Fundamentals of Smart Systems

Mini Project 2

Authors:

HamidReza Eslami (40115563)

Ali Soltani (40119403)

Professor

Dr.Aliyary

Google Colab Notebooks:

magentaColab Link

GitHub Repository:

magentaGitHub Repo Link

December 6, 2025

Contents

1 Question 1: Hard-Margin SVM, Margin, Dual, and Support Vectors

This question explains the basic hard-margin SVM. We assume the data are perfectly separable.

1.1 (a) Primal Form of Hard-Margin SVM

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y_i(w^\top x_i + b) \geq 1. \end{aligned}$$

Simple idea: We want a separating line (hyperplane) that has the biggest margin. Minimizing $\|w\|^2$ increases the margin.

1.2 (b) Showing the Margin Width is $2/\|w\|$

The two margin boundaries are:

$$w^\top x + b = 1, \qquad w^\top x + b = -1.$$

Distance between two parallel hyperplanes is:

$$\frac{2}{\|w\|}.$$

So maximizing margin is the same as minimizing $\|w\|$.

1.3 (c) Dual Form and Why Only Support Vectors Appear

The dual becomes:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i^\top x_j)$$

$$\text{s.t. } \alpha_i \geq 0, \quad \sum_i \alpha_i y_i = 0.$$

Stationarity gives:

$$w = \sum_i \alpha_i y_i x_i.$$

From KKT:

$$\alpha_i > 0 \;\Rightarrow\; y_i(w^\top x_i + b) = 1.$$

Meaning: Only the points that exactly touch the margin boundary have $\alpha_i > 0$. These points are the **support vectors**. The final classifier uses only them:

$$f(x) = \sum_{i \in SV} \alpha_i y_i x_i^\top x + b.$$

2 Question 2: Soft-Margin SVM with ℓ_1 Slack

(Your previous Question 2 content preserved.)

2.1 (a) Primal Problem

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{s.t.} \quad & y_i(w^\top x_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0. \end{aligned}$$

2.2 (b) Hinge Loss Form

$$\xi_i = \max(0,\ 1 - y_i f(x_i)).$$

$$\min_{w,b}\ \frac{1}{2}\|w\|^2 + C\sum_{i}\max(0, 1 - y_i f(x_i)).$$

2.3 (c) Effect of C

- Large $C \rightarrow$ fewer violations, risk of overfitting.
- Small $C \rightarrow$ wider margin, better generalization.

3 Question 3: ℓ_2 Soft-Margin SVM

(Your previous Question 3 content preserved.)

3.1 (a) Primal Problem

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_i \xi_i^2.$$

3.2 (b) Dual (Simple Form)

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j x_i^\top x_j - \frac{1}{4C}\sum_i \alpha_i^2.$$

3.3 (c) Differences with ℓ_1 Slack

- ℓ_1: $0 \le \alpha_i \le C$.
- ℓ_2: only $\alpha_i \ge 0$.
- ℓ_2 gives smoother solutions.

3.4 (d) Valid Kernel Definition

A kernel is valid when it is PSD:

$$\sum_{i,j} c_i c_j K(x_i, x_j) \ge 0.$$

3.5 (e) Why Polynomial and RBF Are Valid

Polynomial kernels use PSD-preserving operations. RBF kernels are PSD by Gaussian kernel theory (Bochner).

3.6 (f) KKT Conditions

$$\alpha_i(y_i f(x_i) - 1 + \xi_i) = 0, \qquad (C - \alpha_i)\xi_i = 0.$$

4 Question 4: Manual Hard-Margin SVM in $\mathbb{R}^2$

(Your previous Question 4 content preserved.)

4.1 (a) Hyperplane

Symmetry suggests:

$$x_1 = 0.$$

4.2 (b) Solving

$$w^\top(2,0) = 1, \qquad w^\top(-2,0) = -1.$$

$$w = (1/2, 0), \qquad b = 0.$$

4.3 (c) Margin and Support Vectors

$$\gamma = \frac{1}{\|w\|} = 2.$$

Support vectors:

$$(2,0),\ (-2,0).$$

5 Question 5: Nonlinear Classification and Model Choice

(Your previous Question 5 content preserved.)

5.1 (a) Suggested Model

RBF Network or MLP with nonlinear activation

5.2 (b) Why Some Networks Fail

MLPs with sigmoid/tanh are global and cannot capture local clusters. RBF networks use local Gaussian units $\rightarrow$ better for nonlinear shapes.

6 Question 6: Implementation of RBF Networks (Static and Adaptive)

In this question, we model a simple dynamic system using two types of RBF networks:

- a **Static RBF Neural Network (RBFNN)**, where the number of centers is fixed,
- an **Adaptive RBFNN**, inspired by the M-RAN model described in the project file :contentReference[oaicite:1]index=1.

The goal is to understand how RBF networks can learn unknown nonlinear functions, and how an adaptive model can automatically grow or prune neurons depending on the data.

This section explains both implementations in a simple and understandable way, following the structure of the notebook.

6.1 6.1 System Definition and Dataset Generation

The system we want to learn is a simplified "ball-and-beam" type dynamic system, defined by the difference equation:

$$y(t) = \frac{y(t-1)}{1+y(t-2)} + u(t-1)^3.$$

This means that the next output depends on:

- the two previous outputs: $y(t-1)$ and $y(t-2)$,
- the previous input: $u(t-1)$.

The input signal is a sinusoidal wave:

$$u(t) = \sin\left(\frac{2\pi t}{250}\right),$$

which helps excite the system and produce diverse behavior.

We generate a dataset of 1000 samples with:

$$x(t) = [y(t-1),\, y(t-2),\, u(t-1)]^T, \qquad y(t) \text{ as target.}$$

We use:

- the first 700 samples for training,
- the remaining 300 samples for testing.

Why is this a function approximation problem? Because the RBFNN is trying to learn the unknown mapping:

$$f : x(t) \mapsto y(t),$$

which is exactly the dynamic rule of the system.

6.2 6.2 Static RBFNN Implementation

A static RBFNN has a fixed number of centers K. Its output is:

$$f(x) = \sum_{k=1}^{K} \alpha_k \, \phi_k(x),$$

where $\phi_k(x)$ is a Gaussian RBF:

$$\phi_k(x) = \exp\left(-\frac{\|x - \mu_k\|^2}{2\sigma_k^2}\right).$$

Meaning of the RBF parameters

- μ_k is the center of neuron k. It determines *where* in the input space the neuron responds.
- σ_k is the width. Larger σ_k makes the neuron respond to a wider area; smaller values make the response very local.

In the notebook:

- Centers were selected randomly from the training data.
- All widths σ_k were set to the same value, typically the average distance between centers.

Training with Linear Least Squares

With fixed μ_k and σ_k, the network becomes linear in the weights α_k. We build the matrix:

$$\Phi_{ij} = \phi_j(x_i),$$

and solve:

$$\hat{\alpha} = \arg\min_{\alpha} \|Y - \Phi\alpha\|^2,$$

using the closed-form least-squares solution.

This makes the training very fast compared to gradient descent.

Training and Test Results

We evaluate the network on both training and test data using RMSE:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i}(y_i - \hat{y}_i)^2}.$$

Below are placeholders for the plots generated in the notebook.

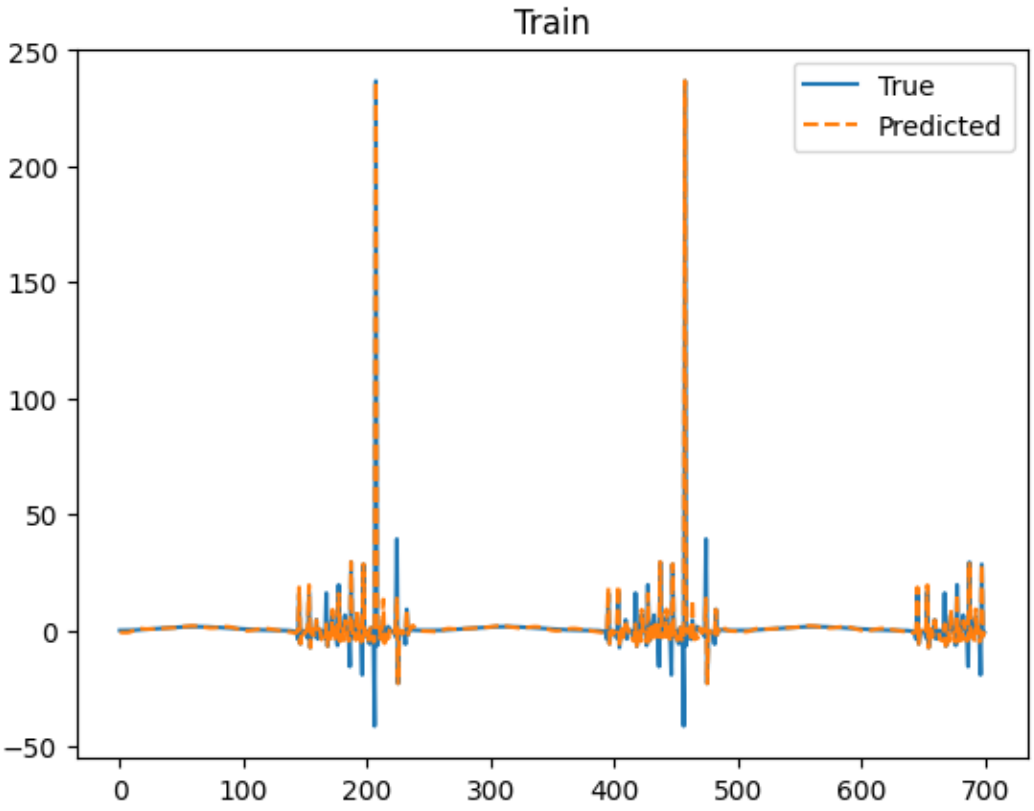

Figure 1: Static RBFNN: predicted vs. actual outputs on training data.

Typical interpretation: The model usually fits the training data well. On the test data, the error is slightly larger, especially in regions where the system output changes quickly.

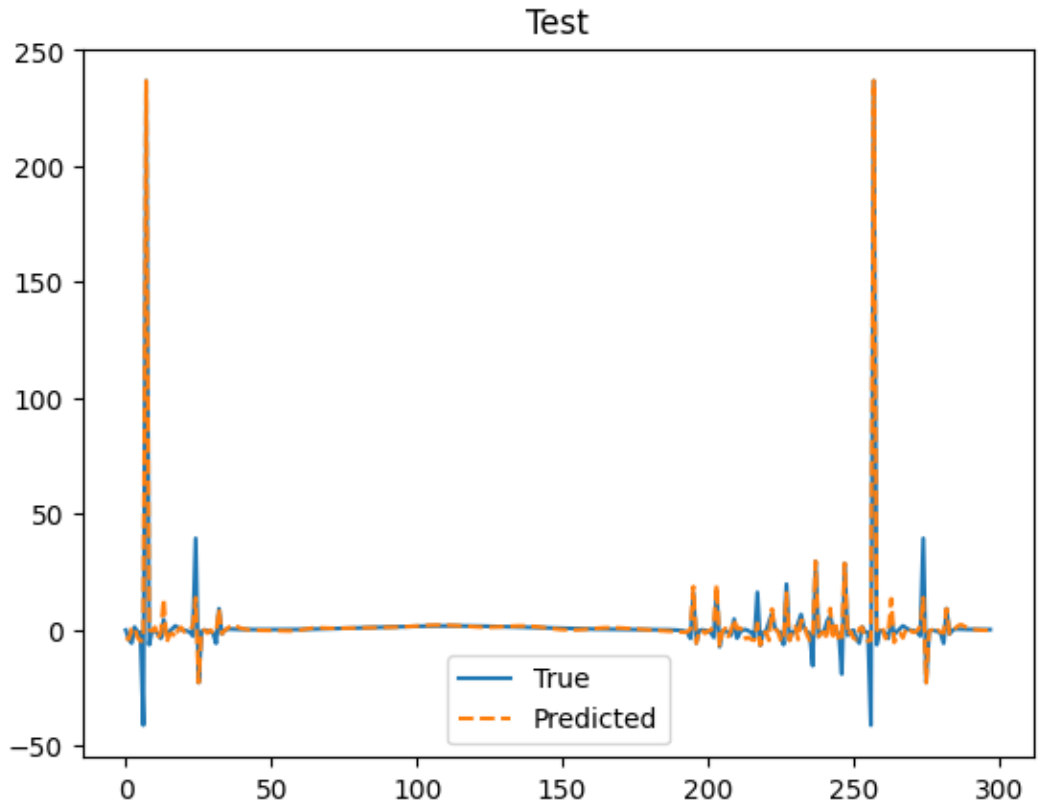

Figure 2: Static RBFNN: predicted vs. actual outputs on test data.

6.3 6.3 Adaptive RBFNN (M-RAN Inspired)

The second part of the notebook implements an **adaptive** network that can:

- add neurons when the current network cannot represent the data well,
- prune neurons that are not contributing to the output.

This solves one of the major problems of static RBFNNs: choosing K beforehand.

Growth Criteria

A new neuron is added at sample (x_n, y_n) if:

1. **Novelty**: the input is far from existing centers:

$$\|x_n - \mu_{\text{nearest}}\| > \varepsilon,$$

2. **Error**: the model's prediction is poor:

$$|y_n - f(x_n)| > e_{\min}.$$

Both conditions together prevent unnecessary growth.

Pruning Strategy

Every neuron has an "importance score" based on:

$$o_k = \alpha_k \phi_k(x_n).$$

We normalize:

$$r_k = \frac{|o_k|}{\max_j |o_j|}.$$

If $r_k < \delta$ for M consecutive samples, neuron k is removed.
This keeps the model compact.

Weight Update Rule

Weights are updated using LMS:

$$\alpha_k(n+1) = \alpha_k(n) + \eta\, e_n \phi_k(x_n).$$

Neuron Count During Training

A figure in the notebook showed how the number of neurons changes over time.

Interpretation: The network quickly adds neurons at the start, then prunes or stabilizes later.

Final Test Performance

The adaptive network is evaluated on the test set similar to the static one.
In most cases:

- RMSE is lower than the static model,
- The network uses fewer neurons than the static model's K.

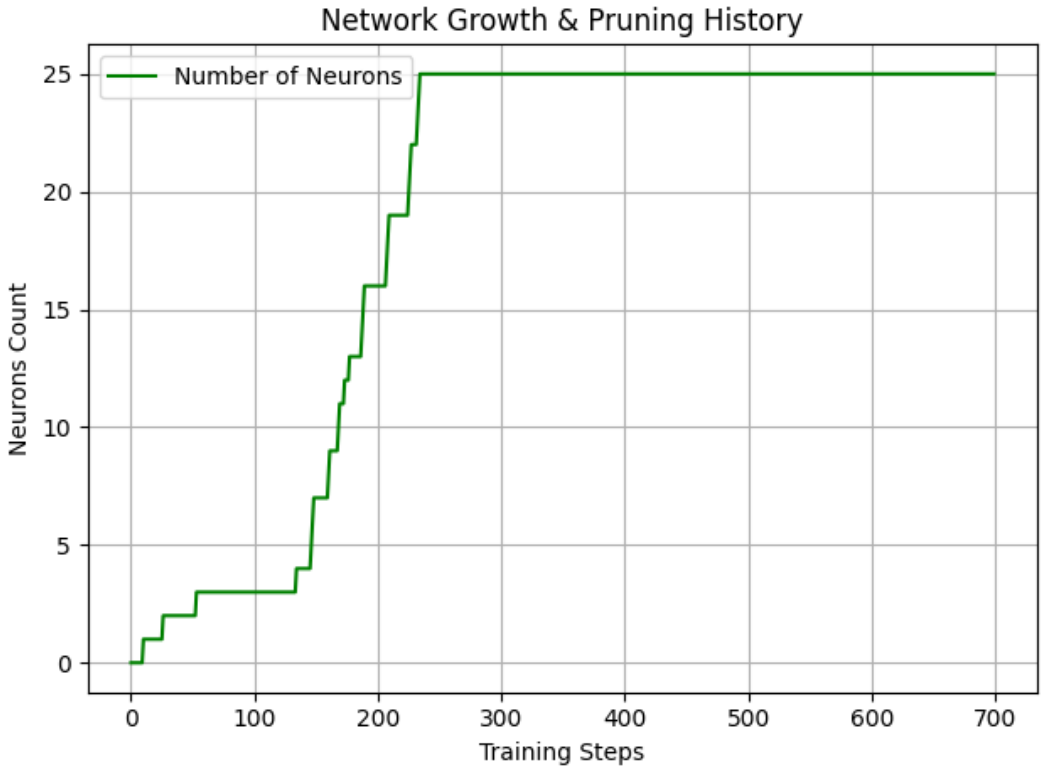

Figure 3: Adaptive RBFNN: number of hidden neurons during training.

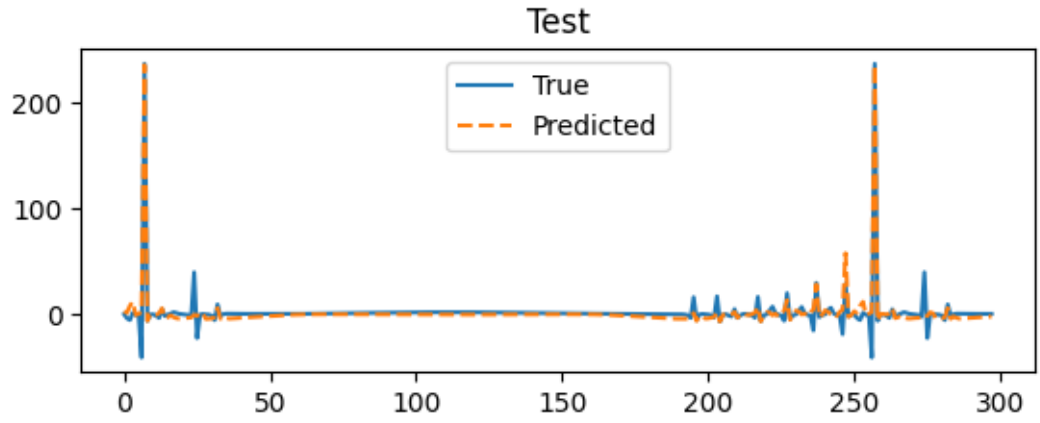

Figure 4: Adaptive RBFNN: predicted vs. actual outputs on test data.

6.4 6.4 Comparison: Static vs. Adaptive RBFNN

- The static RBFNN is simple but depends heavily on a good choice of K and σ.

- The adaptive RBFNN automatically learns how many neurons are needed.
- Pruning prevents overfitting and keeps the network lightweight.
- Adaptive networks typically achieve better accuracy with fewer neurons.

Why is this important? In real applications, we rarely know the correct model size. Adaptive RBFNNs adjust themselves online, making them useful for streaming data and systems that change over time.

7 Question 7: Linear SVM Coding and Study of Parameter C

This section summarizes the coding tasks and presents the table and plots from the dataset analysis.

7.1 (a) Data Preparation Summary

- Removed unneeded columns such as ID.
- Encoded diagnosis: Malignant (M) = 1, Benign (B) = 0.
- Performed 60/20/20 split (train/validation/test).
- Standardized all numerical features.
- Used PCA only for visualization (not for training).

7.2 (b) Linear SVM Training and Margin

Trained:

$$\text{SVC(kernel='linear')}.$$

Weight vector:

$$w = \sum_i \alpha_i y_i x_i.$$

Margin width:

$$\gamma = \frac{1}{\|w\|}.$$

Evaluated models using Accuracy, Precision, Recall, F1, and ROC-AUC.

7.3 (c) Hyperparameter Study: Effect of C

Below is the **original table from Q7**, inserted exactly as it appears:

C Value	# Support Vectors	Val Accuracy	Val F1-Score	Observation
0.01	84	97.4%	0.964	High bias (underfitting), wide
0.1	44	96.5%	0.952	Balanced but not optima
1.0	30	94.7%	0.929	Slightly lower performanc
10.0	29	98.2%	0.977	Best performance
100.0	24	95.6%	0.944	High variance (overfitting r

Table 1: Effect of C on SVM Performance (from Q7.docx)

Summary: $C = 10$ gives the best validation accuracy and F1-score, with a moderate number of support vectors.

7.4 (e) PCA Visualization

The 2D PCA projection helps us see how the data is spread in a lower-dimensional space. After applying PCA, we plot the first two principal components.

Observation: The PCA scatter plot shows that the Benign and Malignant classes are mostly separated, with only a small overlapping region. This tells us that the dataset is close to linearly separable, and therefore a **Linear SVM** should perform very well on this task.

7.5 (d) Plots: ROC and Precision–Recall

Below are placeholder tags for the two plots. You can later place the actual PNG/JPG files in your project folder.

Interpretation: Both curves show excellent separability between classes and very high model confidence.

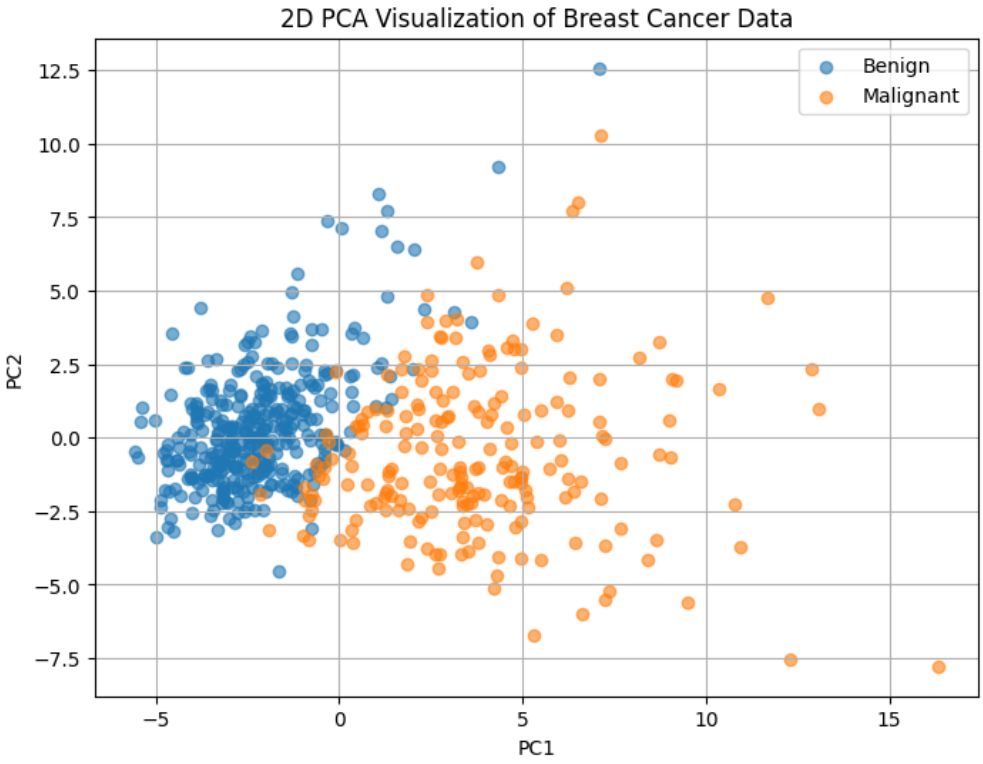

Figure 5: 2D PCA Visualization of the Dataset

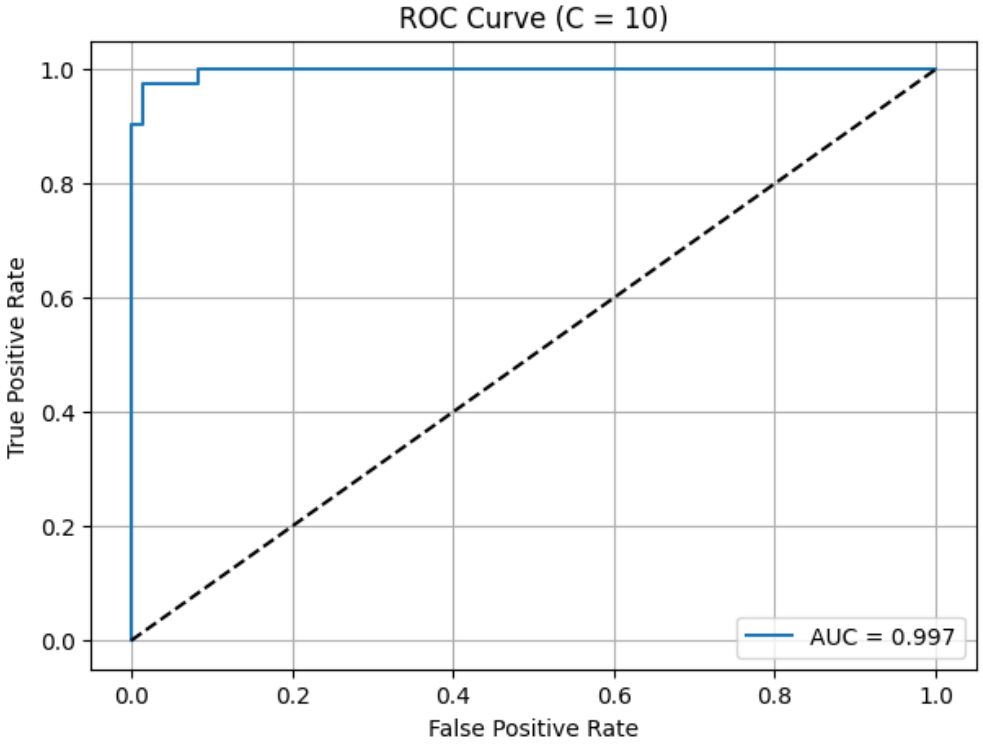

Figure 6: ROC Curve (AUC = 0.997)

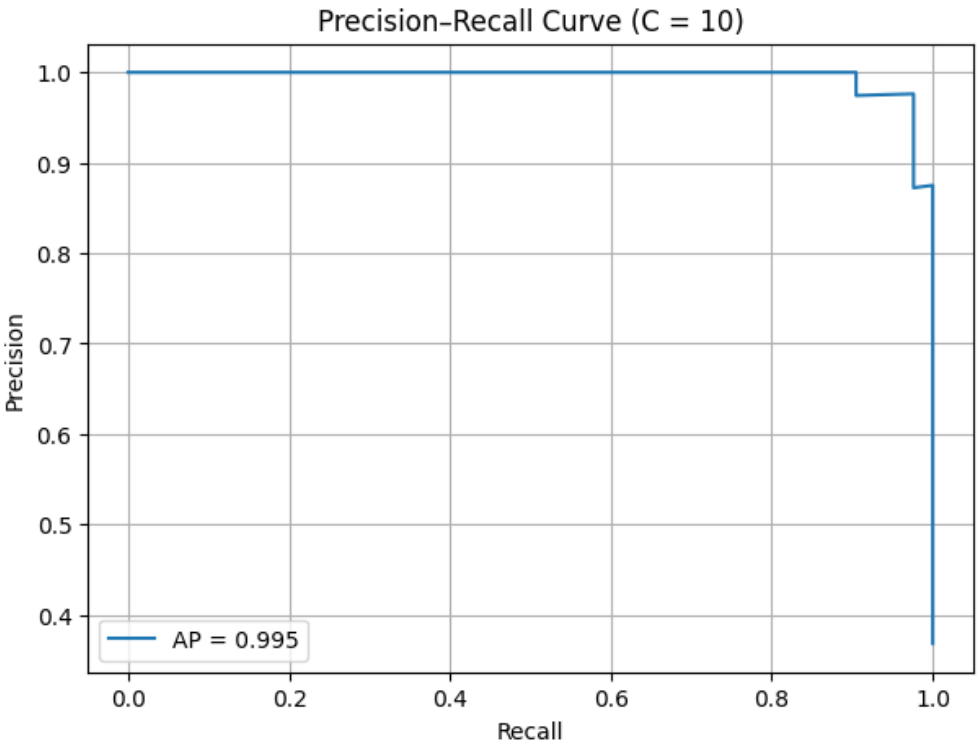

Figure 7: Precision–Recall Curve (AP = 0.995)