Semi-Supervised SVM vs Newton Universum Twin SVM

Saeed Khosravi

April 9, 2025

1 Introduction

In this project, our objective is to implement and analyze two machine learning models: the Semi-Supervised Support Vector Machine (S^3VM) [1] and the Newton-based Universum Twin Support Vector Machine (Newton-UTSVM) [2]. We will evaluate their performance on different datasets and compare their strengths and limitations. Finally, we propose a new approach that bridges these two methods: an unconstrained S^3VM combining the benefits of semi-supervised learning and Universum data utilization.

2 Semi-Supervised Support Vector Machine (Constrained S^3VM)

The constrained Semi-Supervised Support Vector Machine (S^3VM) approach, as introduced by [1], formulates the learning problem as a mixed-integer program (MIP) that explicitly models the unknown labels of unlabeled data through binary decision variables. This method fundamentally extends the standard SVM framework by incorporating additional constraints and optimization variables to handle unlabeled examples.

$$\begin{aligned} \min_{w,b,\eta,\xi,z,d} \quad & C\left[\sum_{i=1}^{\ell}\eta_i + \sum_{j=\ell+1}^{\ell+k}(\xi_j + z_j)\right] + \|w\| \\ \text{subject to} \quad & y_i(w\cdot x_i - b) + \eta_i \geq 1, \quad \eta_i \geq 0, \quad i = 1,\ldots,\ell \\ & w\cdot x_j - b + \xi_j + M(1-d_j) \geq 1, \quad \xi_j \geq 0, \quad j = \ell+1,\ldots,\ell+k \\ & -(w\cdot x_j - b) + z_j + Md_j \geq 1, \quad z_j \geq 0, \quad d_j \in \{0,1\} \end{aligned} \tag{1}$$

The objective function combines three key components to minimize sum of penalty term for misclassifications in the labeled data ($\sum_{i=1}^{\ell}\eta_i$), penalty term for inconsistencies in the unlabeled data ($\sum_{j=\ell+1}^{\ell+k}(\xi_j + z_j)$), and regularization term for controlling model complexity ($\|\mathbf{w}\|_1$).

```python
def objective(x):
    w = x[:n_features]
    eta = x[n_features+1 : n_features+1+n_labeled]
    xi = x[n_features+1+n_labeled : n_features+1+n_labeled+n_unlabeled]
    z = x[n_features+1+n_labeled+n_unlabeled : n_features+1+n_labeled+2*n_unlabeled]
    return (self.C*(np.sum(eta)+np.sum(xi + z))+np.sum(np.abs(w)))
```

The optimization problem is constrained by three types of conditions. The first constrained is for maintaining the standard SVM margin for labeled data, and the others are used for unlabeled data.

1. **Labeled data constraints** enforce the standard SVM margin requirement through

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) + \eta_i \geq 1, \quad \eta_i \geq 0 \quad \forall i \in \{1,\ldots,\ell\}, \tag{2}$$

   ensuring correct classification of known examples.

2. **Unlabeled data constraints** implement competing hypotheses via two parallel conditions:

$$\mathbf{w}^\top \mathbf{x}_j - b + \xi_j \geq 1 - M(1-d_j) \quad \text{(Class +1)} \tag{3}$$

$$-(\mathbf{w}^\top \mathbf{x}_j - b) + z_j \geq 1 - Md_j \quad \text{(Class -1)}, \tag{4}$$

   where the binary variable d_j activates only one constraint per point.

3. **Decision variables** enforce logical label assignment with

$$d_j \in \{0,1\}, \quad \xi_j, z_j \geq 0 \quad \forall j \in \{\ell+1,\ldots,\ell+k\}, \tag{5}$$

using a large constant M (10^5) to deactivate irrelevant constraints.

This joint optimization simultaneously learns the decision boundary while inferring labels for unlabeled data.

```
constraints = [
    {
     'type': 'ineq',
     'fun': lambda x: y_labeled.flatten() *
            (X_labeled @ x[:n_features] + x[n_features]) +
            x[n_features+1:n_features+1+n_labeled] - 1
    },

    {
     'type': 'ineq',
     'fun': lambda x: (X_unlabeled @ x[:n_features] - x[n_features] + x[n_features+1+n_labeled:n_features+
    1+n_labeled+n_unlabeled] + self.M*(1 - x[-n_unlabeled:])) - 1
    },

    {
        'type': 'ineq',
        'fun': lambda x: (-(X_unlabeled @ x[:n_features] - x[n_features]) + x[n_features+1+n_labeled+
    n_unlabeled:n_features+1+n_labeled+2*n_unlabeled] + self.M*x[-n_unlabeled:]) - 1
    }
]
```

For optimization part, we selected SLSQP because it perfectly handles mixed constraints (bounds + nonlinear inequalities) while balancing speed and accuracy for moderate-scale problems. Its ability to enforce strict constraint satisfaction makes it ideal for our semi-SVM's competing hypotheses and margin requirements, unlike unconstrained or derivative-free methods that would compromise precision. For very high-dimensional cases, reformulating constraints as penalties with L-BFGS-B could be considered, but SLSQP remains the gold standard for structured problems like ours.

```
x0 = np.concatenate([
    self.w.flatten(),
    [self.b],
    self.eta,
    self.xi,
    self.z,
    self.d
])

res = minimize(
    objective,
    x0,
    method='SLSQP',
    bounds=bounds,
    constraints=constraints,
    options={'maxiter': 1000}
)
```

3 A novel method for solving universum twin bounded support vector machine in the primal space

In supervised learning, the incorporation of Universum data—a third class that does not belong to either of the primary classes—has shown significant potential for improving classification performance. This study introduces a Newton-based approach, termed NMTBSVM, to solve the optimization problems of Twin Bounded Support Vector Machines with Universum data (MTBSVM) in the primal space. By transforming the constrained programming problems of MTBSVM into unconstrained optimization problems, we propose a generalized Newton's method tailored for these non-smooth objectives. Numerical experiments on synthetic, UCI, and NDC datasets, as well as gender detection from face images, demonstrate the efficiency and effectiveness of NMTBSVM compared to existing methods.

The optimization problems for NMTBSVM are formulated as follows:

$$\min_{w_1,b_1} \varphi_1(w_1,b_1) = \frac{1}{2}\|Aw_1 + e_1b_1\|^2 + \frac{c_1}{2}\|(e_2 + (Bw_1 + e_2b_1))_+\|^2 + \frac{c_2}{2}(\|w_1\|^2 + b_1^2)$$

$$+\frac{c_3}{2}\|((-1+\varepsilon)e_u-(Uw_1+e_ub_1))_+\|^2$$

and

$$\min_{w_2,b_2}\varphi_2(w_2,b_2)=\frac{1}{2}\|Bw_2+e_2b_2\|^2+\frac{c_4}{2}\|(e_1-(Aw_2+e_1b_2))_+\|^2+\frac{c_5}{2}(\|w_2\|^2+b_2^2)$$
$$+\frac{c_6}{2}\|((-1+\varepsilon)e_u+(Uw_2+e_ub_2))_+\|^2$$

For each hyperplane, we need to calculate Gradient and Hessian matrices and finally solve the problem using Numpy linear algebra library.

```
grad = (T1.T @ (T1 @ Z) +
       C1 * T2.T @ self.func(margin_B, 'pf') +
       C2 * Z -
       C3 * T3.T @ self.func(margin_U, 'pf'))

hessian = (T1.T @ T1 +
          C1 * T2.T @ D1 @ T2 +
          C2 * np.eye(Z.shape[0]) +
          C3 * T3.T @ np.diag(D2.flatten()) @ T3)

hessian += 1e-4 * np.eye(hessian.shape[0])

delta = np.linalg.solve(hessian, grad)
Z -= learning_rate * delta
```

Although the computation of the gradient and Hessian matrix is expensive, the hyperplanes typically converge and fit the data within just 5 iterations. However, only by knowing the optimal values of the hyperparameters can we achieve the best possible solution.

```

def fit(self):
    np.random.seed(42)
    self.w1 = np.random.normal(0, 0.01, (self.X.shape[1], 1))
    self.b1 = 0.0
    self.w2 = np.random.normal(0, 0.01, (self.X.shape[1], 1))
    self.b2 = 0.0

    for _ in range(5):
        self.w1, self.b1 = self.plane1(self.X, self.y, self.U,
                                       self.C[0], self.C[1], self.C[2], self.eps)
        self.w2, self.b2 = self.plane2(self.X, self.y, self.U,
                                       self.C[3], self.C[4], self.C[5], self.eps)
```

4 Semi-Supervised SVM (Unconstrained S^3VM)

The unconstrained variant of the Semi-Supervised SVM reformulates the optimization problem by eliminating binary decision variables while maintaining the core objective of margin maximization. This approach simplifies the solution process through continuous optimization.

4.1 Formulation

The objective function combines hinge loss and regularization. hinge loss helps remove constraints and automatically penalizes misclassified points. Also, No need for explicit slack variables (η, ξ, z) because the hinge loss already accounts for margin violations.

$$\min_{\mathbf{w},b} C\left[\sum_{i=1}^{\ell}\max(0,1-y_i(\mathbf{w}^\top\mathbf{x}_i+b))+\sum_{j=\ell+1}^{\ell+k}\max(0,1-|\mathbf{w}^\top\mathbf{x}_j+b|)\right]+\|\mathbf{w}\|_1 \tag{6}$$

4.2 Implementation

The Python implementation handles unlabeled data through pseudo-labeling:

```python
class S3VM_unconstrained:
    def __init__(self, C=1.0, eps=1e-4):
        self.C = C
        self.eps = eps
        self.w = None
        self.b = None

    def fit(self, X_labeled, y_labeled, X_unlabeled):

        n_features = X_labeled.shape[1]
        self.w = np.zeros((n_features, 1))
        self.b = 0.0

        X_aug = np.hstack([np.vstack([X_labeled, X_unlabeled]),
                           np.ones((len(X_labeled)+len(X_unlabeled), 1))])
        y = np.vstack([y_labeled,
                       np.sign(X_unlabeled @ self.w + self.b)])

        self._optimize(X_aug, y)

    def _optimize(self, X, y):
        def objective(params):
            w, b = params[:-1], params[-1]
            margins = y * (X[:, :-1] @ w + b)
            return self.C*np.sum(np.maximum(0,1-margins)) + np.sum(np.abs(w))

        res = minimize(objective, np.zeros(X.shape[1]),
                       method='L-BFGS-B',
                       bounds=[(None,None)]*X.shape[1])
        self.w, self.b = res.x[:-1], res.x[-1]

    def _optimize(self, X_aug, y):
        _, n_features = X_aug.shape

        def objective(params):
            w = params[:-1].reshape(-1, 1)
            b = params[-1]
            margins = y * (X_aug[:, :-1] @ w + X_aug[:, -1] * b)

            hinge_loss = np.sum(np.maximum(0, 1 - margins))

            norm1_w = np.sum(np.abs(w))

            return self.C * hinge_loss + norm1_w

        x0 = np.zeros(n_features)
        x0[-1] = 0
```

4.3 Key Differences from Constrained S^3VM

- **No integer variables**: Eliminates binary d_j decision variables
- **Continuous relaxation**: Uses absolute value for unlabeled data
- **Optimization**: Employes L-BFGS-B instead of SLSQP

Characteristic	Constrained	Unconstrained
Variables	Mixed-integer	Continuous
Constraints	Complex	Simple bounds
Optimizer	SLSQP	L-BFGS-B
Scalability	Limited	Improved

Table 1: Comparison of S^3VM formulations

5 Comparative Analysis

We conducted a comparative analysis of three semi-supervised learning approaches: S3VM Constrained, S3VM Unconstrained, and NewtonUTSVM across a set of UCI benchmark datasets and a synthetically generated dataset using the

Normally Distributed Cubic Clusters (NDCC) generator. The NDCC dataset, is designed to simulate complex distributions with adjustable separability by varying the variance of multivariate normal clusters. Introduces controlled randomness through the generation of cluster centers, class labels (based on a randomly generated separating plane), and noise.

Table 2: Accuracy (%) and Time (sec) for Different Methods Across Datasets

Dataset (Samples × Features)	Newton	S3VM Constrained	S3VM Unconstrained
Diabetes Pima (768 × 8)	**76.63** **0.03**	75.48 13.50	75.10 0.49
Ionosphere (350 × 34)	**86.55** **0.03**	**86.55** 12.35	84.03 1.40
Musk (476 × 166)	**86.42** **0.25**	81.48 158.12	80.86 3.12
Breast Cancer (568 × 31)	97.41 **0.03**	**97.93** 58.16	94.30 4.18
Sonar (207 × 60)	**85.71** **0.02**	80.00 6.43	67.14 0.90
Gender (5001 × 7)	**96.47** **0.80**	96.29 9993.28	95.71 22.28

As evidenced by the performance metrics summarized in Table 2 and Table 3, NewtonUTSVM consistently delivers strong results across both benchmark and synthetic datasets. These results highlight NewtonUTSVM as a promising and scalable solution for semi-supervised classification, striking a desirable balance between accuracy and computational cost. In nearly all cases, NewtonUTSVM achieved the highest or competitive classification accuracy while maintaining significantly lower computation time compared to both constrained and unconstrained S3VM approaches. Particularly on high-dimensional datasets such as Musk and Sonar, as well as large-scale datasets like Gender, NewtonUTSVM preserved strong performance with remarkable speed, whereas the constrained S3VM incurred substantial computational overhead. Even on the NDCC dataset (Table 3), which challenges classifiers with increasing data complexity and class inseparability, NewtonUTSVM maintained superior accuracy across all tested sizes, outperforming both S3VM variants consistently. These results emphasize its robustness and adaptability to various data conditions.

Table 3: Accuracy (%) and Time (sec) for NewtonUTSVM (Best of 1000 Random C Configs) vs S3VMs on NDCC Dataset

NDCC Dataset Size	NewtonUTSVM (Best of 1000)	S3VM Constrained	S3VM Unconstrained
100 × 10	**82.35** **0.01**	76.47 0.07	58.82 0.05
500 × 10	**71.76** **0.01**	62.94 3.07	59.41 0.66
1000 × 10	**69.12** **0.04**	63.53 34.82	57.94 3.28
100 × 100	**61.76** **0.03**	55.88 4.25	52.94 0.32

These results highlight NewtonUTSVM as a promising and scalable solution for semi-supervised classification, striking a desirable balance between accuracy and computational cost.

References

[1] Kristin Bennett and Ayhan Demiriz. "Semi-Supervised Support Vector Machines". In: *Advances in Neural Information Processing Systems*. Ed. by M. Kearns, S. Solla, and D. Cohn. Vol. 11. MIT Press, 1998. URL: https://proceedings.neurips.cc/paper_files/paper/1998/file/b710915795b9e9c02cf10d6d2bdb688c-Paper.pdf.

[2] Saeed Khosravi Hossein Moosaei. "A novel method for solving universum twin bounded support vector machine in the primal space". In: *Springer Nature Switzerland* (2023). URL: https://link.springer.com/article/10.1007/s10472-023-09896-5.