Machine Learning Basics

Instructors: Babak n. Arabi, Mohammadreza A. Dehaqani, Mostafa Tavassolipour

Hamed Soltani, Parisa Mohammadi

Fall 2025

Homework 3

Question 1: Answer These Questions

1. Why is the dual formulation of SVM usually preferred when kernels are used? Provide two reasons.
2. Explain the statement: "Only support vectors influence the final SVM classifier." Why do non-support vectors have zero dual coefficients?
3. Explain mathematically why the kernel trick allows SVMs to operate in infinite-dimensional spaces without computing the feature map explicitly.
4. Explain how the choice of kernel (linear, polynomial, RBF) affects the decision boundary, model capacity, and tendency to overfit.

Question 2: RBF kernel feature space intuition

Explain why the Gaussian (RBF) kernel $K(x, z) = \exp(-\|x-z\|^2/(2\sigma^2))$ can be interpreted as an inner product in an infinite-dimensional Hilbert space.

Question 3: SVM dual symmetric 4-point example

Consider the two-class data

$$x_1 = (1,1)^\top,\ y_1 = +1, \qquad x_2 = (1,-1)^\top,\ y_2 = +1,$$
$$x_3 = (-1,1)^\top,\ y_3 = -1, \qquad x_4 = (-1,-1)^\top,\ y_4 = -1.$$

Using a hard-margin SVM with linear kernel (i.e. feature map is identity), solve the dual problem to find the optimal dual variables α_i. Then compute the primal parameters w and b.

Question 4: Harder Kernel-Trick Question (Explicit cubic feature map)

Problem. Let $x = (x_1, x_2)^\top$ and $z = (z_1, z_2)^\top$ be vectors in $\mathbb{R}^2$. Consider the cubic polynomial kernel with bias

$$K(x, z) = (1 + x^\top z)^3 = \big(1 + x_1 z_1 + x_2 z_2\big)^3.$$

1. Expand $K(x, z)$ using the multinomial theorem and determine the dimensionality of the corresponding feature space.
2. Construct an explicit feature map $\phi : \mathbb{R}^2 \to \mathbb{R}^{10}$ (list the 10 components) such that
$$K(x, z) = \phi(x)^\top \phi(z).$$
You should include the correct combinatorial (square-root) scaling factors so the dot product of $\phi(x)$ and $\phi(z)$ equals $K(x, z)$.
3. Verify the equality $K(x, z) = \phi(x)^\top \phi(z)$ by showing the dot product reproduces the expanded polynomial.

Question 5: The Mechanics of Splitting (ID3)

Consider the following training dataset S for a binary classification problem where $Y \in \{0, 1\}$. There are two binary features, $X_1 \in \{A, B\}$ and $X_2 \in \{C, D\}$.

Sample	X_1	X_2	Y
1	A	C	0
2	A	C	0
3	A	D	1
4	B	C	0
5	B	D	1
6	B	D	1

1. **Root Entropy:** Calculate the Entropy of the dataset $H(Y)$ before any split. Use base-2 logarithms.
2. **Information Gain:** Calculate the Information Gain $Gain(S, X_1)$ and $Gain(S, X_2)$. Which feature would the ID3 algorithm choose for the root node?
3. **Tree Construction:** Draw/Describe the full decision tree produced by the ID3 algorithm. Does the tree achieve 0 training error?
4. **Comparison:** The lecture slides mention that trees are built "Top-down". If we used a "Bottom-up" approach, would the resulting tree necessarily be identical? Explain briefly.

Question 6: Optimization Geometry & Jensen's Inequality

The Entropy function $H(p)$ is often visualized as a concave parabola. Let $H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$.

1. **Proof of Concavity:** Compute the second derivative $\frac{\partial^2 H(p)}{\partial p^2}$ and prove that Entropy is strictly concave for $p \in (0, 1)$.
2. **Jensen's Inequality:** Jensen's Inequality states that for a strictly concave function f: $\sum \lambda_i f(x_i) \leq f(\sum \lambda_i x_i)$. Use this to prove that Information Gain is always non-negative ($Gain \geq 0$).
3. **Optimization Trap:** The ID3 algorithm uses "Hill Climbing". Describe a scenario where this greedy approach gets stuck in a local optimum that is worse than the global optimum.

Question 7: The Failure of Greedy Search (XOR)

Your lecture slides state that decision trees can express "any boolean function" but usually prefer simpler trees. Consider the classic XOR problem with 4 training examples:

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

1. **The Greedy Trap:** Compute $Gain(S, X_1)$ and $Gain(S, X_2)$. Based on these values, explain why a standard ID3 algorithm might fail to start building the tree or pick a random feature.
2. **Existence:** Draw/Describe a decision tree of Depth 2 that perfectly classifies this data.
3. **Inductive Bias:** Mitchell defines the Inductive Bias of ID3 as a "Preference Bias" rather than a "Restriction Bias".
   (a) Define "Preference Bias".
   (b) Explain why the failure in part (1) illustrates that ID3 does not search the complete hypothesis space (like BFS-ID3), but specifically prefers trees with high Information Gain near the root.

Question 8: From Data to Diagnosis: Diabetes Classification Using SVM (Programming Question)

Objective

In this assignment, you will build a complete classification pipeline using the Support Vector Machine (SVM) algorithm. You will apply different SVM kernels, perform preprocessing, parameter tuning, and analyze the results through visualization and performance metrics.

The dataset used is the `Pima Indians Diabetes Dataset`, which includes 768 samples, 8 features, and a binary target variable indicating whether a patient has diabetes.

Part 1: Data Preprocessing and Exploration

1) Load the dataset and separate features and labels.
2) Check if there are unrealistic zero values for the following features: `Glucose, BloodPressure, BMI, SkinThickness, Insulin` because zero values in these features are not medically meaningful.
3) Replace these zero values using one of the missing value handling Methods.
4) Split the dataset into training (70%) and testing (30%).
5) Apply feature scaling.

Question: Why is feature scaling important when using SVM?

Part 2: Linear SVM Classification

1) Train an SVM classifier using a `linear` kernel.
2) Evaluate the model on the test set and compute:
   - Accuracy
   - Confusion Matrix
3) Compare the performance before and after feature scaling.

Question: Does normalization significantly affect the performance of the model? Why?

Part 3: Comparison of Different Kernels

Train SVM classifiers using the following kernels:

1. Linear
2. RBF
3. Polynomial

For each model:

- Compute Accuracy, Precision, Recall, and F1-score.
- Plot and analyze the confusion matrix.

Task: Create a comparison table of the three kernels and discuss which one performs best and why.

Part 4: Hyperparameter Tuning

Using the RBF kernel, tune the parameters using `GridSearchCV`.
The grid is defined as:

$$C \in \{0.1, 1, 10, 100\}$$
$$\gamma \in \{0.001, 0.01, 0.1, 1\}$$

1) Find the optimal values for C and γ.

2) Report the best accuracy.

Conceptual Question: What is the effect of very large values of C on the decision boundary?

Part 5: Dimensionality Reduction and Visualization

1) Apply PCA to reduce the dataset from 8 dimensions to 2 dimensions.
2) Plot the 2D data points using different colors for each class.
3) Train an SVM (RBF kernel) on the reduced dataset.
4) Visualize the decision boundary in 2D space.

Question: Does the dimensionality reduction affect the classification accuracy? Why or why not?

Part 6: Class Imbalance Analysis

1) Compute the number of samples in each class.
2) Check if the dataset is imbalanced.
3) Apply SVM using `class_weight = 'balanced'`.
4) Compare the performance (especially recall) with the standard SVM.

Analysis Question: Did class balancing improve the performance for the minority class?

Question 9: The QP-SVM Primal Challenge: Optimizing and Visualizing the Soft-Margin Hyperplane SVM (Programming Question)

Problem Statement and Objective

The goal of this challenge is to implement and solve the **Primal Formulation** of a Soft-Margin Linear Support Vector Machine (SVM) using the provided dataset, `Social_Network_Ads.csv`. Instead of using high-level machine learning APIs (like `LinearSVC` from Scikit-learn), the task requires transforming the problem into a **Quadratic Programming (QP)** problem and solving it directly using an optimization library such as `cvxpy` in Python.

Dataset and Variables

- **Features (X):** `Age` and `EstimatedSalary`.
- **Target (y):** `Purchased`.

Mathematical Formulation (The Primal SVM Problem) The task is to find the optimal weight vector $\mathbf{w}$, bias b, and slack variables ξ by minimizing the following objective function:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i \tag{9.1}$$

Subject to the constraints:

$$y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{for } i = 1,\ldots,n \tag{9.2}$$

$$\xi_i \geq 0 \quad \text{for } i = 1,\ldots,n \tag{9.3}$$

- $\mathbf{w} \in \mathbb{R}^d$: The vector of feature weights.
- $b \in \mathbb{R}$: The bias (intercept) term.
- $\xi_i \in \mathbb{R}$: The slack variable for the i-th sample.
- $C > 0$: The regularization hyperparameter controlling the trade-off between margin size and classification error penalty.

Implementation Steps and Deliverables

1. **Data Preprocessing:**
   - Select features ($\mathbf{X} = \{\text{Age, EstimatedSalary}\}$).
   - Convert the target labels $\mathbf{y}$ from $\{0, 1\}$ to $\{-\mathbf{1}, \mathbf{1}\}$.
   - Split the data and apply Feature Scaling (StandardScaler) using only the training data statistics.
2. **QP Formulation and Solution (using `cvxpy`):**
   - Set the regularization parameter C (e.g., $C = 1.0$).
   - Define the optimization variables $\mathbf{w}$, b, and ξ.
   - Implement and solve the QP problem to retrieve the optimal parameters $\mathbf{w}^*$ and b^*.
3. **Model Evaluation:**
   - Calculate the predictions on the scaled test set using the decision function: $y_{pred} = \text{sign}(\mathbf{w}^* \cdot \mathbf{x}_{test} + b^*)$.
   - Report the final Accuracy Score of the custom QP-SVM model.
4. **Final Deliverable: Visualizing the Decision Boundary**
   - Using the optimal parameters $\mathbf{w}^*$ and b^*, plot the decision boundary defined by $\mathbf{w}^* \cdot \mathbf{x} + b^* = 0$ on the 2D feature space of the scaled training data.
   - The plot must clearly show the two classes (e.g., different colors) and the resulting linear separation line.

Question 10: Project: Decision Tree Classifier (Programming Question)

The objective of this project is to implement a Decision Tree classifier from scratch and analyze its behavior on a synthetic dataset. You will implement the ID3 algorithm, handling both discrete and continuous features, and explore techniques to prevent overfitting.

1. The Dataset We will use the **Extended Jeeves "Play Tennis" Corpus**.

- **Download:** You can download the dataset file here: `jeeves_tennis.csv`.
- **Features:**
  - `Day_ID`: Unique identifier for each row.
  - `Outlook`: Discrete (Sunny, Overcast, Rain).
  - `Temperature`: Continuous (Real-valued, e.g., 22.5°C).
  - `Humidity`: Discrete (High, Normal, Low).
  - `Wind`: Discrete (Strong, Weak).
  - **Target:** `PlayTennis` (Yes/No).

1. **Data Loading & Pre-processing (Coding):**
   - Load the dataset using `pandas`.
   - **The High Cardinality Trap:** Implement a function to calculate the Information Gain for the `Day_ID` feature.
   - **Analysis:** Report the Information Gain for `Day_ID`. Explain why this value is high (mathematically) but why the feature is useless for generalization. Automatically drop this column from your dataframe before proceeding.
2. **Entropy & Information Gain Engine (Coding):** Implement the core mathematical functions that will drive your tree. Do not use `sklearn` for this part.
   - Implement `calculate_entropy(y)`: Computes $H(S)$ for a given array of labels.

- Implement `calculate_information_gain(data, feature, target)`: Computes the expected gain for splitting `data` on `feature`.
- **Verification:** Run your functions on the entire dataset. Which feature (`Outlook`, `Humidity`, or `Wind`) has the highest Information Gain at the root?

3. **Handling Continuous Features (Coding):** Standard ID3 handles discrete features. You must extend this to handle continuous values (specifically `Temperature`).
   - Implement a function `find_best_split(data, attribute)` that:
     (a) Sorts the data by the attribute.
     (b) Identifies all split points (midpoints between adjacent values with different labels).
     (c) Calculates Information Gain for every split point.
     (d) Returns the threshold t that maximizes Gain.
   - Apply this function to `Temperature`. What is the optimal threshold found by your algorithm?
4. **Recursive Tree Construction (Coding):** Implement the full ID3 algorithm class `DecisionTreeID3`.
   - Your 'fit()' method should recursively build the tree.
   - Your 'predict()' method should traverse the tree to classify new samples.
   - **Evaluation:** Split your dataset into **80% Training** and **20% Testing**. Train your model and report the Accuracy, Precision, and Recall on the test set.
5. **Overfitting & Pruning Analysis (Simulation):** Decision trees effectively memorize data. With a large dataset, full trees can become massive.
   - **Depth Analysis:** Train a tree on the full training set. Report the maximum depth and number of leaf nodes.
   - **Noise Simulation:** Artificially flip the labels of 10% of your training data (simulating noise). Retrain the tree. How does the tree size (depth/nodes) change compared to the clean data?
   - **Post-Pruning Experiment:** Implement a pruning function (e.g., Reduced Error Pruning). Vary the pruning strength (or validation threshold). Plot a graph of **Tree Size vs. Test Accuracy**. Discuss the trade-off where the model generalizes best.