LAB 2: SUPERVISED LEARNING – CLASSIFICATION

(Duration: 2 sessions)

(PART A: LOGISTIC REGRESSION)

Exercise 1: Implement Logistic Regression from Scratch

1. Implement the sigmoid function.
2. Write a class LogisticRegression with fit, predict, and predict_proba methods.
3. Train your model on a simple dataset (e.g., make_classification() or custom).
4. Compare your implementation with sklearn.linear_model.LogisticRegression.

Exercise 2 : Visualizing Logistic Regression Decision Boundaries

1. Train Logistic Regression on a 2D Dataset:
   a) Use a simple 2D dataset such as make_moons() or make_blobs().
   b) Train a logistic regression model on this dataset.
2. Plot the Data and Decision Boundary:
   a) Plot the training data.
   b) Visualize the decision boundary of the logistic regression model.
3. Plot the Probability Map:
   a) Use predict_proba() to plot the probability map of the classifier over the 2D space.
   b) Color each data point by its predicted probability of belonging to class 1. Use a color gradient to highlight the model's confidence across the 2D space.
4. Effect of Regularization:
   a) Train logistic regression with L2 regularization.
   b) Visualize how the decision boundary changes with different values of regularization strength λ.
   c) Discuss how regularization impacts the decision boundary.

Exercise 3 : Multiclass Logistic Regression with Softmax

1. Implement Multiclass Logistic Regression:
   a) Using gradient descent, implement softmax regression from scratch.
   b) Apply the model to the Iris dataset which has 3 classes.
2. Compare with sklearn's Multinomial Logistic Regression:
   a) Train a sklearn.linear_model.LogisticRegression with multi_class='multinomial' on the Iris dataset.
   b) Compare the predictions from your softmax implementation and sklearn's multinomial logistic regression.
   c) Discuss any differences in implementation and performance.

Exercise 4 : Logistic Regression with Polynomial Features for Non-linear Data

1. Generate a Non-linear Dataset (Use make_circles() to generate a non-linearly separable dataset)
2. Apply Polynomial Features:
   a) Use PolynomialFeatures to transform the data into a higher-dimensional space (quadratic terms).
   b) Train a logistic regression model on the transformed features.
3. Visualize the Decision Boundary:
   a) Plot the transformed data and the decision boundary of the logistic regression model after applying polynomial features.
   b) Discuss how the transformation allows the model to handle non-linearly separable data.

(Part B: Support Vector Machines)

Exercise 5 : Hard-Margin SVM Implementation (Linearly Separable Case)

1. Generate a linearly separable dataset (e.g., make_blobs).
2. Formulate the primal problem:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y^{(i)}(\mathbf{w}^\top x^{(i)} + b) \geq 1$$

3. Use an optimization package (cvxopt, scipy.optimize) to solve.
4. Visualize the decision boundary, support vectors, and margin.

Exercise 6: Soft-Margin SVM (Non-Separable Data)

1. Create the Dataset
   a) Use make_classification to generate a 2D dataset with two overlapping classes. (Use flip_y=0.1 and class_sep=0.8 to add overlap)
   b) Plot the data to check that it's not linearly separable.
2. Train a Soft-Margin SVM
   a) Use SVC(kernel='linear', C=1) to train a linear SVM.
   b) Plot the decision boundary and highlight the support vectors.

      (What do you notice about the support vectors and the margin?)
3. Change the Regularization Parameter C
   a) Train the SVM with different values of C: 0.01, 0.1, 1, 10, and 100.
   b) Plot the decision boundaries for each case.

      (Observe how the margin and number of misclassified points change.)
4. Evaluate the Model
   a) For each value of C, calculate the accuracy on the training set.
   b) Which value of C gives a good balance between accuracy and margin width?
   c) What happens to the margin width when C increases?
   d) If your data contains noise, would you prefer a large or small value for C? Why?
   e) How can cross-validation help you choose the best value for C?

Exercise 7: Exploring Non-Linear SVM and Kernel Functions

1. Start with a Linearly Separable Dataset
   a) Generate a simple linearly separable dataset using make_classification() or manual points.
   b) Train a linear SVM and plot the decision boundary.
   c) Calculate and display the training accuracy
   d) Observe the margin and explain the classification results.
2. Try a Non-Linearly Separable Dataset
   a) Generate a non-linear dataset using make_moons() or the XOR dataset.
   b) Train a linear SVM and visualize the decision boundary.
   c) Explain why the model fails to separate the classes.
3. Introduce and Apply the Polynomial Kernel
   a) Train an SVM with a polynomial kernel on the previous non-linear dataset.
   b) Experiment with different degrees d.
   c) Plot and discuss how the decision boundary changes with increasing d.
   d) Calculate and display the training accuracy
   e) Compare results with the linear kernel.
4. Explore the RBF Kernel
   a) Train an SVM using the RBF kernel on the same dataset.
   b) Tune the parameter γ and observe its impact on the decision boundary.
   c) Visualize the support vectors and the margin.
   d) Calculate and display the training accuracy
5. Compare All Kernels
   a) Train SVMs with linear, polynomial, and RBF kernels on the same dataset (e.g., make_moons() or XOR).
   b) Plot all decision boundaries.
   c) Discuss which kernel performed best and why.

Exercise 8: SVM on Real-World Data

1. Use the breast cancer dataset or Iris dataset.
2. Train with different kernels: linear, RBF, and polynomial.
3. Evaluate with accuracy, confusion matrix, and ROC (if binary).
4. Discuss when SVM performs best compared to other models.

Exercise 9: Multi-class SVM

1. Train SVC on the Iris dataset.
2. Understand One-vs-One (OvO) and One-vs-Rest (OvR) strategies.
3. Use multi_class='ovr' and multi_class='ovo' options.
4. Plot 2D decision regions after dimensionality reduction (e.g., PCA).