TECHIN 513 – Lab 5

Logistic Regression, Support Vector Machine

Lab Instructions

1. Please approach the course instructor or grader for assistance if you need help.
2. We highly recommend working in groups of **2 members**. When collaborating with students in other groups, please do not share code but only discuss the relevant concepts and implementation methods.
3. Please document your code well by **using appropriate comments, variable names, spacing, indentation, docstrings**, etc. Please refer to TECHIN 509 for more information.
4. The starter code is not binding on you. Feel free to modify it as you wish. Everything is fine so long as you are getting the correct results.
5. Please upload the .ipynb file to canvas. Use Markdown cells appropriately to answer the questions. Each team only needs to submit one report.
6. Please enter the names of all the team members in your Jupyter notebook so that you do not lose credit for your work.

Lab Submission Requirements

In your notebook, briefly describe your results and discuss the problems you encountered, the solutions that you came up with. Also, answer the questions asked in each section. Make sure to include the code of all tasks in code cells.

Lab Objectives

The goal of this lab is to apply logistic regression or support vector machine for classification tasks. We will also explore how to extend logistic regression and support vector machine for multi-class problems.

Preparation

You will need a few packages such as torch for tasks in this lab.

If you would like to perform this lab locally, setting up a virtual environment for this lab would reduce the chance of creating conflicts among dependencies. Please refer to TECHIN 509, or https://docs.python.org/3/library/venv.html, or https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html on how to setup virtual environment.

Please follow the steps (https://pytorch.org/get-started/locally) to install PyTorch locally.

Alternatively, Google Colab has the dependencies well-configured, and should be ready for your use.

Task 1: Predict Breast Cancer using Logistic Regression (6 points)

In this task, we will implement logistic regression to predict breast cancer.

1. Load data from breast_cancer.csv file. In this dataset, the data in column "diagnosis" is the target label, the rest columns (except id) are features.
2. Examine the data. Remove all invalid entries such as NaN from the dataset. Code for this step is given.
3. Use appropriate visualization to check the clean data. Based on the visualization, identify features that are likely correlated with each other. If two features are highly correlated with each other, they may contain redundant information, and should not be used simultaneously for logistic regression.
4. Keep one feature from the correlated group and drop the rest. Justify your choice(s).
5. Fix random seed 42. Shuffle and split the preprocesed data. Use 80% for training and 20% for testing.
6. Build a logistic regression model using the training data.
7. Evaluate your model over the testing dataset. Pick appropriate metrics, and explain whether the model has good/bad performance.

Task 2: Digit Recognition using SVM (6 points)

In this task, we will use SVM to recognize handwritten digits from 0 to 9.

1. Load the MNIST dataset. Some sample code using pytorch to load MNIST is given. You can either use the sample code after installing required dependencies, or use other sources to load the dataset. If you cannot complete this step, the MNIST dataset is also provided in the folder named "data". Directly run your code from the cell that defines X_train, y_train.
2. Rescale the pixel values of all data such that the pixel values fall within 0 to 1. Code for this step is given.
3. Complete the code to build an SVM model with a linear kernel. Evaluate the model performance over testing data.
4. Complete the code to build another SVM model with a non-linear kernel. Evaluate the model performance over testing data.
5. Compare the performance of these two models. Which model would you prefer? Justify your choice.

Discussion (2 points)

We discussed how to use SVM to separate two classes in lecture. Handwritten digits, however, contain ten classes. Read the document from this link:

https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/

SVM uses OvO for multi-class classification. Explain how many binary classifiers are required to classify ten classes.

Optional Discussion

Read through the following links on k-fold cross validation and hyperparameter search using GridSearchCV:

https://machinelearningmastery.com/k-fold-cross-validation/

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

https://www.mygreatlearning.com/blog/gridsearchcv/

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

Use GridSearchCV to tune hyperparameters to further improve the SVM (non-linear kernel) performance. Report the values of parameters found by your code. How much performance improve is observed?

Task 3: k-Nearest Neighbor Algorithm for Breast Cancer Prediction (6 points)

In this task, we will explore a simple machine learning algorithm named **k-nearest neighbor** (k-NN). k-NN is one of the simplest algorithms used for classification and regression tasks. It operates based on the **principle of similarity**: data examples that are close in feature space are likely to have similar labels or values, which turns out to be naïve yet effective in various applications.

k-NN is a non-parametric learning algorithm. That is, k-NN does not have a specific training phase to learn parameters θ as we have seen in linear/logistic regression and SVM. Instead, it simply stores the training data.

k-NN follows the steps below during the inference phase:

1. For a new data example, calculate the distance between this example and all examples in the training data.
2. Identify the k nearest neighbors (examples with the smallest distances). Here, k can be tuned by users, depending on the application.
3. For **classification tasks**:
    - Assign the label based on the majority class among the k neighbors.
4. For **regression tasks**:
    - Assign the average value of the k neighbors.

Given the background on k-NN, let us implement for human activity recognition. We will use Euclidean distance as the distance measure. For two data examples x^i, x^j of length n, their distance is computed as $dist = \sqrt{\sum_{k=1}^{n}(x_k^i - x_k^j)^2}$.

1. Implement k-NN algorithm as a function. Clearly document the function, e.g., input and output of the function.
2. Load the preprocessed features from Task 1. (Hint: To simplify the calculation, you may want to convert training and testing dataset from DataFrame to ndarrays).
3. Perform k-NN with k = 5, and evaluate the classification accuracy.
4. Repeat Step 2 with k = 3, 5, 7, 9. Use appropriate visualization to present how classification performance (your choice of performance metric) varies with k. Discuss which k value you would choose to optimize the performance. Justify your choice using the result.