DDA3020 Homework 1

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 one pdf file plus two python scripts. Please name your solution files as "DDA3020HW1_studentID_name.pdf", "HW1_yourID_Q1.ipynb" and "HW1_yourID_Q2.ipynb". (.py files also acceptable)
- 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 $\boldsymbol{y}_i \in \mathbb{R}^k$, $i = 1, 2, \ldots, N$.

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

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

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

iii) (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 $\hat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{MLE})^2$.

1.2. (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)

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))

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).

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]`.

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]`.

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) ...>
```

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) ...>
```

Submission Submit your executable code in a "HW1_yourID_Q2.ipynb" Jupyter notebook(".py" file is also acceptable). 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.

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

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

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

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

i)

$$J(W) = \sum_{i=1}^{N} \| y_i - W x_i - b \|_2^2$$

$$= (y - Wx - b)^T (y - Wx - b)$$

$$= (y^T - x^T W^T - b^T)(y - Wx - b)$$

$$= (y^T y - y^T W x - y^T b - y x^T W^T + x^T W^T W x + x^T W^T b - b^T y + b^T W x + b^T b)$$

- Set $\frac{\partial}{\partial W} J(W) = 0$:

$$0 = -y^T x - y x^T + 2 x^T x W + x^T b + b^T x$$

$$2(y^T x - x^T b) = 2 x^T x W$$

$$W = (x^T x)^{-1}(y^T x - x^T b)$$

ii) Gradient descent $w \leftarrow w - \alpha \cdot \frac{\partial J(w,b)}{\partial w}$, $b \leftarrow b - \lambda \cdot \frac{\partial J(w,b)}{\partial b}$, where α, λ are learning rates

let cost function $J(w, b) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$, where $\hat{y}_i = w x_i + b$

$$\Rightarrow J(w, b) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - (w x_i + b))^2$$

$$\frac{\partial J}{\partial w} = \frac{1}{N} \sum_{i=1}^{N} (-x_i)(y_i - (w x_i + b))$$

At min point: $\frac{\partial J}{\partial w} = 0$, $0 = \sum_{i=1}^{N} (-x_i y_i + w x_i^2 + x_i b)$

Rearranging: $\sum_{i=1}^{N} wx_i + \sum_{i=1}^{N} x_i b = \sum_{i=1}^{N} x_i y_i$ ——— (1)

Now partial derivative w.r.t. 'b'

$$\frac{\partial J}{\partial b} = \frac{1}{N} \sum_{i=1}^{N} -\left(y_i - (wx_i + b)\right)$$

At min point: $\frac{\partial J}{\partial b} = 0$

$$0 = \sum_{i=1}^{N} (y_i + wx_i + b) \iff \sum_{i=1}^{N} y_i = w \sum_{i=1}^{N} x_i + \sum_{i=1}^{N} b$$

$$\iff \sum_{i=1}^{N} y_i = w \sum_{i=1}^{N} x_i + Nb \quad \text{——— (2)}$$

With the partial derivatives $\frac{\partial J}{\partial w}$ and $\frac{\partial J}{\partial b}$, we can update our weight & bias using the formulas:

$$(*) \quad w \leftarrow w - \alpha \frac{\partial J}{\partial w}$$

$$(**) \quad b \leftarrow b - \lambda \frac{\partial J}{\partial b}$$

} This is the gradient descent looping.

Stopping criterion:

(1) Max iterations: # of loops exceed a defined threshold.

(2) change in cost b/t iterations is less than some small threshold ε.

i.e. $\| \nabla f(w,b) \| < \varepsilon \iff | f(w,b)_{prev} - f(w,b)_{current} | < \varepsilon$.

iii) $L = (\mu, \sigma^2 \mid x_1 \dots x_n) = \prod_{i=1}^{n} f(x_i \mid \mu, \sigma^2)$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_i-\mu)^2/2\sigma^2}$$

$$= \left(\frac{1}{\sqrt{2\pi\sigma^2}^n}\right) e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2}$$

$$l(\mu, \sigma^2) = 0 - \frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2$$

$$= -\frac{n}{2}\ln 2 - \frac{n}{2}\ln\pi - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2$$

To obtain $\hat{\sigma}^2_{MLE}$, take partial derivative wrt to σ^2

$$\frac{\partial l(\mu,\sigma^2)}{\partial \sigma^2} = 0 - \frac{n}{2\sigma^2} - \frac{1}{2}\sum_{i=1}^{n}(x_i-\mu)^2 \cdot \frac{1}{(\sigma^2)^2}$$

At max, $\frac{\partial l(\mu,\sigma^2)}{\partial \sigma^2} = 0$

$$\Rightarrow \quad 0 = -\frac{n}{2\sigma^2} - \frac{1}{2\sigma^4}\sum_{i=1}^{n}(x_i-\mu)^2$$

$$0 = n\sigma^2 - \sum_{i=1}^{n}(x_i-\mu)^2$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2$$

$$\therefore \hat{\sigma}^2_{MLE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_{MLE})^2 \quad \text{(shown)}$$

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:

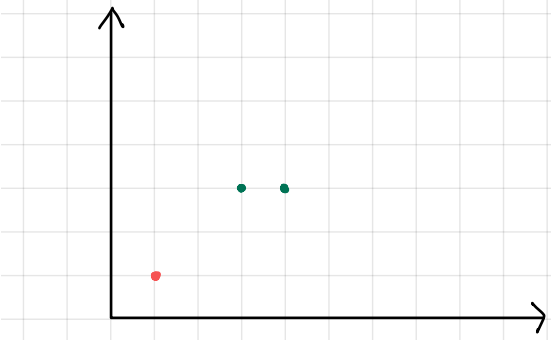
i) Formulating the Optimization Problem (5 pts)

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)



i) Hyperplane: $w^T x + b = 0$

impose constraints on x_i's, $i = 1, 2, 3$

+ve: $w^T x_1 + b \geq 1$, $w^T x_2 + b \geq 1$

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

Then, $y_i (w^T x_i + b) \geq 1$, $\forall i$, with margin $\frac{1}{\|w\|}$

$\therefore$ obj function: $\min_{w,b} \frac{1}{2}\|w\|^2$ s.t. $y_i(w^T x_i + b) \geq 1$, $\forall i$

$\Leftrightarrow \min_{w,b} \frac{1}{2}\|w\|^2$ s.t. $1 - y_i(w^T x_i + b) \leq 0$, $\forall i$

ii) The Lagrangian: $\mathcal{L}(w, b, \alpha) = \frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \alpha_i (1 - y_i (w^T x_i + b))$

Substituting the datapoints: $\{(x_1 = (3,3)^T, y_1 = 1), (x_2 = (4,3)^T, y_2 = 1), (x_3 = (1,1)^T, y_3 = -1)\}$

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

where $x_1 = (3,3)^T$, $x_2 = (4,3)^T$, $x_3 = (1,1)^T$

iii) The primal & dual optimal solutions should satisfy KKT conditions

Stationarity: $\frac{\partial \mathcal{L}}{\partial w} = \frac{1}{2}(2w) - \alpha_1 x_1 - \alpha_2 x_2 + \alpha_3 x_3 = w - \alpha_1 x_1 - \alpha_2 x_2 + \alpha_3 x_3 = 0$

$\Rightarrow w = \alpha_1 x_1 + \alpha_2 x_2 - \alpha_3 x_3 = \alpha_1 (3,3)^T + \alpha_2 (4,3)^T - \alpha_3 (1,1)^T$ (*)

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

$\Leftrightarrow \alpha_1 + \alpha_2 = \alpha_3$ (**)

Feasibility: $\alpha_1 \geq 0$, $1 - w^T x_1 - b \leq 0 \Leftrightarrow 1 - w^T (3,3)^T - b \leq 0$

$\alpha_2 \geq 0$, $1 - w^T x_2 - b \leq 0 \Leftrightarrow 1 - w^T (4,3)^T - b \leq 0$

$\alpha_3 \geq 0$, $1 + w^T x_3 + b \leq 0 \Leftrightarrow 1 + w^T (1,1)^T + b \leq 0$

Complementary Slackness:

$$\alpha_1 - \alpha_1 w^T x_1 - \alpha_1 b = 0 \iff \alpha_1 - \alpha_1 w^T (3,3)^T - \alpha_1 b = 0$$

$$\alpha_2 - \alpha_2 w^T x_2 - \alpha_2 b = 0 \iff \alpha_2 - \alpha_2 w^T (4,3)^T - \alpha_2 b = 0$$

$$\alpha_3 + \alpha_3 w^T x_3 + \alpha_3 b = 0 \iff \alpha_3 - \alpha_3 w^T (1,1)^T - \alpha_3 b = 0$$

iv) solving the Eqns:

From the feasibility conditions, we have the equalities

$$\left.\begin{array}{l} (1): w^T (3,3)^T + b = 1 \\ (2): w^T (4,3)^T + b = 1 \\ (3): w^T (1,1)^T + b = -1 \end{array}\right\} \text{Sub } w \text{ into these eqns}$$

For (1): $\left[\alpha_1 (3,3)^T + \alpha_2 (4,3)^T - \alpha_3 (1,1)^T\right]^T (3,3)^T + b = 1$

$$\left[\begin{pmatrix} 3\alpha_1 \\ 3\alpha_1 \end{pmatrix} + \begin{pmatrix} 4\alpha_2 \\ 3\alpha_2 \end{pmatrix} - \begin{pmatrix} \alpha_3 \\ \alpha_3 \end{pmatrix}\right]^T \begin{pmatrix} 3 \\ 3 \end{pmatrix} + b = 1$$

$$\begin{bmatrix} 3\alpha_1 + 4\alpha_2 - \alpha_3 \\ 3\alpha_1 + 3\alpha_2 - \alpha_3 \end{bmatrix}^T \begin{pmatrix} 3 \\ 3 \end{pmatrix} + b = 1$$

$$9\alpha_1 + 12\alpha_2 - 3\alpha_3 + 9\alpha_1 + 9\alpha_2 - 3\alpha_3 + b = 1$$

$$18\alpha_1 + 21\alpha_2 - 6\alpha_3 + b = 1 \quad \text{——— (4)}$$

For (2): $\begin{bmatrix} 3\alpha_1 + 4\alpha_2 - \alpha_3 \\ 3\alpha_1 + 3\alpha_2 - \alpha_3 \end{bmatrix}^T \begin{pmatrix} 4 \\ 3 \end{pmatrix} + b = 1$

$$12\alpha_1 + 16\alpha_2 - 4\alpha_3 + 9\alpha_1 + 9\alpha_2 - 3\alpha_3 + b = 1$$

$$21\alpha_1 + 25\alpha_2 - 7\alpha_3 + b = 1 \quad \text{——— (5)}$$

For (3): $\begin{bmatrix} 3\alpha_1 + 4\alpha_2 - \alpha_3 \\ 3\alpha_1 + 3\alpha_2 - \alpha_3 \end{bmatrix}^T \begin{pmatrix} 1 \\ 1 \end{pmatrix} + b = -1$

$$6\alpha_1 + 7\alpha_2 - 2\alpha_3 + b = -1 \quad \text{——— (6)}$$

Taking ④ − 3× ⑥ : $b - 3b = 1 - (-3)$

$-2b = 4$

$b = -2$

Taking ⑤ − 3× ⑥ : $3\alpha_1 + 4\alpha_2 - \alpha_3 - 2b = 1 - (-3)$

Sub in $\alpha_3 = \alpha_1 + \alpha_2$, $b = -2$

$3\alpha_1 + 4\alpha_2 - \alpha_1 - \alpha_2 + 4 = 4$

$2\alpha_1 + 3\alpha_2 = 0$ ——— ⑦

sub in $\alpha_3 = \alpha_1 + \alpha_2$, $b = -2$ into ⑤:

$21\alpha_1 + 25\alpha_2 - 7\alpha_1 - 7\alpha_2 - 2 = -1$

$14\alpha_1 + 18\alpha_2 = 3$ ——— ⑧

solving ⑦ and ⑧ : $\alpha_1 = 3/2$, $\alpha_2 = -1$, however, $\alpha_2 \geq 0$ to satisfy KKT

$\therefore \alpha_1 = 3/2$, $\alpha_2 = 0$ and $\alpha_3 = 3/2$

and $w = \frac{3}{2}\begin{pmatrix} 3 \\ 3 \end{pmatrix} + (0)\begin{pmatrix} 4 \\ 3 \end{pmatrix} - \frac{3}{2}\begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$= \begin{pmatrix} 9/2 \\ 9/2 \end{pmatrix} - \begin{pmatrix} 3/2 \\ 3/2 \end{pmatrix}$

$= \begin{pmatrix} 3 \\ 3 \end{pmatrix}$

v)

Hyperplane Eqn. $\begin{pmatrix} 3 \\ 3 \end{pmatrix} x_i - 2 = 0 \iff 3x_1 + 3x_2 = -2$

The support vectors : $x_1 = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$, $x_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

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.)

In [4]:
```
# Bias of fitted model
print('Bias:', linreg.intercept_)

# Coefficients of fitted model
for i, coef in enumerate(linreg.coef_, start=1):
    print(f'w_{i}: {coef}')
```

```
Bias: 3.6136460266690635
w_1: 0.015325690386246947
w_2: 0.0002522764063409011
w_3: 0.000720386849964616
w_4: 0.9991635645789793
w_5: 0.9997402360310115
w_6: 1.0006233993583626
w_7: 0.9988323588429893
w_8: 1.0000013015236555
w_9: 1.0002245483558736
w_10: 0.9990396151482012
w_11: 0.9993448460859441
```

In [6]:
```
# Predict on test data
y_hat = linreg.predict(X_test)

# Output predictions with labels
for i, y in enumerate(y_hat, start=1):
    print(f'y_hat_{i}: {y}')
```

```
y_hat_1: -56.11129687958417
y_hat_2: -173.5165197093181
y_hat_3: -6.770877912226868
y_hat_4: 209.51709044187203
y_hat_5: 116.89029785098518
y_hat_6: -100.29084527235771
y_hat_7: -310.12783900144865
y_hat_8: 501.3863019426078
y_hat_9: 244.11476780856393
y_hat_10: 18.566393254550345
```

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.)

In [13]:
```
# Output bias
print(f"Manual Bias (w_0): {w_0_manual}")
```

```
Manual Bias (w_0): 3.6136460266689294
```

In [14]:
```
# Output the coefficients
for i, coef in enumerate(w_manual, start=1):
    print(f'Manual Coefficient w_{i}: {coef}')
```

```
Manual Coefficient w_1: 0.015325690386246597
Manual Coefficient w_2: 0.00025227640634090803
Manual Coefficient w_3: 0.0007203868499653398
Manual Coefficient w_4: 0.9991635645789786
Manual Coefficient w_5: 0.999740236031012
Manual Coefficient w_6: 1.0006233993583618
Manual Coefficient w_7: 0.9988323588429896
Manual Coefficient w_8: 1.0000013015236555
Manual Coefficient w_9: 1.0002245483558734
Manual Coefficient w_10: 0.9990396151482013
Manual Coefficient w_11: 0.9993448460859444
```

In [15]:
```
# Output the y_hats
for i, y in enumerate(y_hat_manual, start=1):
    print(f'Manual Predicted value y_hat_{i}: {y}')
```

```
Manual Predicted value y_hat_1: -56.11129687958437
Manual Predicted value y_hat_2: -173.5165197093183
Manual Predicted value y_hat_3: -6.770877912226993
Manual Predicted value y_hat_4: 209.51709044187183
Manual Predicted value y_hat_5: 116.89029785098501
Manual Predicted value y_hat_6: -100.29084527235779
Manual Predicted value y_hat_7: -310.1278390014488
Manual Predicted value y_hat_8: 501.3863019426076
Manual Predicted value y_hat_9: 244.11476780856376
Manual Predicted value y_hat_10: 18.56639325455041
```

Question 2: SVM

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].

```
Q2.2.1 Split training set and test set:
Training set: [1, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 18, 22, 25, 26, 27, 28, 29, 30, 33, 35, 36, 38, 39, 40, 41, 42,
43, 44, 46, 47, 48, 49, 52, 54, 55, 57, 58, 59, 60, 62, 63, 67, 68, 69, 70, 71, 72, 73, 76, 78, 80, 82, 83, 84, 85, 86, 87, 8
8, 89, 92, 93, 94, 95, 96, 97, 98, 100, 101, 102, 103, 105, 106, 107, 108, 110, 112, 113, 117, 118, 119, 120, 122, 123, 126,
127, 128, 129, 130, 131, 132, 133, 136, 137, 138, 140, 141, 143, 144, 146, 147, 149, 150]
Length of training set: 105

Test set: [2, 3, 9, 17, 19, 20, 21, 23, 24, 31, 32, 34, 37, 45, 50, 51, 53, 56, 61, 64, 65, 66, 74, 75, 77, 79, 81, 90, 91, 9
9, 104, 109, 111, 114, 115, 116, 121, 124, 125, 134, 135, 139, 142, 145, 148]
Length of test set: 45
```

70-30 train test split with Stratify = y, y: categories
→ ensure class balance across train and test set.

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 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:

$$\text{error} = \frac{\text{wrong predictions}}{\text{number of data}}$$

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: 0.0000, total testing error: 0.0444

Class Iris-setosa:
  Training error: 0.0000, Testing error: 0.0000
  w: [-0.08757428  0.44848304 -0.85088498 -0.44006999], b: 1.600566373049051
  Support vector indices: [39 15 28]


Class Iris-versicolor:
  Training error: 0.2571, Testing error: 0.2667
  w: [-0.99913856 -3.1328945   1.73950159 -3.13977114], b: 11.166458055426018
  Support vector indices: [  1   6  10  15  16  22  24  28  71  75  76  78  80  82  83  84  85  87
  90  91  95  96  99 100 101 104  35  37  38  39  40  41  42  45  48  51
  52  53  54  55  56  57  58  59  61  62  63  64  66  68  69]


Class Iris-virginica:
  Training error: 0.0000, Testing error: 0.0444
  w: [ -1.16007155 -15.28442522  11.40002276  45.05463381], b: -82.99849854088012
  Support vector indices: [56 75 83 90]


Linear separable classes: ['Iris-setosa']
```

3. **(6 points)** Calculation using SVM with Slack Variables (Linear Kernel). For each C = 0.25 × t, where t = 1, 2, . . . , 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 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 · f(X)). The output format is as follows:

```
Q2.2.3 Calculation using SVM with Slack Variables (C = 0.25):
Total training error: 0.0286, total testing error: 0.0222

Class Iris-setosa:
  Training error: 0.0000, Testing error: 0.0000
  w: [-0.08005903  0.34791736 -0.80803542 -0.38898819], b: 1.7271981074005207
  Support vector indices: [39 64 15 16 28]
  Slack variable ζ:
  [1.14405493e-01 7.16078542e-02 8.72311025e-02 0.00000000e+00
 6.63229678e-08]

Class Iris-versicolor:
  Training error: 0.3429, Testing error: 0.3111
  w: [-0.29652586 -0.92272353  0.25916813 -0.24939231], b: 2.9813980553423742
  Support vector indices: [  1   6   9  10  16  20  22  24  25  28  29  31  70  71  72  73  74  75
  76  78  79  80  82  83  84  85  86  87  88  89  90  91  93  96  99 101
 102 104  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50
  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
  69]
  Slack variable ζ:
  [9.58099040e-02 3.17913772e-02 1.27799503e-01 1.98311994e-01
 9.53887258e-02 0.00000000e+00 3.17913772e-02 3.17913772e-02
 1.95553803e-01 7.86868455e-01 1.10090979e-02 7.79210416e-02
 0.00000000e+00 6.18106615e-01 1.13261981e-01 2.40321453e-01
 1.46416740e-01 9.63902179e-01 3.24714104e-01 4.92024725e-01
 9.85524879e-02 2.62327938e-01 5.13725540e-01 1.09400332e+00
 5.08366578e-01 3.52164902e-01 0.00000000e+00 3.54412712e-01
 2.25437405e-01 4.27624349e-01 1.82388736e-01 3.10561015e-01
 4.02685118e-01 1.99708171e-01 6.18106615e-01 5.76174618e-04
 6.28471579e-01 3.36576202e-01 2.07691462e+00 1.05929577e+00
 1.71156098e+00 2.11263999e+00 1.08025260e+00 1.78360746e+00
 1.39028347e+00 1.82185742e+00 1.04046866e+00 1.65514922e+00
 1.41660843e+00 1.09488592e+00 1.24953141e+00 1.92571894e+00
 1.69857305e+00 1.29768831e+00 1.95265266e+00 1.90162207e+00
 1.45018437e+00 1.15450087e+00 1.51832052e+00 1.36638086e+00
 1.59584405e+00 2.16778821e+00 2.02176640e+00 1.19284921e+00
 1.70893801e+00 1.75255611e+00 1.40013136e+00 1.01763284e+00
 1.40620414e+00 1.68773456e+00 1.62040143e+00 1.74274755e+00
 1.55404589e+00]

Class Iris-virginica:
  Training error: 0.0286, Testing error: 0.0222
  w: [-0.20999813 -0.40119896  1.51745415  1.15132542], b: -6.9279255322176825
  Support vector indices: [ 35  37  38  44  46  48  50  52  56  57  58  59  60  62  71  75  78  80
  83  84  87  88  90  96  97  99 101 102 104]
  Slack variable ζ:
  [0.00000000e+00 2.91006792e-01 3.99284896e-01 2.48019900e-01
 4.42980189e-01 9.05414541e-01 9.08602351e-01 1.00601412e+00
 1.30998537e+00 2.90019525e-01 1.18673608e-01 2.80392895e-01
 0.00000000e+00 1.79633710e-01 3.02617380e-01 1.17411685e+00
 1.25125425e-01 7.81266363e-02 7.56293109e-01 4.89095939e-01
 9.97105314e-01 9.04599877e-01 1.54162660e-04 9.72467193e-02
 8.59357303e-03 3.02617380e-01 0.00000000e+00 4.79122067e-01
 5.59109421e-01]

------------------------------------------------------------
```

```
Q2.2.3 Calculation using SVM with Slack Variables (C = 0.5):
Total training error: 0.0095, total testing error: 0.0000

Class Iris-setosa:
  Training error: 0.0000, Testing error: 0.0000
  w: [-0.09064252  0.43699535 -0.82820392 -0.42711552], b: 1.6080746216144002
  Support vector indices: [39 64 15 28]
  Slack variable ζ:
  [0.05252663 0.         0.00023585 0.        ]

Class Iris-versicolor:
  Training error: 0.2667, Testing error: 0.2444
  w: [-0.59353897 -1.84475159  0.519061   -0.50001632], b: 6.962308796885159
  Support vector indices: [  1   6   9  10  16  20  22  24  25  28  29  31  70  71  72  73  74  75
  76  78  79  80  82  83  84  85  86  87  88  89  90  91  93  96  99 101
 102 104  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50
  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68
  69]
```

```
  Slack variable ζ:
  [1.91887812e-01 6.38277514e-02 2.55750708e-01 3.96801897e-01
 1.90853481e-01 0.00000000e+00 6.38277514e-02 6.38277514e-02
 3.91258567e-01 1.57322915e+00 2.23082488e-02 1.55747445e-01
 0.00000000e+00 1.23613353e+00 2.26352917e-01 4.80568571e-01
 2.92926127e-01 1.92783559e+00 6.49749575e-01 9.83822342e-01
 1.96790211e-01 5.24856798e-01 1.02718790e+00 2.18790196e+00
 1.01655233e+00 7.04430279e-01 0.00000000e+00 7.08526112e-01
 4.50835791e-01 8.55062219e-01 3.65101078e-01 6.21057006e-01
 8.05060587e-01 3.99735537e-01 1.23613353e+00 4.22547066e-04
 1.25640826e+00 6.73355785e-01 2.15379572e+00 1.18861449e-01
 1.42334288e+00 2.22510642e+00 1.60551026e-01 1.56716868e+00
 7.80608125e-01 1.64379422e+00 8.11508827e-02 1.31001423e+00
 8.32912784e-01 1.90336337e-01 4.99068501e-01 1.85131283e+00
 1.39736063e+00 5.95491314e-01 1.90545767e+00 1.80337986e+00
 9.00520325e-01 3.09050012e-01 1.03672825e+00 7.32569372e-01
 1.19130643e+00 2.33533208e+00 2.04357006e+00 3.86068230e-01
 1.41763536e+00 1.50487598e+00 8.00346987e-01 3.54297645e-02
 8.12303784e-01 1.37508153e+00 1.24060800e+00 1.48547139e+00
 1.10803894e+00]

Class Iris-virginica:
  Training error: 0.0095, Testing error: 0.0000
  w: [-0.47842617 -0.73060146  1.79897066  1.39397694], b: -6.076031617746529
  Support vector indices: [ 37  38  44  46  48  50  52  56  57  59  62  71  75  83  84  87  88  90
  96  97  99 102 104]
  Slack variable ζ:
  [1.34744660e-01 1.84423927e-01 2.39310856e-01 5.36736319e-01
 9.07546995e-01 9.89301531e-01 8.91322789e-01 1.48590092e+00
 3.34996090e-01 0.00000000e+00 4.05971435e-02 2.20766684e-04
 7.81694708e-01 4.68093115e-01 1.97992117e-01 9.43740274e-01
 8.62120882e-01 4.81114591e-02 0.00000000e+00 2.43549723e-04
 2.20766684e-04 2.73210627e-01 4.06641515e-01]

----------------------------------------------------------

Q2.2.3 Calculation using SVM with Slack Variables (C = 0.75):
Total training error: 0.0095, total testing error: 0.0222

Class Iris-setosa:
  Training error: 0.0000, Testing error: 0.0000
  w: [-0.08757428  0.44848304 -0.85088498 -0.44006999], b: 1.600566373049051
  Support vector indices: [39 15 28]
  Slack variable ζ:
  [0.         0.         0.00017836]

Class Iris-versicolor:
  Training error: 0.2667, Testing error: 0.2667
  w: [-0.66491966 -1.91408278  0.62765747 -0.68013498], b: 7.370522498738445
  Support vector indices: [  1   6   9  10  16  22  24  25  28  29  31  71  72  73  74  75  76  78
  79  80  82  83  84  85  86  87  88  89  90  91  93  96  99 102 104  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  65  66  67  68  69]
  Slack variable ζ:
  [0.18369467 0.05223227 0.24736676 0.39152935 0.17190083 0.05223227
 0.05223227 0.38255538 1.58790786 0.         0.11133977 1.25476162
 0.18224022 0.45041277 0.28914062 1.99543838 0.69576805 0.98134132
 0.13065312 0.53416952 1.04055201 2.28810733 1.00279228 0.73624445
 0.         0.67710174 0.4235429  0.84220329 0.39304999 0.62713937
 0.77418979 0.4092532  1.25476162 1.2423526  0.68205832 2.20577205
 0.0624716  1.44386516 2.27317037 0.09024782 1.56573841 0.75940806
 1.67879291 0.         1.29101977 0.76077437 0.15870534 0.43851887
 1.88905548 1.41846478 0.54835915 1.95069168 1.84462965 0.87946861
 0.23813662 1.02233286 0.67418181 1.15803584 2.39063424 2.08830818
 0.34334434 1.40605576 1.49270035 0.76815883 0.76906518 1.34176848
 1.2183737  1.48806779 1.08973117]
```

```
Class Iris-virginica:
  Training error: 0.0095, Testing error: 0.0222
  w: [-0.47993535 -0.82062349  1.91535002  1.40786102], b: -6.444012709712783
  Support vector indices: [ 37  38  44  46  48  50  52  56  57  75  83  84  87  88  90  97  99 102
 104]
  Slack variable ζ:
  [6.10633402e-02 7.90597706e-02 1.37345459e-01 5.05883038e-01
 8.26203467e-01 9.77842462e-01 8.48663783e-01 1.48155448e+00
 2.33332529e-01 8.34815845e-01 4.40454883e-01 2.28459325e-01
 9.89527743e-01 9.14123904e-01 0.00000000e+00 1.01219714e-04
 1.00145618e-04 2.67478126e-01 4.35066830e-01]

----------------------------------------------------------

Q2.2.3 Calculation using SVM with Slack Variables (C = 1.0):
Total training error: 0.0190, total testing error: 0.0222

Class Iris-setosa:
  Training error: 0.0000, Testing error: 0.0000
  w: [-0.08757428  0.44848304 -0.85088498 -0.44006999], b: 1.600566373049051
  Support vector indices: [39 15 28]
  Slack variable ζ:
  [0.         0.         0.00017836]

  Class Iris-versicolor:
    Training error: 0.2571, Testing error: 0.2667
    w: [-0.85020736 -1.93586122  0.78235794 -0.85925556], b: 8.090107653809966
    Support vector indices: [  1   6   9  10  16  22  24  25  28  29  31  71  72  73  74  75  76  78
    79  80  82  83  84  85  87  88  89  90  91  93  96  99 102 104  35  36
    37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
    55  56  57  58  59  60  61  62  63  65  66  67  68  69]
    Slack variable ζ:
    [0.17966979 0.01053314 0.2109042  0.4013005  0.11140876 0.01053314
   0.01053314 0.3868258  1.57098236 0.         0.03905309 1.28951955
   0.05752685 0.40348992 0.18007372 2.14431478 0.65179134 0.93586672
   0.         0.51248476 0.93225374 2.35287512 1.02357776 0.64638658
   0.60706866 0.38315295 0.80513687 0.3238981  0.51795959 0.71921131
   0.40391938 1.28951955 1.17335232 0.70966601 2.31424801 0.02611414
   1.54668847 2.35226737 0.         1.65344421 0.70955777 1.73667947
   0.         1.24690988 0.71950838 0.20834532 0.4046918  1.91221362
   1.50416917 0.56118125 2.08942748 1.96281012 0.91031628 0.19663097
   1.04783108 0.62274525 1.0768684  2.44726287 2.21925251 0.39333686
   1.38800194 1.50785221 0.77600916 0.72900778 1.30886132 1.20120076
   1.54806865 1.08585043]

  Class Iris-virginica:
    Training error: 0.0190, Testing error: 0.0222
    w: [-0.47621069 -0.88197452  1.96671932  1.48916104], b: -6.71548102224447
    Support vector indices: [ 37  38  44  46  48  50  52  56  57  75  83  84  87  88  90  96 102 104]
    Slack variable ζ:
    [2.71245956e-04 1.14159714e-04 5.57940327e-02 4.75647235e-01
   7.73300029e-01 9.50121536e-01 8.13154139e-01 1.45884981e+00
   1.51036171e-01 8.72039013e-01 4.45750968e-01 2.36542814e-01
   1.01677337e+00 9.48875273e-01 4.91857360e-04 0.00000000e+00
   2.57542115e-01 4.60289271e-01]

  ----------------------------------------------------------
```

We observed that the slack variables help to introduce "soft margin" that allow the SVM model to better approximate and handle real-world (noisy) data. Our slack variables can also improve generalization on y_test and deal with outliers. (smaller C = less error)

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) 3rd-order Polynomial Kernel
   (c) Radial Basis Function Kernel with σ = 1
   (d) Sigmoidal Kernel with σ = 1

```
----------------------------------------
((a) 2nd-order Polynomial),
Total training error: 0.000, total testing error: 0.030

Class Iris-setosa:
training error: 0.000, testing error: 0.000,
support vector indices: [39, 15, 28]

Class Iris-versicolor:
training error: 0.000, testing error: 0.044,
support vector indices: [15, 28, 75, 83, 88, 90, 48, 52, 56]

Class Iris-virginica:
training error: 0.000, testing error: 0.044,
support vector indices: [48, 56, 75, 83, 90]
```

```
----------------------------------------
((b) 3rd-order Polynomial),
Total training error: 0.000, total testing error: 0.030

Class Iris-setosa:
training error: 0.000, testing error: 0.000,
support vector indices: [39, 15, 28]

Class Iris-versicolor:
training error: 0.000, testing error: 0.044,
support vector indices: [28, 75, 83, 88, 90, 48, 52, 56]

Class Iris-virginica:
training error: 0.000, testing error: 0.044,
support vector indices: [46, 48, 56, 75, 83, 88, 90]
```

```
----------------------------------------
((c) RBF with σ=1),
Total training error: 0.000, total testing error: 0.030

Class Iris-setosa:
training error: 0.000, testing error: 0.000,
support vector indices: [39, 43, 51, 53, 58, 70, 75, 77, 81, 82, 83, 84, 90, 92, 101, 103, 10, 11, 12, 15, 21, 28, 30]

Class Iris-versicolor:
training error: 0.000, testing error: 0.044,
support vector indices: [10, 11, 12, 15, 28, 30, 75, 81, 82, 83, 88, 90, 104, 39, 46, 48, 52, 56]

Class Iris-virginica:
training error: 0.000, testing error: 0.044,
support vector indices: [10, 11, 12, 15, 21, 28, 30, 39, 46, 48, 52, 56, 75, 81, 82, 83, 88, 90, 104]
```

```
----------------------------------------
((d) Sigmoidal with σ=1),
Total training error: 0.333, total testing error: 0.333

Class Iris-setosa:
training error: 0.333, testing error: 0.333,
support vector indices: [70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94,
95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
2, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34]

Class Iris-versicolor:
training error: 0.333, testing error: 0.333,
support vector indices: [70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94,
95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54,
55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69]

Class Iris-virginica:
training error: 0.333, testing error: 0.333,
support vector indices: [35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59,
60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 9
1, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104]
```

We also observed higher order poly-kernels (2nd, 3rd) and RBF kernel performed better than linear kernels (lower testing error in most cases).

=> It is possible that using non-linear kernels to set more complex boundaries is more suitable for achieving higher accuracy classification, particularly for identifying 'iris-versicolor' from other classes