your code can logically produce the required results for each question in the required format. Please note that you need to write clear comments and use appropriate function/variable names. Excessively unreadable code may result in point deductions.