Introduction to Machine earning

SUPPORT VECTOR MACHINEs

1962, Vapnik, Alexey et al.

Advisor: PhD. Tran Luong Quoc Dai

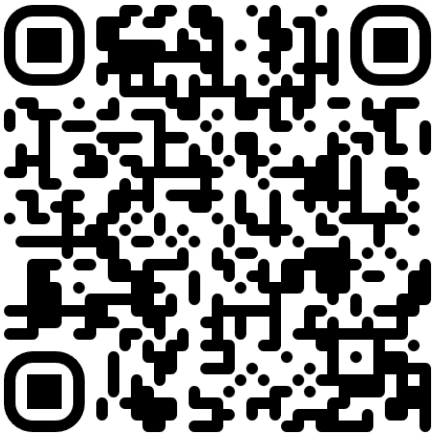
Research Group:

1. Phan Anh Kiệt - 523H0046
2. Nguyễn Quang Huy - 523H0140

QR Code

Presentation slide

Implementation



Github Resources

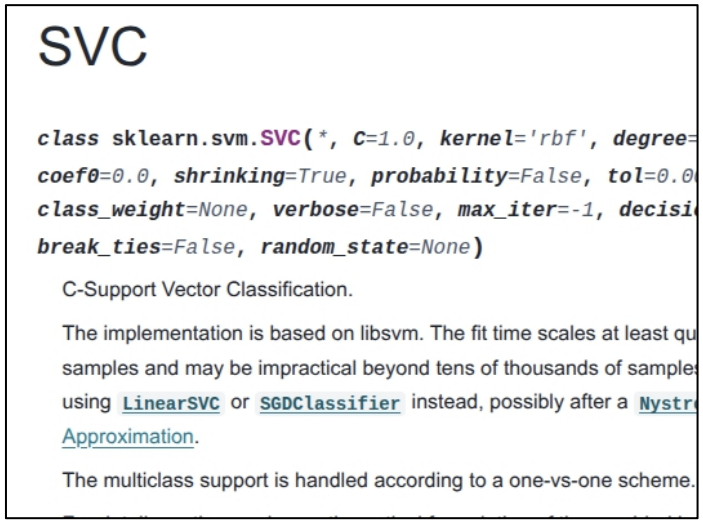
Outline

1. Introduction & General Idea
2. Architecture
3. Challenges & improvements
4. Applications
5. Demo / Visualization

1. Introduction & General Idea

What is SVM ?

A supervised machine learning algorithm primarily used for classification (and also regression).



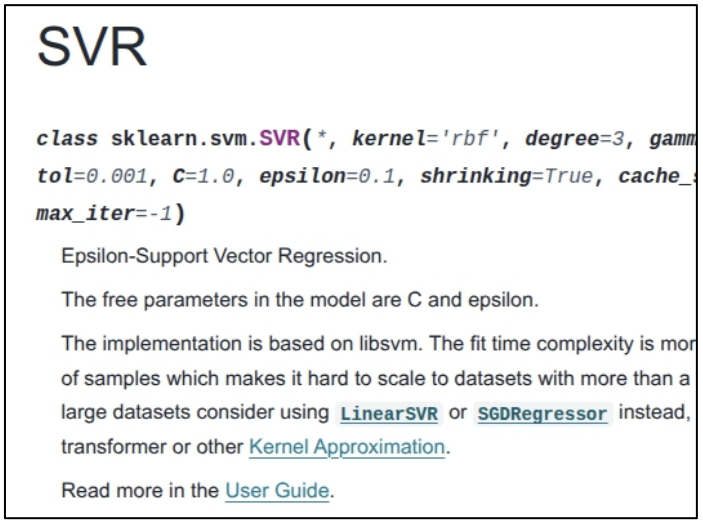
SVC

```
class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=
coef0=0.0, shrinking=True, probability=False, tol=0.0
class_weight=None, verbose=False, max_iter=-1, decisi
break_ties=False, random_state=None)
```

C-Support Vector Classification.

The implementation is based on libsvm. The fit time scales at least qu
samples and may be impractical beyond tens of thousands of sample
using `LinearSVC` or `SGDClassifier` instead, possibly after a `Nystr`
Approximation.

The multiclass support is handled according to a one-vs-one scheme.



SVR

```
class sklearn.svm.SVR(*, kernel='rbf', degree=3, gamm
tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache_
max_iter=-1)
```

Epsilon-Support Vector Regression.

The free parameters in the model are C and epsilon.

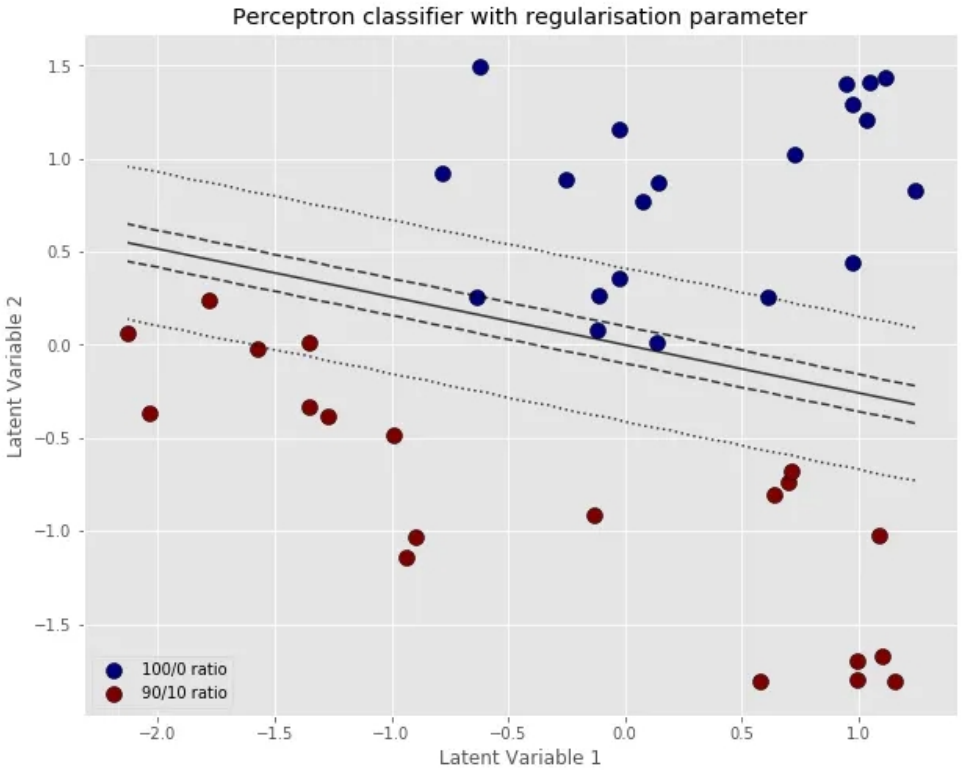
The implementation is based on libsvm. The fit time complexity is mor
of samples which makes it hard to scale to datasets with more than a
large datasets consider using `LinearSVR` or `SGDRegressor` instead,
transformer or other Kernel Approximation.

Read more in the User Guide.

1. Introduction & General Idea

Early Algorithms (Pre-1980s)

The Perceptron algorithm guaranteed convergence for linearly separable data but failed to provide a optimal solution.



1. Introduction & General Idea

Philosophy of SVM

Based on the principle of Structural Risk Minimization (SRM) - **General Model**

$$\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \min_{\mathbf{w}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} l(h_{\mathbf{w}}(\mathbf{x}_i), y_i)}_{\text{training loss}} + \lambda \underbrace{r(\mathbf{w})}_{\text{regularizer}}$$

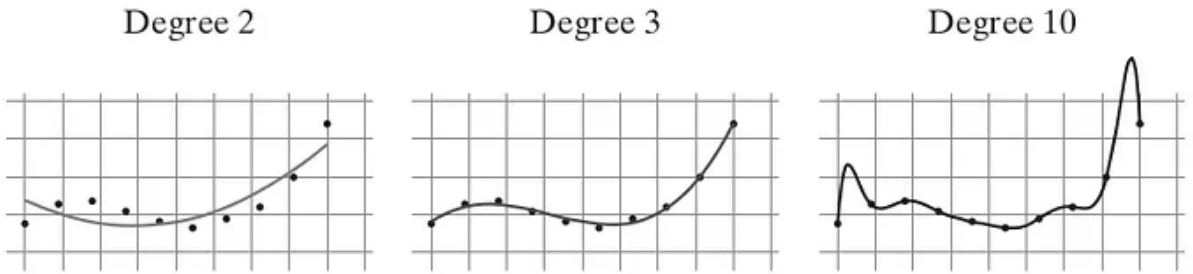
not Empirical Risk Minimization (ERM) - **Overfitting Model**

$$\min_{h} R_{emp}[h] = \min_{h} \frac{1}{n} \sum_{i=1}^{n} l(h(\mathbf{x}_i), y_i)$$

1. Introduction & General Idea

Philosophy of SVM

Let the hypothesis space is the class of polynomials of degree up to 10

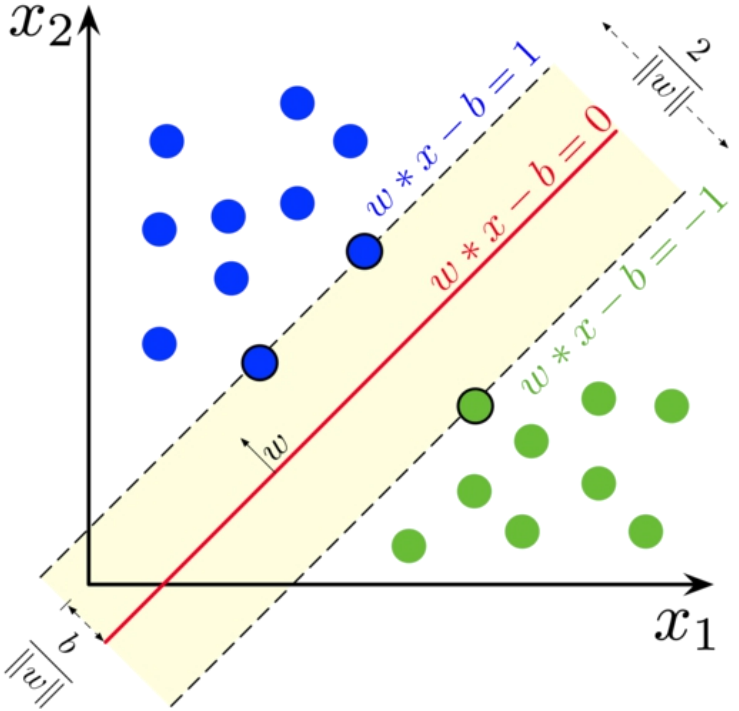


- ERM will choose a polynomial of degree 10. But that will overfit the data.
- Define the weight function by the 1/degree of the polynomials, then SRM will balance between training loss and polynomial degree.

1. Introduction & General Idea

Idea of SVM Hyperplanes & Margin

- Finds an optimal separating hyperplane that divides the data into classes.
- The optimal hyperplane is the one that maximizes the margin, the distance between the hyperplane and the nearest data points (the Support Vectors).



2. Architectures

Mathematical Architecture

In an n-dimensional, a hyperplane is defined as:

$$w^T x + b = 0 \quad with \quad x \in \mathcal{X}, \mathcal{X} = \mathbb{R}^n$$

- Orientation of the surface:

$$y(x) = w \cdot x + b, \quad y(x) \in \mathbb{R}$$

$$x_1, x_2 \in \{x \mid y(x) = 0\}$$

$$\implies w \cdot x_1 + b = 0, \quad w \cdot x_2 + b = 0 | v = x_2 - x_1, \quad v \in y$$

$$y(x_2) - y(x_1) = w \cdot (x_2 - x_1) = w \cdot v = 0$$

$$\implies w \perp v \implies w \perp y$$

2. Architectures

Mathematical Architecture

In an n-dimensional, a hyperplane is defined as:

$$w^T x + b = 0 \quad with \quad x \in \mathcal{X}, \mathcal{X} = \mathbb{R}^n$$

- Distance of the hyperplane from the origin:

$$d = \frac{|w \cdot x_0 + b|}{||w||}$$

$$with \quad x_0 = 0 \implies d = \frac{b}{||w||}$$

2. Architectures

Mathematical Architecture

Distance between 2 points

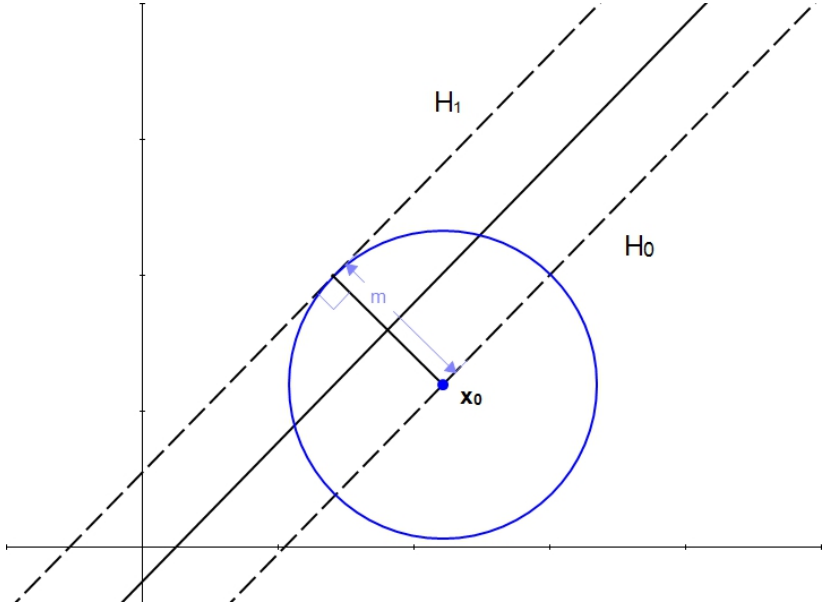
$$w \cdot x + b = 0, \quad x_0 \in \mathbb{R}^n, \quad x_p \in \mathbb{R}^n$$

$$x_0 - x_p = \lambda w$$

$$w \cdot x_p + b = 0 \implies w \cdot (x_0 - \lambda w) + b = 0$$

$$w \cdot x_0 - \lambda \|w\|^2 + b = 0 \implies \lambda = \frac{w \cdot x_0 + b}{\|w\|^2}$$

$$r = \|x_0 - x_p\| = \|\lambda w\| = |\lambda| \, \|w\| = \frac{|w \cdot x_0 + b|}{\|w\|}$$



2. Architectures

Mathematical Architecture **Functional margin**

$$(x_i, y_i) \in \mathbb{R}, \quad w \cdot x + b = 0$$

$$\hat{\gamma}_i = y_i(w \cdot x_i + b)$$

- Tells whether the hyperplane is correct in classification and how “strong” it is:
  - gamma > 0 : Correct classified

    y = ax + b, A = (3, -1)

    If y^(3) = -3 -> Gamma = (-1) x (-3) = 3
  - gamma < 0 : Wrong classififed

    y = ax + b, B = (1, 4)

    If y^(1) = -2 -> Gamma = 4 x (-2) = -8
- Does not reflect actual distance.

2. Architectures

Mathematical Architecture Geometric margin

$$\gamma_i = \frac{\hat{\gamma}_i}{||w||} = \frac{y_i(w \cdot x_i + b)}{||w||}$$

- Perpendicular distance from point x_i to hyperplane.
- The hard-margin SVM problem is to maximize the geometric margin.

2. Architectures

Mathematical Architecture

$$y = 2x - 1$$

$$\gamma^{(g)}(1) = \frac{1 \cdot (2 \cdot 1 - 1)}{||2||} = 0.5$$

$$\gamma^{(f)}(1) = 1 \cdot (2 \cdot 1 - 1) = 1$$

$$\gamma^{(g)}(20) = \frac{1 \cdot (20 \cdot 1 - 10)}{||20||} = 0.5$$

$$\gamma^{(f)}(20) = 1 \cdot (20 \cdot 1 - 10) = 10$$

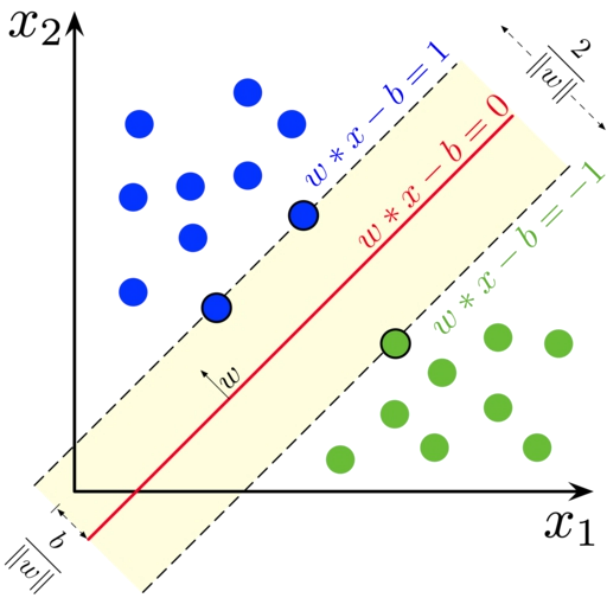
2. Architectures

Mathematical Architecture

Canonical Form to Geomatric Margin

- Rescale the parameters w and b such that the functional margin of the closest data point(s) to the decision boundary is exactly 1.

$$\gamma_i = \frac{1}{||w||} y_i(w \cdot x_i + b) = \frac{\pm 1}{||w||}$$



Mathematical Architecture Primal Formulation

- Goal

$$max \frac{1}{||w||} \quad or \quad min \frac{1}{2}||w||^2$$

$$y_i(w \cdot x_i + b) \geq 1$$

$$\left(\frac{1}{||w||}\right)' = \frac{||w||'}{||w||}$$

$$= \frac{\left(w_1^2 + w_2^2 + \cdots + w_n^2\right)'}{2 \cdot \left(w_1^2 + w_2^2 + \cdots + w_n^2\right)} \cdot ||w|| \neq 0$$

$$\gamma = \frac{2}{||w||} \quad \forall w \models w^T x + b = \pm 1$$

$$\max_{w,b} \gamma \Leftrightarrow \max_{w,b} \frac{2}{||w||} \Leftrightarrow \max_{w,b} \frac{1}{||w||}$$

$$\Leftrightarrow \min_{w,b} ||w|| \Leftrightarrow \min_{w,b} ||w||^2 \Leftrightarrow \min_{w,b} \frac{1}{2}||w||^2$$

2. Architectures

Mathematical Architecture **Why not derivative directly**

$$max\frac{1}{||w||} \quad or \quad min\frac{1}{2}||w||^2$$

$$3x_1 - 1x_2 + 3 = 0$$

$$w = (3, -1) \implies ||w||^2 = 3^2 + (-1)^2 = 10$$

$$\left(\frac{1}{2}||w||^2\right)' = ||w|| = 0 \implies w = (0, 0)$$

$$w = (0, 0) \implies 3 \cdot 0 - 1 \cdot 0 + 3 = 0$$

Hyperplane disappear (Without constraint)

2. Architectures

Mathematical Architecture Lagrangian Function and KKT Conditions.

- Lagrangian $\mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i (y_i (w^T x_i + b) - 1) \quad \alpha_i \geq 0$

- KTT

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \implies w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \implies \sum_{i=1}^{n} \alpha_i y_i = 0$$

Mathematical Architecture Dual Formulation

$$\mathcal{L}(w, b, \alpha) = \frac{1}{2} w^\top w - \sum_{i=1}^{n} \alpha_i y_i w^\top x_i - b \sum_{i=1}^{n} \alpha_i y_i + \sum_{i=1}^{n} \alpha_i$$

$$\mathcal{L}(\alpha) = \frac{1}{2} \Big(\sum_k \alpha_k y_k x_k \Big)^\top \Big(\sum_\ell \alpha_\ell y_\ell x_\ell \Big) * \sum_i \alpha_i y_i \Big(\sum_k \alpha_k y_k x_k \Big)^\top x_i - \sum_i \alpha_i$$

$$= \frac{1}{2} \sum_{k,\ell} \alpha_k \alpha_\ell y_k y_\ell x_k^\top x_\ell * \sum_{i,k} \alpha_i \alpha_k y_i y_k x_k^\top x_i - \sum_i \alpha_i.$$

$$\frac{1}{2} A - A = -\frac{1}{2} A.$$

$$\mathcal{L}(\alpha) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^\top x_j + \sum_{i=1}^{n} \alpha_i.$$

Mathematical Architecture Dual Formulation

- Standard Form

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i^T x_j)$$

$$subject\ to\ \alpha_i \geq 0 \quad and \quad \sum_{i=1}^{m} \alpha_i y_i = 0$$

- Decision Function

$$f(x) = sign\left(\sum_{i=1}^{n} \alpha_i y_i x_i^T x + b \right)$$

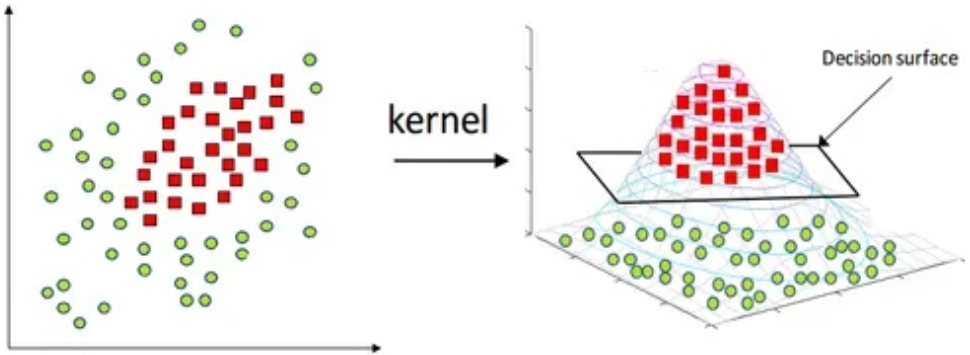
2. Architectures

Mathematical Architecture Kernel Trick

- A technique to map non-linearly separable data into a higher-dimensional Feature Space where a linear separation is possible.

$$\Phi : \mathcal{X} \rightarrow \mathcal{F}$$

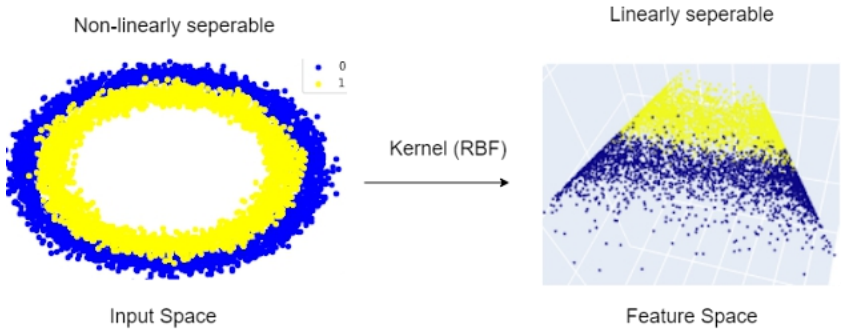
$$x = [x_1, x_2]^T \rightarrow \Phi(x) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T$$

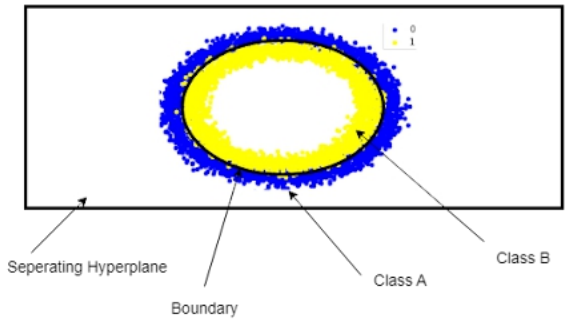


2. Architectures

Mathematical Architecture RBF Kernel

- The Kernel Trick replaces the expensive calculation of the dot product in the high-dimensional space with a fast, pre-defined Kernel Function in the low-dimensional space.

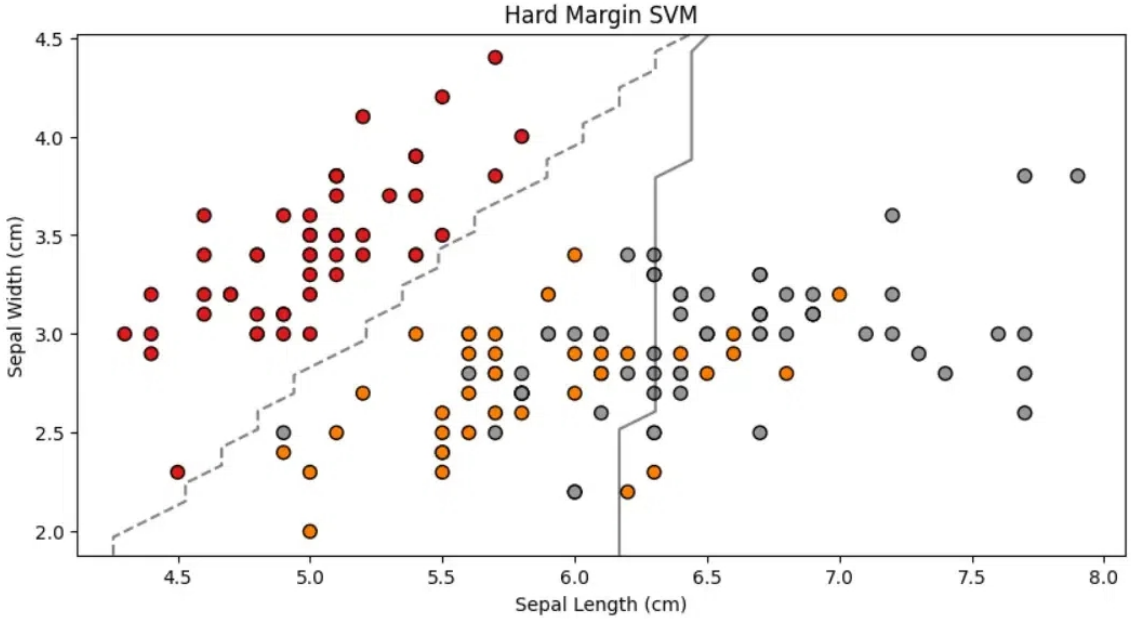




3. Challenges & Improvement

Hard Margin Challenge

- The Hard Margin SVM fails if the data is not perfectly linearly separable.



3. Challenges & Improvement

Soft Margin Solution

- Introduces Slack Variables to allow some data points to violate the margin constraint or even be misclassified.

$$y_i(w^T x_i + b) \geq 1 - \xi_i \quad \forall(x_i, y_i)$$

- On the boundary $\xi_i = 0$
- Margin Violation $0 < \xi_i < 1$
- Wrong classifier $\xi_i \geq 1$

3. Challenges & Improvement

Regularization Parameter

$$min \sum \xi_i$$

A hyperparameter that controls the trade-off:

- **High C**: Penalizes classification errors/slack heavily (stricter, narrower margin, higher risk of overfitting).
- **Low C**: Allows more errors/slack (looser fit, wider margin, higher risk of underfitting).

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

$$subject\ to \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

3. Challenges & Improvement

Loss Function

- The objective function incorporates the Hinge Loss, which penalizes points that are inside the margin or on the wrong side.

$$\min_{w,b} \underbrace{\frac{1}{2}||w||^2}_{Regularization} + C\sum_{i=1}^{m} \underbrace{max(0, 1 - y_i(w^T x_i + b))}_{Hinge\ Loss}$$

  - Correctly classfied $y \cdot f(x) \geq 1$
  - Within margin $0 < y \cdot f(x) < 1$
  - Misclassified $y \cdot f(x) < 0$

3. Challenges & Improvement

Sequential Minimal Optimization (SMO)

The Dual Optimization problem for SVM is a complex, large-scale Quadratic Programming (QP) problem. Solving it using general QP solvers is very slow, especially for large datasets.

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

1. Select a couple of Lagrange multipliers by Heuristics
2. Freezing other ...
3. The original complex QP optimization problem is reduced to a simple quadratic function of one variable.

4. Manual calculation

$$\mathbf{x}_i \in \mathbb{R}^n$$

$$G_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$$

$$\max_{\boldsymbol{\alpha}} W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

We define a linearly separable dataset with 3 points (m=3).

Negative Class ($y = -1$):

$$x_1 = [1, 1]^T$$

Positive Class ($y = +1$):

$$x_2 = [3, 3]^T$$

$$x_3 = [3, 1]^T$$

4. Manual calculation

First, we compute the Gram Matrix G, where $G_{i,j} = x_i^T x_j$ (Linear Kernel)

$$x_1^T x_1 = 1 \cdot 1 + 1 \cdot 1 = 2$$
$$x_2^T x_2 = 3 \cdot 3 + 3 \cdot = 18$$
$$x_3^T x_3 = 3 \cdot 3 + 1 \cdot 1 = 10$$
$$x_1^T x_2 = 1 \cdot 3 + 1 \cdot 3 = 6$$
$$x_1^T x_3 = 1 \cdot 3 + 1 \cdot 1 = 4$$
$$x_2^T x_3 = 3 \cdot 3 + 3 \cdot 1 = 12$$

The Gram Matrix is:

$$G = \begin{pmatrix} 2 & 6 & 4 \\ 6 & 18 & 12 \\ 4 & 12 & 10 \end{pmatrix}$$

We want to maximize the Dual Objective function:

$$W(\alpha) = \sum_{i=1}^{3} \alpha_i - \frac{1}{2} \sum_{i=1}^{3} \sum_{j=1}^{3} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^\top \mathbf{x}_j)$$

$$\alpha_i \geq 0$$

$$\sum \alpha_i y_i = 0 \implies -\alpha_1 + \alpha_2 + \alpha_3 = 0 \implies \boldsymbol{\alpha_1 = \alpha_2 + \alpha_3}$$

$$W = (\alpha_1 + \alpha_2 + \alpha_3) - \frac{1}{2} \begin{bmatrix} \alpha_1^2(2) + \alpha_2^2(18) + \alpha_3^2(10) \\ + 2\alpha_1\alpha_2(-1)(1)(6) \\ + 2\alpha_1\alpha_3(-1)(1)(4) \\ + 2\alpha_2\alpha_3(1)(1)(12) \end{bmatrix}$$

Simplifying the quadratic part:

$$W = (\alpha_1 + \alpha_2 + \alpha_3) - \left[\alpha_1^2 + 9\alpha_2^2 + 5\alpha_3^2 - 6\alpha_1\alpha_2 - 4\alpha_1\alpha_3 + 12\alpha_2\alpha_3\right]$$

4. Manual calculation

To solve this, we substitute the constraint alpha_1 = alpha_2 + alpha_3 into the equation to eliminate alpha_1. By geometric inspection, point x_2 is farther from the boundary than x_3. Thus, we hypothesize that x_2 is not a support vector, meaning alpha_2 = 0. Let's test this hypothesis. If alpha_2 = 0, then alpha_1 = alpha_3.

Substituting alpha_2 = 0 and alpha_1 = alpha_3 into W:

$$W = (2\alpha_3) - [\alpha_3^2 + 0 + 5\alpha_3^2 - 0 - 4\alpha_3^2 + 0]$$

$$W = 2\alpha_3 - 2\alpha_3^2$$

To find the maximum, we take the derivative with respect to alpha_3 and set to 0:

$$\frac{dW}{d\alpha_3} = 2 - 4\alpha_3 = 0 \implies \alpha_3 = 0.5$$

4. Manual calculation

- Recover the weight vector w using the formula

$$w = \sum \alpha_i y_i x_i$$

$$w = 0.5(-1)\begin{pmatrix}1\\1\end{pmatrix} + 0(1)\begin{pmatrix}3\\3\end{pmatrix} + 0.5(1)\begin{pmatrix}3\\1\end{pmatrix}$$

$$w = \begin{pmatrix}-0.5\\-0.5\end{pmatrix} + \begin{pmatrix}1.5\\0.5\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix}$$

- Computing the bias b: We use a support vector (e.g., x_3) and the support vector equation

$$y_3(w^T x_3 + b) = 1$$

$$1 \cdot ([1, 0]\begin{pmatrix}3\\1\end{pmatrix} + b) = 1$$

$$1 \cdot (3 + b) = 1 \implies b = -2$$

4. Manual calculation

Final Model: The optimal hyperplane is defined by

$$w = [1, 0]^T$$

and

$$b = -2$$

The decision boundary equation is:

$$x_1 - 2 = 0 \implies x_1 = 2$$

This is a vertical line passing through x=2, which perfectly separates x=1 (Negative) and x=3 (Positive).

5. Demonstration

The Iris Case Study

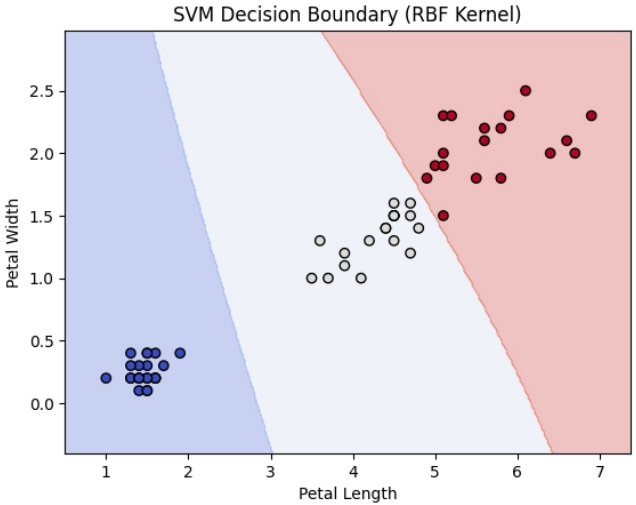
Flower classification

- Samples: 150 Flower samples (50 per class).
- Features (n = 4): Sepal length, Sepal width, Petal length, Petal width (all in cm).
- Classes (K=3): Iris setosa, Iris versicolor, and Iris virginica.
- RBF Kernel.
- Regularization Hyperparameter (C) = 1

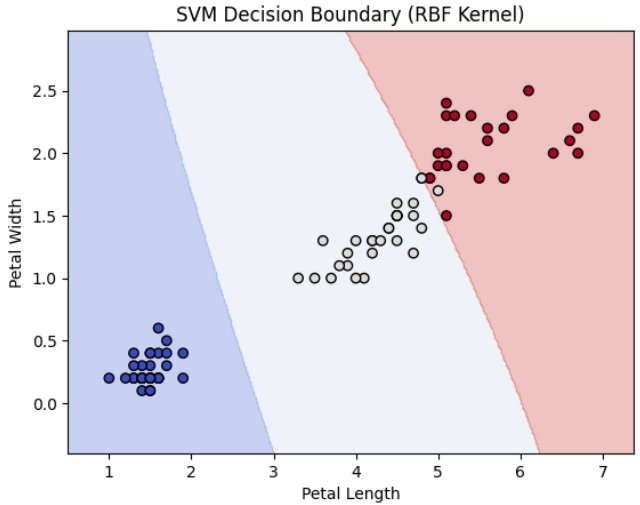
5. Demonstration

The Iris Case Study

Flower classification



Train : Test = 6:4 - Accuracy = 100%
C = 1

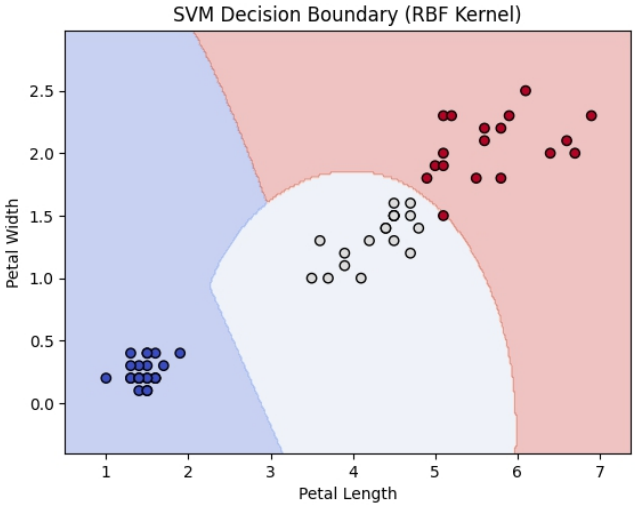


Train : Test = 4:6 - Accuracy = 97,78%
C = 1

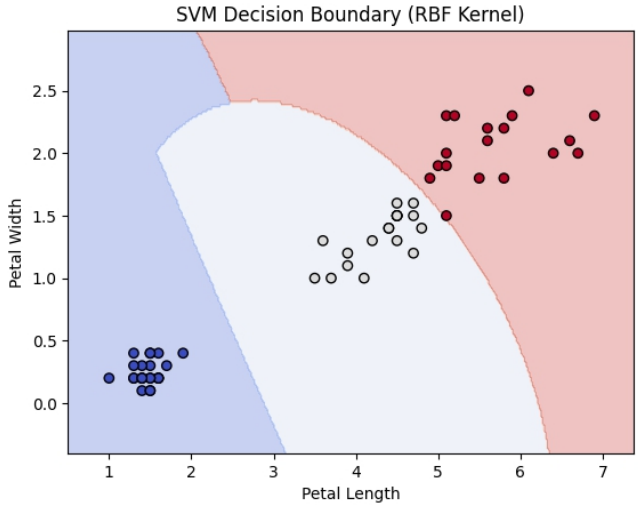
5. Demonstration

The Iris Case Study

Flower classification



Train : Test = 6:4 - Accuracy = 98.33%
C = 30



Train : Test = 6:4 - Accuracy = 100%
C = 10

References

- PyTorch implementations of GANs
- A list of all named GANs
- Vanilla GANs paper
- WGAN paper
- Fréchet Inception Distance
- A mix of GAN implementations including progressive growing
- GAN-play
- Style GAN