From Math to Machine: Building a Banknote Authenticator from Scratch

A technical case study implementing a Linear Support Vector Machine using only NumPy.

$$\min_{i \in \mathcal{C}} x_i = \lambda \|w\|^2 + \sum_{i=1} \max(0, 1 - y_i(w \cdot x_i + b))$$

The Mission: Deconstruct an Algorithm to Solve a Real Problem

The Problem

Counterfeit banknotes pose a significant threat to financial systems. Our challenge is to build a transparent, effective classifier to distinguish genuine notes from fakes.

The Objectives

- **Build:** Construct a robust machine learning model for banknote classification.
- **Implement:** Code the Linear SVM algorithm entirely from scratch to understand its mechanics.
- **Evaluate & Visualize:** Rigorously measure performance and create intuitive visualizations of the model's decision-making process.

The Raw Material: The Banknote Authentication Dataset

Source
UCI Machine Learning Repository

Samples
~1,300 Banknotes

Features
4 Numerical Inputs

Task
Binary Classification

Features from Wavelet Transforms

- **Variance:** `variance` of the transformed image
- **Skewness:** `skewness` of the transformed image
- **Kurtosis:** `kurtosis` of the transformed image
- **Entropy:** `entropy` of the image

Target Label Mapping

Original Class	Mapped to SVM	SVM Label
Class 1	→	+1 (Genuine)
Class 0		-1 (Counterfeit)

Data Preparation: Forging a Clean Foundation

1. Load & Split

Ingest CSV data into NumPy arrays (Features X, Labels y).

2. Map Labels

Convert target labels to +1 (Genuine) and -1 (Counterfeit) for SVM compatibility.

+1 -1

3. Standardize Features

Scale features to have a mean of 0 and a standard deviation of 1.

$$z = \frac{x - \mu}{\sigma}$$

4. Train-Test Split

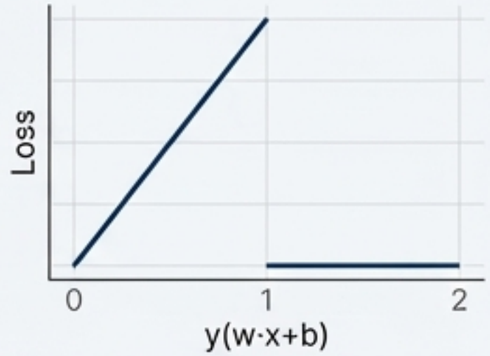
Partition the data with an 80/20 split for training and testing, done via simple indexing.

The Algorithm's Core: Deconstructing the SVM Objective Function

Penalizing Errors (Hinge Loss)

The cost for misclassifications and margin violations.

$$\max(0, 1 - y_i * (\boldsymbol{w} \cdot \boldsymbol{x}_i + b))$$



Preventing Overconfidence (L2 Regularization)

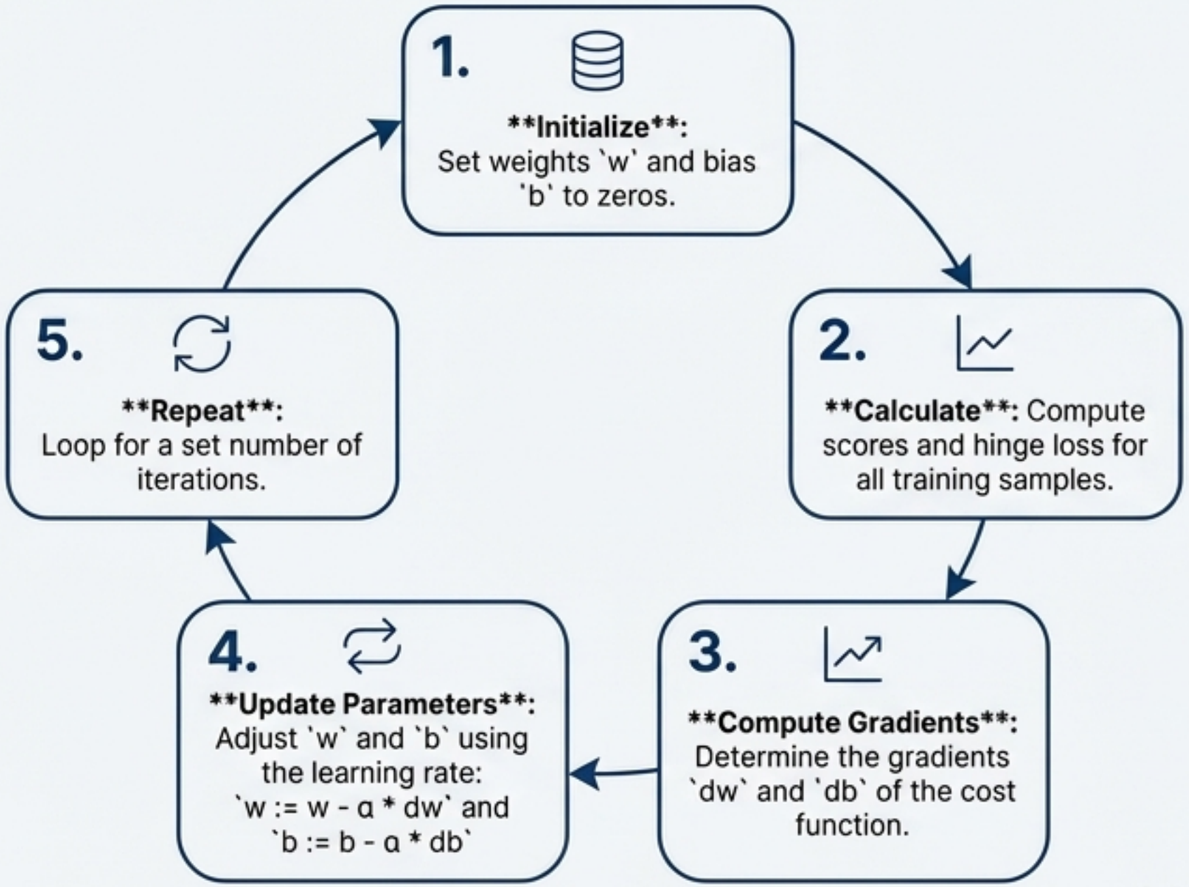
A penalty on large weights to create a simpler, more generalizable model.

$$\frac{\lambda}{2} * \|\boldsymbol{w}\|^2$$

Combined Cost Function

$$\text{Cost} = \frac{1}{N} * \sum \max(0, 1 - y_i * (\boldsymbol{w} \cdot \boldsymbol{x}_i + b)) + \frac{\lambda}{2} * \|\boldsymbol{w}\|^2$$

The Learning Engine: Training with Batch Gradient Descent



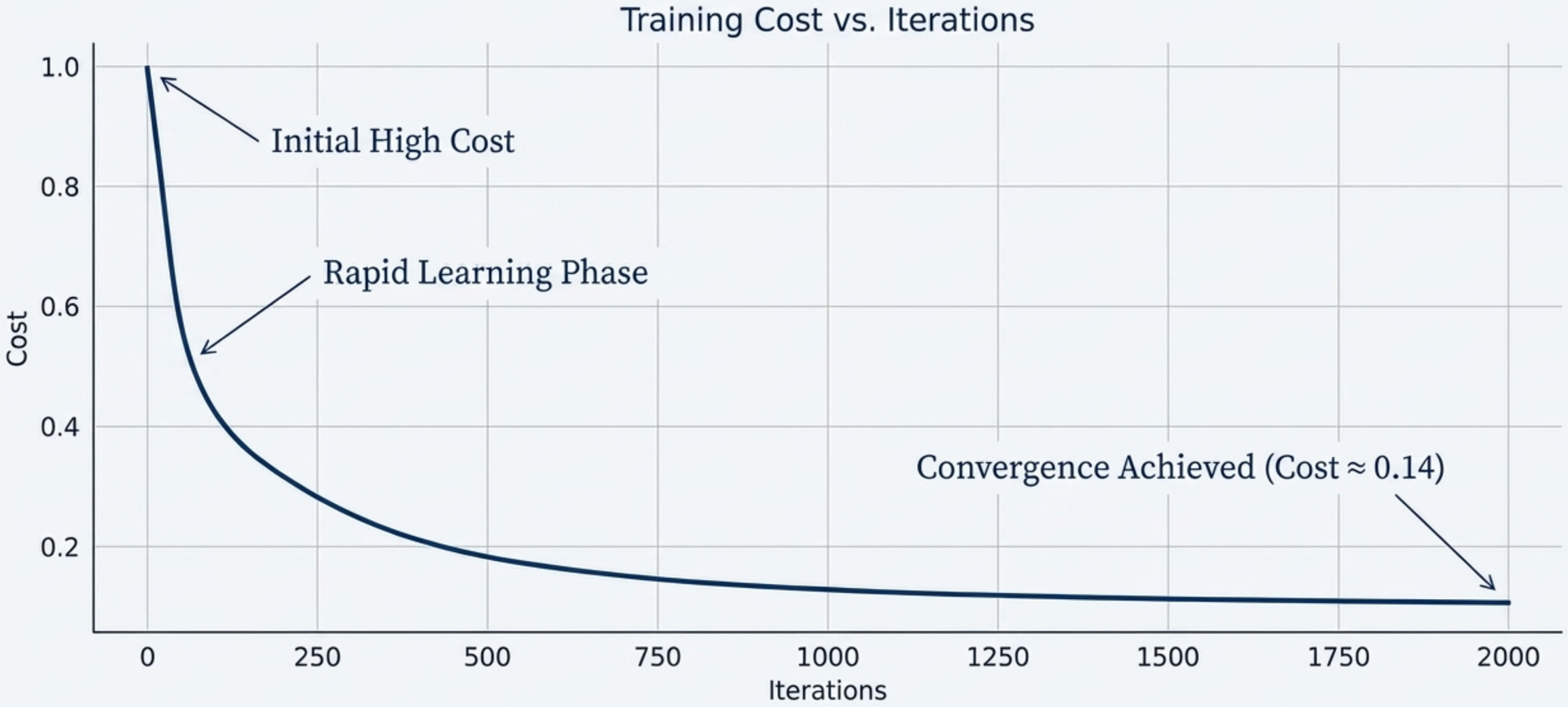
Experimental Settings

Learning Rate (α): `0.001`

Regularization (λ): `0.01`

Iterations: `2000`

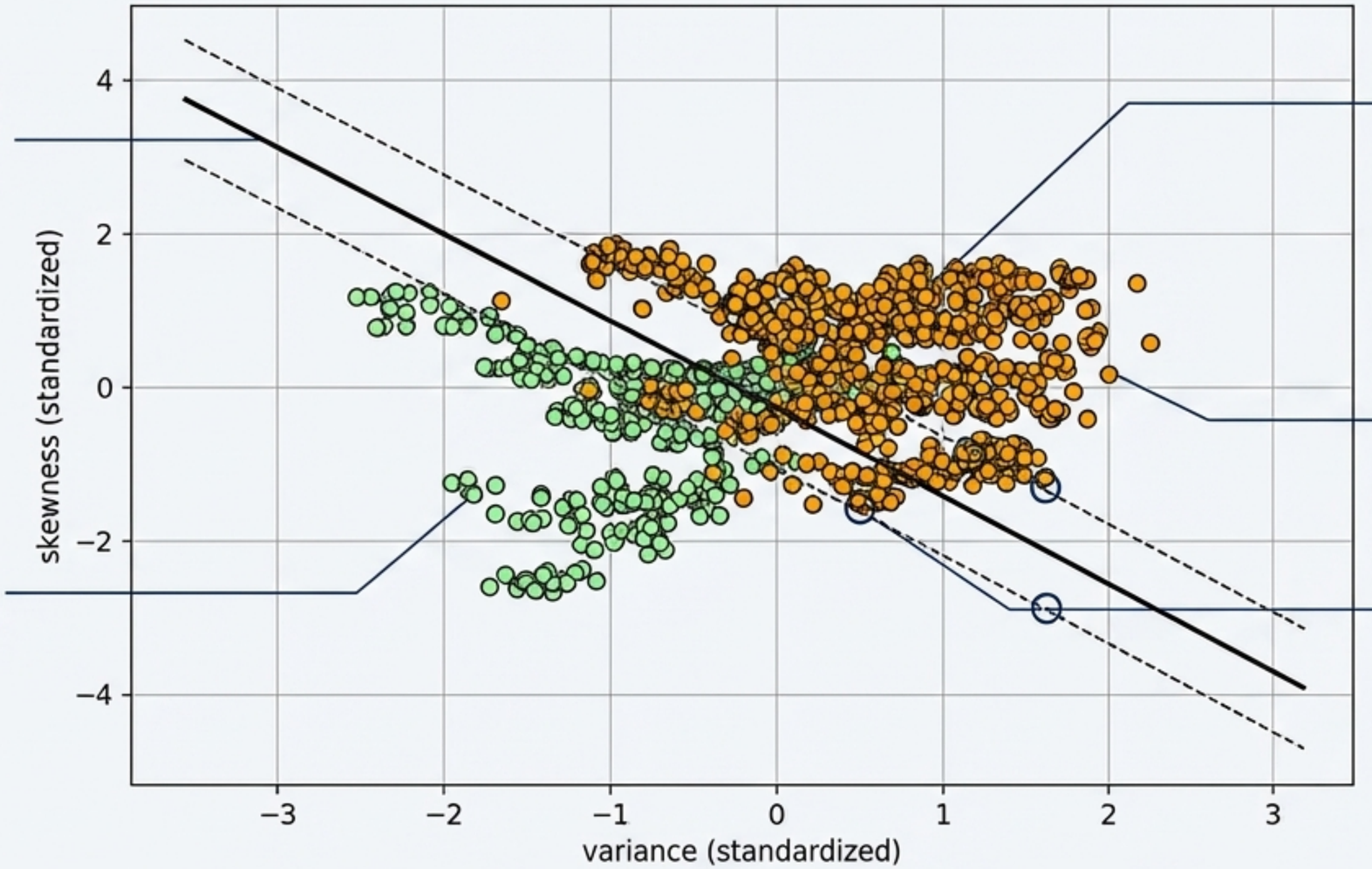
The Proof of Convergence: Watching the Model Learn



A Clear Line Between Genuine and Fake

Decision Boundary:
The model's dividing line where a note is classified as genuine or fake.

Class +1 (Genuine Banknotes)



Margins:
The 'buffer zone' the SVM maximizes to ensure a confident separation.

Class -1 (Counterfeit Banknotes)

Support Vectors:
The critical data points that define the boundary.

Visualization uses two of the four features for clarity: `variance` and `skewness`.

Performance Under the Microscope: The Final Metrics

Test Accuracy

99.27%

Precision

1.000

Training Accuracy

97.63%

Recall

0.993

F1 Score

0.996

Note: The custom confusion matrix function (`[273, 0], [2, 0]]`) has a minor printing bug affecting the TN value. The metrics above are calculated correctly from the underlying TP/FP/FN/TN values and are accurate.

Key Insights from the Build

- A from-scratch Linear SVM can achieve over 99% test accuracy on this real-world classification task.

- Proper feature scaling (standardization) and L2 regularization are essential for stable convergence and preventing overfitting.

- Visualizing the cost curve and decision boundary provides invaluable, intuitive insight into a model's behavior during and after training.

- Implementing an algorithm manually, rather than just importing a library, builds a deep and lasting understanding of its core mechanics.

The Horizon: Enhancements and Future Work

Current Implementation

- Linear Decision Boundary
- Batch Gradient Descent
- Manual Hyperparameters

Future Directions

- **Next Level: Exploring Non-Linearity:** Implement kernel SVMs (e.g., RBF) for more complex datasets.
- **Scaling Up: Advanced Optimization:** Explore faster optimizers for very large datasets.
- **Enhancing Robustness:** Integrate cross-validation and systematic hyperparameter tuning.
- **From Model to Tool:** Wrap the model in a simple API or CLI for practical application.

Appendix: A Glossary of Terms

SVM (Support Vector Machine)	A supervised algorithm that finds an optimal hyperplane to separate classes with the maximum possible margin.
Hinge Loss	A loss function used by SVMs that penalizes points inside the margin and those that are misclassified.
Regularization	A technique that penalizes large model weights to prevent overfitting and improve generalization.
Margin	The distance between the decision boundary and the closest data points (the support vectors) from each class.
Precision	Of all positive predictions, the proportion that were actually positive. `TP / (TP + FP)`
Recall	Of all actual positives, the proportion that were correctly identified. `TP / (TP + FN)`
F1 Score	The harmonic mean of Precision and Recall, providing a single score that balances both.