DDA3020 Homework 2

Due date: Nov 13, 2023

Instructions

- The **deadline** is 23:59, Nov 13, 2023.
- The weight of this assignment in the final grade is 15%.
- **Electronic submission**: Turn in solutions electronically via Blackboard. Be sure to submit your homework as a single file. Please name your solution file as *A1_studentID_name*
- Note that **late submissions** will result in discounted scores: 0-24 hours → 80%, 24-120 hours → 50%, 120 or more hours → 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.
- If you have any questions concerning this homework, feel free to reach out to TA. For Written parts, you may refer to TA Fanzeng Xia (223040232@link.cuhk.edu.cn). For Programming parts, you may refer to TA Xiaozhuang Song (xiaozhuangsong1@link.cuhk.edu.cn). You're also welcome to physically visit them during their office hours with your questions.

1 Written Problems (50 points)

1. Exploring Asymmetric Regularization in 2D Logistic Regression (Inspired by Exercise 8.7 from Murphy's book) (15 points)

(1) Consider the data in Figure 1, where we fit the model

$$p(y = 1 \mid \mathbf{x}, \mathbf{w}) = \sigma\left(2w_0 + w_1x_1 + w_2x_2\right).$$

Suppose we fit the model by maximum likelihood, i.e.,

$$J(\mathbf{w}) = -\ell\left(\mathbf{w}, \mathcal{D}_{\text{train}}\right),$$

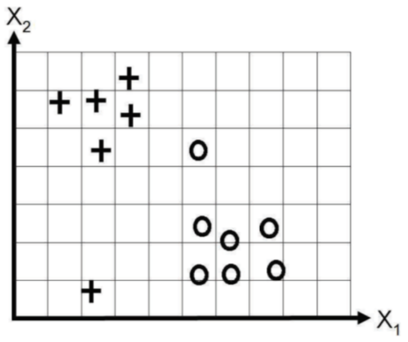

Figure 1: Data for logistic regression question

where $\ell(\mathbf{w}, \mathcal{D}_{\text{train}})$ is the log likelihood on the training set. Sketch a possible decision boundary corresponding to $\hat{\mathbf{w}}$. (Copy the figure first (a rough sketch is enough), and then superimpose your answer on your copy, since you will need multiple versions of this figure). Is your answer (decision boundary) unique? How many classification errors does your method make on the training set?

(2) Now suppose we regularize only the w_0 parameter, i.e., we minimize

$$J_0(\mathbf{w}) = -\ell(\mathbf{w}, \mathcal{D}_{\text{train}}) + \lambda w_0^2.$$

Suppose λ is a very large number, so we regularize w_0 all the way to 0 , but all other parameters are unregularized. Sketch a possible decision boundary. How many classification errors does your method make on the training set? Hint: consider the behavior of simple linear regression, $2w_0 + w_1x_1 + w_2x_2$ when $x_1 = x_2 = 0$.

(3) Now suppose we heavily regularize only the w_1 parameter, i.e., we minimize

$$J_1(\mathbf{w}) = -\ell(\mathbf{w}, \mathcal{D}_{\text{train}}) + \lambda|w_1|$$

Sketch a possible decision boundary. How many classification errors does your method make on the training set?

(4) Now suppose we heavily regularize only the w_2 parameter. Sketch a possible decision boundary. How many classification errors does your method make on the training set?

2. Manual Construction of a Non-Linear SVM Classifier (Inspired by Exercise 14.1 in Murphy's book) (15 points)

Consider a dataset with two points in 1D: $(x_1 = -1, y_1 = -1)$ and $(x_2 = 2, y_2 = 1)$. We will transform each point into 3D using the mapping $\phi(x) = \left[1, x, x^2\right]^\top$. This is analogous to employing a quadratic kernel. The optimal margin classifier is described by:

$$\begin{gathered} \min \|\mathbf{w}\|^2 \quad \text{subject to} \\ y_1\left(\mathbf{w}^T\phi(\mathbf{x}_1) + w_0\right) \geq 1 \\ y_2\left(\mathbf{w}^T\phi(\mathbf{x}_2) + w_0\right) \geq 1. \end{gathered}$$

(1) Propose a vector that could be perpendicular to the optimal orientation of $\mathbf{w}$ in the transformed space.

(2) Calculate the margin accomplished with your suggested w. Remember, the margin reflects the distance of each support vector to the decision boundary. For guidance, consider the spatial relationship between two points and a plane dividing them.

(3) Deduce the exact $\mathbf{w}$ by leveraging the principle that the margin equates to $1/\|\mathbf{w}\|$.

3. Exploring Advanced Concepts in SVM Classification (10 points)

Given a data set:

$$\text{Class - 1}: \begin{bmatrix} (1 & 0) \\ (0 & 1) \\ (-1 & 0) \\ (0 & -1) \end{bmatrix} \quad \text{Class } +1: \begin{bmatrix} (2 & 0) \\ (0 & 2) \\ (-2 & 0) \\ (0 & -2) \end{bmatrix}$$

2

(1) Can you find a svm linear classifier for this data set? If not, how to find a nonlinear classifier?

(2) Explain the concept of the RBF kernel and how it allows for non-linear separation in the feature space. Discuss its advantages over linear SVM, especially in relation to this specific dataset.

4. Analyzing Margin Width in Relation to Dual Variables in SVM (10 points)

In SVM, the margin width γ plays a crucial role in understanding the geometry of the decision boundary. Given the dual formulation of the SVM optimization problem, we seek to establish a direct relationship between γ and the Lagrange multipliers $\{\alpha_n\}$.

Consider a different perspective on the SVM's optimization, focusing on an alternate representation of the decision function involving a transformation ϕ applied to the input vectors $\mathbf{x}$, such that the dual problem now reads:

$$\begin{aligned} &\max_{\alpha} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m \phi(\mathbf{x}_n)^\top \phi(\mathbf{x}_m) \\ &\text{s.t.} \sum_{n=1}^{N} \alpha_n y_n = 0 \\ &\qquad \alpha_n \geq 0 \quad \forall n = 1, 2, \ldots, N \end{aligned}$$

Your task involves two primary objectives:

(1) Derive an expression for γ in terms of $\{\alpha_n\}$, assuming that the transformation ϕ influences

the inner product between feature vectors. Substantiate your derivation by exploring how ϕ affects the geometry of the decision boundary.

(2) Prove that the original relationship, $\frac{1}{\gamma^2} = \sum_{n=1}^{N} \alpha_n$, holds true even under the transformation ϕ. In your proof, emphasize any new mathematical insights or properties that emerge due to the involvement of ϕ.

Hint: You might need to revisit the concept of kernels and how they implicitly apply transformations on input vectors, influencing the SVM's decision function.

2 Programming (50 points + 12 points)

2.1 Logistic Regression Implementation: Gradient Descent (8 points)

Task Description In this task, your goal is to complete the gradient descent part of the logistic regression implementation. The provided code and instructions will be included in a Jupyter notebook. Follow the steps outlined in the notebook and the knowledge covered in the class to complete the task. Ensure that your code runs correctly, and can visualize the decision boundary using the provided plotting code.

Requirements Submit the modified Jupyter notebook containing your implemented code. Verify that the code runs without errors, and plots the decision boundary with the provided code.

2.2 Support Vector Machines Implementations (42 points)+ 12 points)

Task Description In this task, your goal is to conduct experiments on SVMs with different kernel functions and slack variables.

Datasets: You are provided with the training and testing datasets, comprising 120 training data and 30 testing data. These datasets are derived from the Iris dataset (`https://archive.ics.uci.edu/ml/datasets/iris`), which contains three classes (setosa, versicolor, and virginica) of 50 instances each, with each class representing a type of iris plant. Your task is to classify each iris plant into one of the three possible types.

What you should do: Utilize the SVM function from packages like the **scikit-learn (sklearn) package**, that offer various forms of SVM functions. It is recommended to use the `sklearn.svm.SVC()`[1] function.

[1]`https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`

Instructions This task involves training a SVM classifier for each iris type, that is, 3 SVMs for the 3 distinct iris types. This note is to ensure clarity and **prevents confusion with the concept of multi-class classification**. One method for handling the data is provided to facilitate your entry into the SVM training pipeline (Note that this is not a mandatory format to follow; it is just an approach provided for students who are not familiar with data processing. You are free to use any form to write your own pipeline to complete the task):

```
from sklearn.svm import SVC

y_train=pd.read_csv('y_train.csv')['target']
y_test=pd.read_csv('y_test.csv')['target']
# Assume you want to train a SVM for the "setosa" class, that is, label '0'
class_label = 0
y_train = y_train.apply(lambda x: 1 if x == class_label else -1)
y_test = y_train.apply(lambda x: 1 if x == class_label else -1)
# SVM training code ..
...
```

(12 Points Bonus: Implementing SVM from Scratch) This bonus challenge encourages students to delve into the details of SVM implementation, fostering a deeper understanding of the underlying algorithms and optimization processes. For those students who choose to implement the SVM functionality without relying on external packages such as scikit-learn, an additional bonus of 12 points will be awarded. Note that the total score, including the bonus, should not exceed the maximum score of this assignment only. One recommended approach for solving the SVM dual solution is the Sequential Minimal Optimization (SMO) method[2]. However, you are also free to explore other generic quadratic programming solvers to achieve this.

2.2.1 (10 points + 3 points)

Calculation using Standard SVM Model (Linear Kernel): Employ the standard SVM model with a linear kernel. Train your SVM on the provided 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 $C = 1e5$. Print out the results for each different class separately. The output format is as follows:

```
Q2.2.1 Calculation using Standard SVM Model:
```

[2]`https://pages.cs.wisc.edu/~dpage/cs760/SMOlecture.pdf`

```
setosa training error: xx, testing error: xx,
w_of_setosa: xx, b_of_setosa: xx,
support_vector_indices_of_setosa: xx,
versicolor training error: xx, testing error: xx,
w_of_versicolor: xx, b_of_versicolor: xx,
support_vector_indices_of_versicolor: xx,
virginica training error: xx, testing error: xx,
w_of_virginica: xx, b_of_virginica: xx,
support_vector_indices_of_virginica: xx,
```

2.2.2 (12 points + 3 points)

Calculate using SVM with Slack Variables (Linear Kernel) For each $C = 0.2 \times t$, where $t = 1, 2, \ldots, 5$, train your SVM on the provided 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, the indices of support vectors, and the slack variable ξ of support vectors (you may compute it as $\max(0, 1 - y \cdot f(\mathrm{X}))$. The output format is as follows:

```
Q2.2.2 Calculate using SVM with Slack Variables (C = 0.2 × t, where t = 1, 2, ..., 5):
C: 0.2,
setosa training error: xx, testing error: xx,
w_of_setosa: xx, b_of_setosa: xx,
support_vector_indices_of_setosa: xx,
slack_variable_of_setosa: xx,
versicolor training error: xx, testing error: xx,
w_of_versicolor: xx, b_of_versicolor: xx,
support_vector_indices_of_versicolor: xx,
slack_variable_of_versicolor: xx,
virginica training error: xx, testing error: xx,
w_of_virginica: xx, b_of_virginica: xx,
support_vector_indices_of_virginica: xx,
slack_variable_of_virginica: xx,
-----------------------------------------
C: 0.4 ..
```

Replace xx with the actual value in your output.

2.2.3 (20 points + 6 points)

Calculate using SVM with Kernel Functions and Slack Variables: Conduct experiments with different kernel functions for SVM, and set $C = 1$ for all cases. Calculate the classification error for both the training and testing datasets, the indices of support vectors, and the slack variable ξ of support vectors for each kernel type:

(a) 2nd-order Polynomial Kernel

(b) 3rd-order Polynomial Kernel

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

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

The output format for each file is as follows:

```
Q2.2.3 Calculate using SVM with Kernel Functions and Slack Variables:
(a)2nd-order Polynomial Kernel:
setosa training error: xx, testing error: xx,
support_vector_indices_of_setosa: xx,
slack_variable_of_setosa: xx,
versicolor training error: xx, testing error: xx,
support_vector_indices_of_versicolor: xx,
slack_variable_of_versicolor: xx,
virginica training error: xx, testing error: xx,
support_vector_indices_of_virginica: xx,
slack_variable_of_virginica: xx,
-----------------------------------------
(b) 3rd-order Polynomial Kernel:
<...results for (b)...>
-----------------------------------------
(c) Radial Basis Function Kernel with \(\sigma = 1\):
<...results for (c)...>
-----------------------------------------
(d) Sigmoidal Kernel with \(\sigma = 1\):
<...results for (d)...>
```

Replace xx with the actual value in your output.

Submission Instructions:

For Q2.1:

1. Submit your executable `A2_yourID_Q2.1.ipynb` Jupyter notebook.

For Q2.2, you can submit your code in two forms:

1. Place your executable code in a `A2_yourID_Q2.2.ipynb` Jupyter notebook and submit it. 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.
2. Submit your executable `.py` file. In this submission format, you can choose to place all three sub-questions in one Python file named `A2_yourID_Q2.py`; or you can choose to separate them into `A2_yourID_Q2.2.1.py`, `A2_yourID_Q2.2.2.py`, and `A2_yourID_Q2.2.3.py`.

Please note that regardless of the method chosen, you need to write clear comments and use appropriate function/variable names to indicate their corresponding question numbers (Especially if you decide to submit all 3 questions of Q2.2 in one file). Please note that excessively unreadable code may result in point deductions.