Support Vector Machine

Nasim Javedani, Mahdi Akbari, Masoud Mozafari

January 29, 2025 — Sharif University of Technology

History of SVM

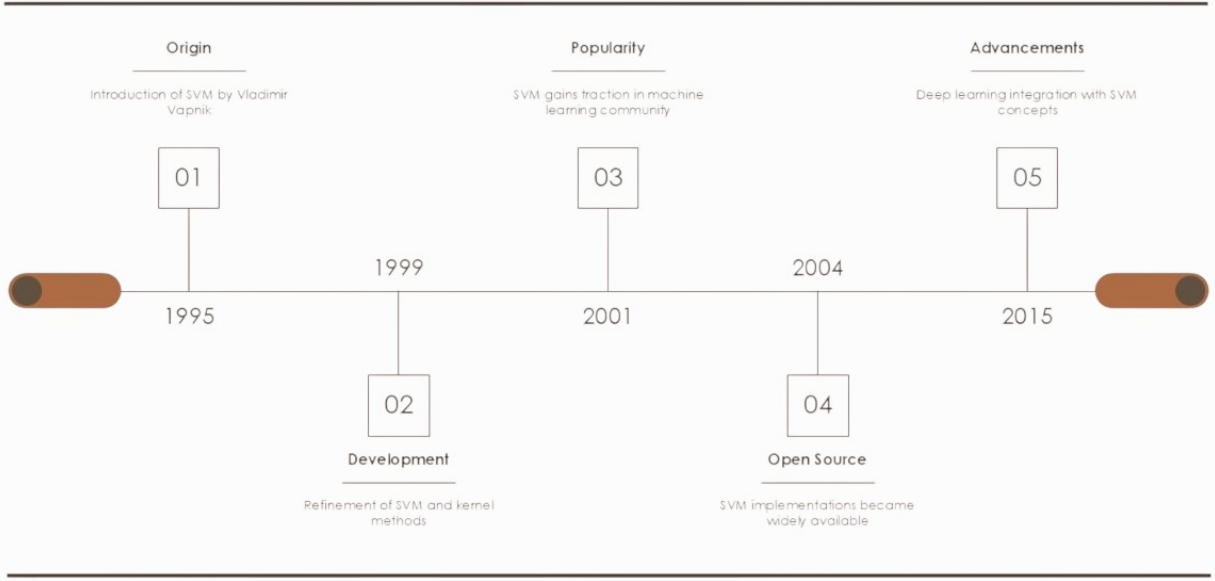


Figure: This is an example image.

Intro

What is SVM?

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. It is particularly effective in high-dimensional spaces and when the number of dimensions exceeds the number of samples.
Best used for:

- Image classification
- Text categorization
- Bioinformatics

Frame title

Core idea of SVM

1. Maximizing Margin: SVMs try to find the decision boundary (a hyperplane in feature space) that separates classes with the largest possible margin, because a larger margin leads to better generalization.

2. Support vectors: The training samples that lie closest to the decision boundary are called “support vectors.” They effectively “support” or define the boundary.

Frame title

Core idea of SVM

Why it might seem special:

- Instead of focusing on the idea of fitting data with minimal training error alone, SVM emphasizes structural risk minimization (i.e., controlling model complexity to avoid overfitting).

Non-important trivia:

- The name “Support Vector Machine” might sound like a physical machine but is simply an algorithm in linear algebra and optimization.

The Optimization Problem

$$\text{maximize } \frac{1}{\|\mathbf{w}\|}$$

$$\text{subject to } \min_{n=1,2,\ldots,N} \left|\mathbf{w}\mathbf{x}_n + b\right| = 1$$

Notice $\left|\mathbf{w}\mathbf{x}_n + b\right| = y_n \left(\mathbf{w}\mathbf{x}_n b\right)$

$$\text{Minimize } \frac{1}{2}\mathbf{w}\mathbf{w}$$

$$\text{subject to } y_n \left(\mathbf{w}\mathbf{x}_n + b\right) \geq 1, \quad \text{for } n 1, 2, \ldots, N$$

Lagrange formulation

$$\text{Minimize } (\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}\mathbf{w} - \sum_{n=1}^{N} \alpha_n \left(y_n(\mathbf{w}\mathbf{x}_n + b) - 1 \right)$$

w.r.t. $\mathbf{w}$ and b, and maximize w.r.t. each $\alpha_n \geq 0$.

$$\nabla_{\mathbf{w}} = \mathbf{w} - \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n = 0 \qquad \frac{\partial}{\partial b} = -\sum_{n=1}^{N} \alpha_n y_n = 0$$

Substituting

subject to $y_n(\mathbf{w}\mathbf{x}_n + b) \geq 1, \quad \text{for } n = 1, 2, \ldots, N \quad \mathbf{w} \in \mathbb{R}^d, \quad b \in \mathbb{R}$

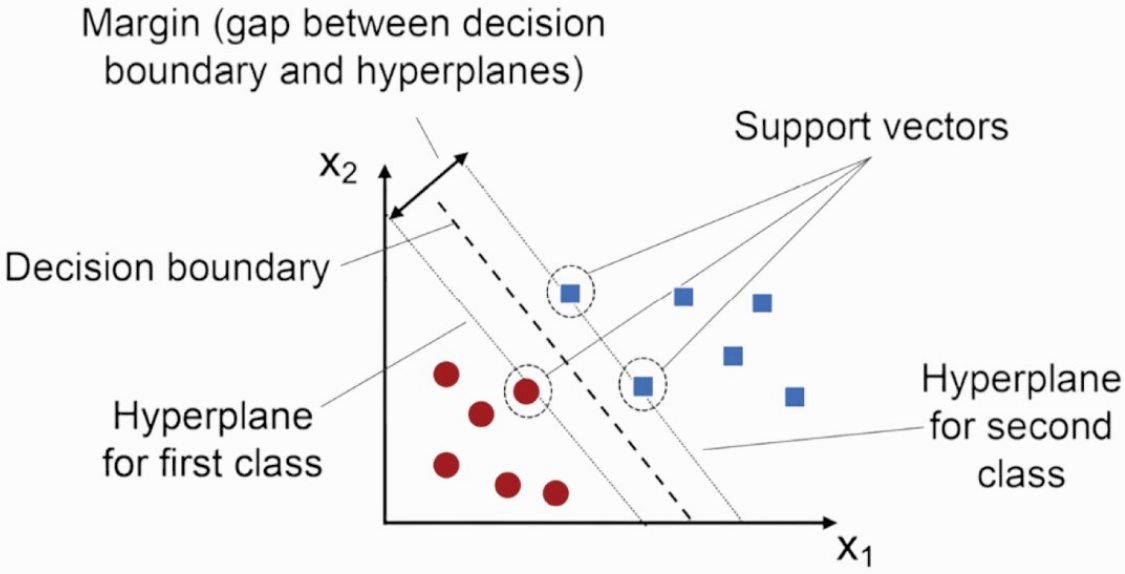
Lagrange? inequality constraints $\Rightarrow$ KKT

Substituting: $\mathbf{w} = \sum_{n=1}^{N} \alpha_n y_n \mathbf{x}_n$ and $\sum_{n=1}^{N} \alpha_n y_n = 0$

in the Lagrangian $(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}\mathbf{w} - \sum_{n=1}^{N} \alpha_n \left(y_n(\mathbf{w}\mathbf{x}_n + b) - 1\right)$

we get $(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} y_n y_m \alpha_n \alpha_m \mathbf{x}_n \mathbf{x}_m$

visualization

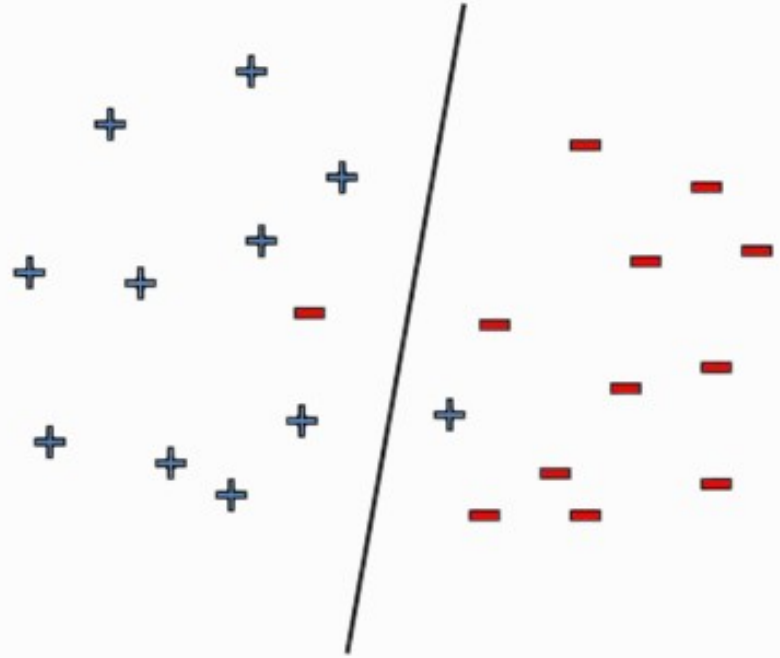


Problem

What if data is not linearly separable Use features of features of features...

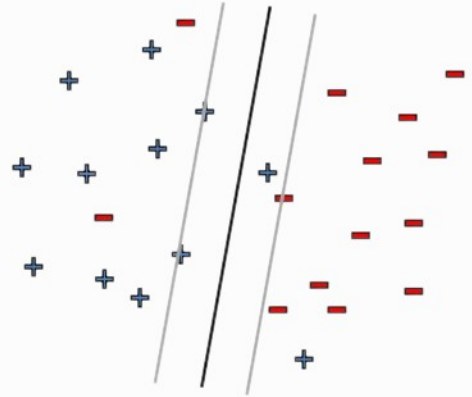
$$x_1^2, x_2^2, x_1 x_2, \dots, \exp(x_1)$$

But run risk of overfitting!

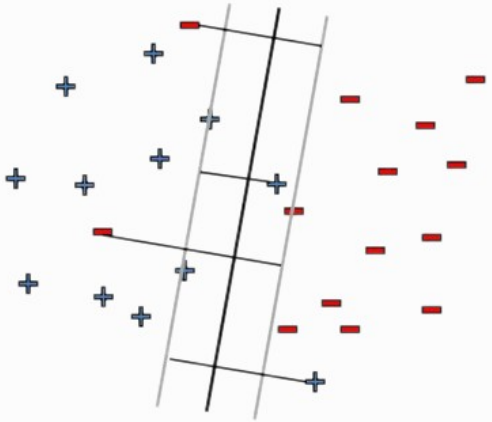


Still Problem...

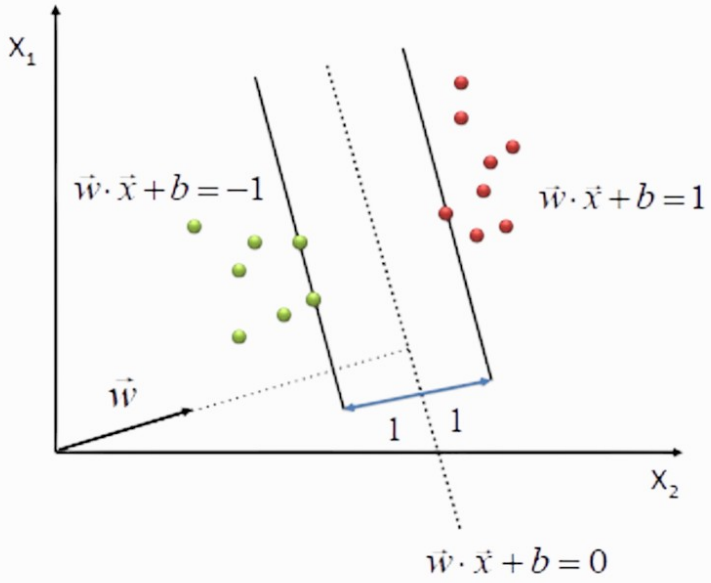
What if data is still not linearlyseparable?
then we allow "error" in classification



Smaller margin ⇔ larger ‖w‖



visualization



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

s.t.

$(w\cdot x+b)\geq 1, \forall x$ of class 1

$(w\cdot x+b)\leq -1, \forall x$ of class 2

Soft margin

The Soft-Margin Extension

Real data is often not perfectly separable. SVM extends to “soft margin” by introducing slack variables $\xi \geq 0$ that allow some points to violate the margin or even be on the wrong side of the hyperplane. We then minimize:

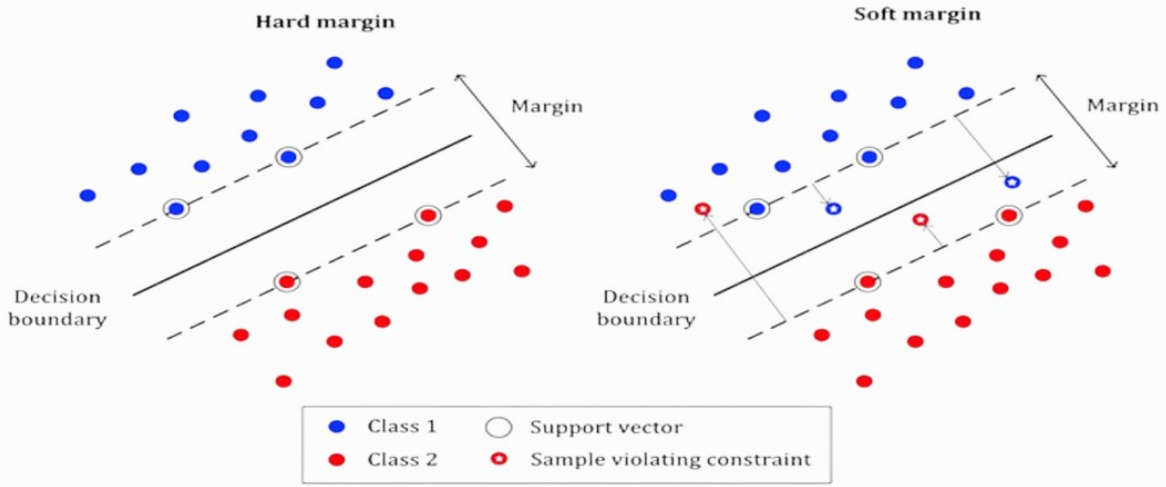
The soft margin formulation for a support vector machine is the following:

$$\min \quad \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i$$

$$\text{subject to } y_i(x_i^T w + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0$$

Soft margin

Non-important detail: “Soft margin” was sometimes considered a confusing name, but it stuck. Think of it as “we’re allowed to be a bit sloppy” with the boundary.



Kernel trick

Why We Use Kernels?

- Some data can't be separated linearly in the original feature space. By mapping the data to a higher-dimensional space (potentially infinite-dimensional) using a function (x), the data might become linearly separable there.

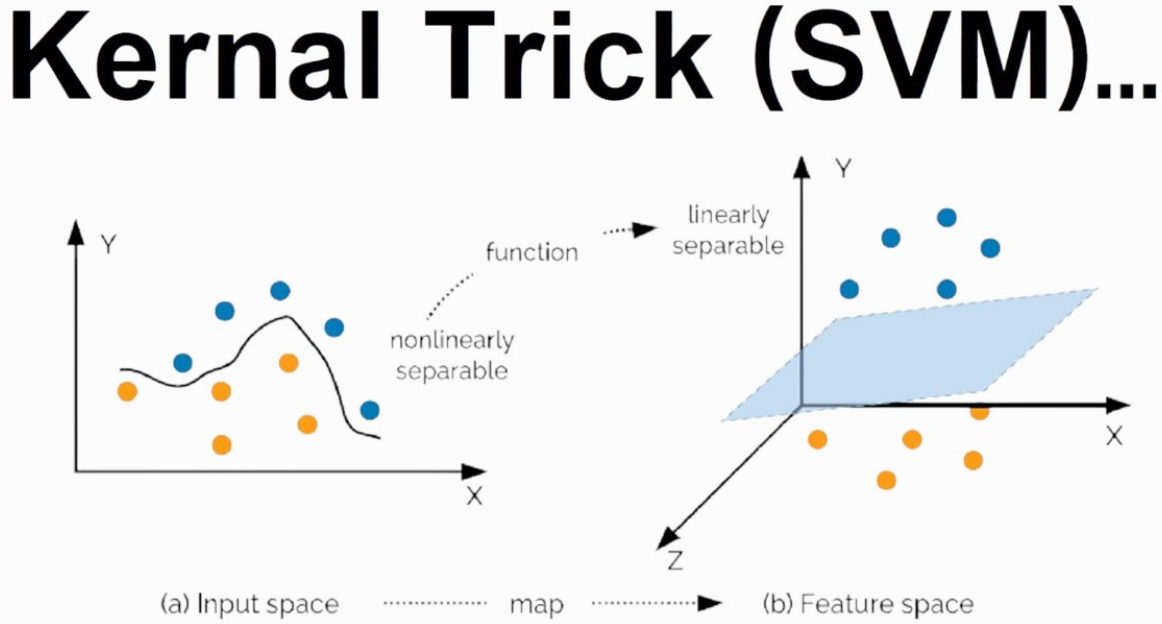
- Directly computing □(x) could be extremely large or computationally impossible. Kernel functions allow us to compute inner products in the transformed space without explicitly mapping the data:

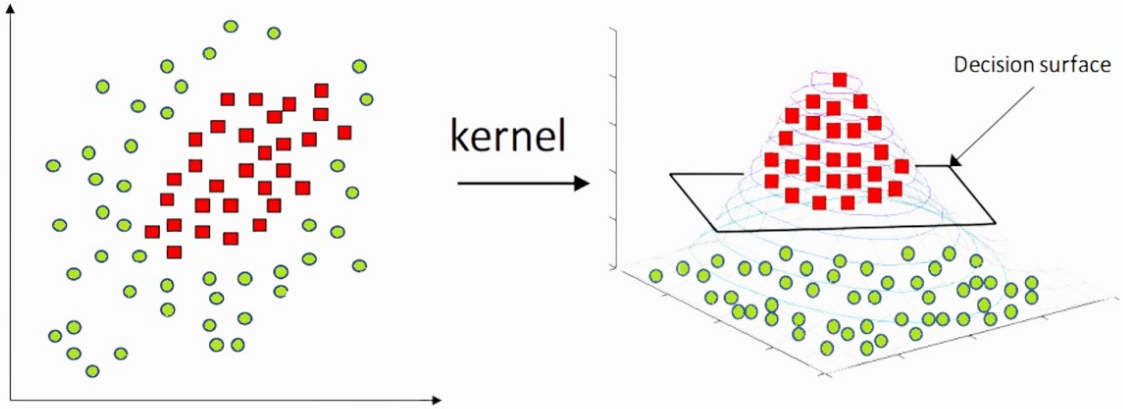
$$K(x_i, x_j) = (x_i) \cdot (x_j).$$

Common Kernels

- Linear Kernel: When the data are linearly separable or when computational efficiency is a priority, it is also a good starting point before trying more complex kernels. $K(x_i, x_j) = x_i^T x_j$
- Polynomial Kernel: Creates high-dimensional feature spaces, but SVMs handle this efficiently with the kernel trick, avoiding memory issues common in linear algorithms. $K(x_i, x_j) = (x_i^T x_j + c)^d$
- Sigmoid Kernel: Inspired by neural networks (tanh activation), it often behaves similarly to the RBF kernel with proper parameter tuning. $K(x_i, x_j) = \tanh(\alpha x_i^T x_j + c)$

Kernel visualization




kernel
Decision surface

Lagrange Multipliers

SVM optimization problems are often expressed in their dual form because it is easier to incorporate the kernel trick. In the dual form, we solve for Lagrange multipliers i, and the decision function becomes:

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i K(x_i, x) + b$$

where only those α_i associated with support vectors (points on the margin boundary or misclassified points) are non-zero.
“Karush–Kuhn–Tucker (KKT) conditions” are behind how the primal and dual optimization solutions relate. You might not need to dwell on them unless your audience is mathematically inclined or you want to show off your optimization knowledge.

Dual Formulation of SVMs

- **Hyperparameters**:
  - C: Balances decision boundary smoothness and correct classification.
  - γ: Controls influence of training points in RBF/polynomial kernels.
- **Feature Scaling**:
  - Crucial for SVMs (especially RBF) as they rely on distance metrics.
- **Computational Complexity**:
  - Training is $O(n^3)$ for naive implementations. Use SMO for scalability.
- **Interpretability**:
  - Linear kernel: Interpretable (like logistic regression weights).
  - Kernel SVMs: Less interpretable due to higher-dimensional boundaries.
- **Large Datasets**:
  - Use specialized solvers or approximate methods for millions of samples.

Pros and Cons

- **Good Generalization**: Maximizing the margin leads to robust performance on unseen data.
- **Effective in High Dimensions**: Handles many features well using kernels and margin-based approaches.
- **Kernel Flexibility**: Adapts to various data patterns (linear, polynomial, RBF, etc.).
- **Parameter Tuning**: Choosing the right kernel and tuning hyperparameters (C, γ, degree, etc.) can be challenging.
- **Scalability**: Training is slow and memory-intensive for large datasets.
- **Interpretability**: Non-linear SVMs are less interpretable; linear SVMs are more transparent.

Sources

[1] Corinna Cortes and Vladimir Vapnik, *Support-Vector Networks*, AT&T Bell Labs., Hohndel, NJ 07733, USA. Available at: https://link.springer.com/article/10.1007/BF00994018

[2] A. J. Smola and B. Schölkopf, Support Vector Machines: Theory and Applications, Conference Paper in Lecture Notes in Computer Science, September 2001.


دانشگاه صنعتی شریف