KERNELS, SVM

CS5691- PR & MACHINE LEARNING

Murphy 14.1, 14.2.1-14.2.6, 14.3, 14.4, 14.5

C. M. Bishop

+ Wiki + Online Tut notes;

Binary Classification Problem

We are given a training set

$$D = \{(x_n, t_n)\}_{n=1}^{N}$$

where

- $\mathbf{x}_n \in \mathrm{R}^D$ is an input vector
- $t_n \in \{-1, +1\}$ is the class label

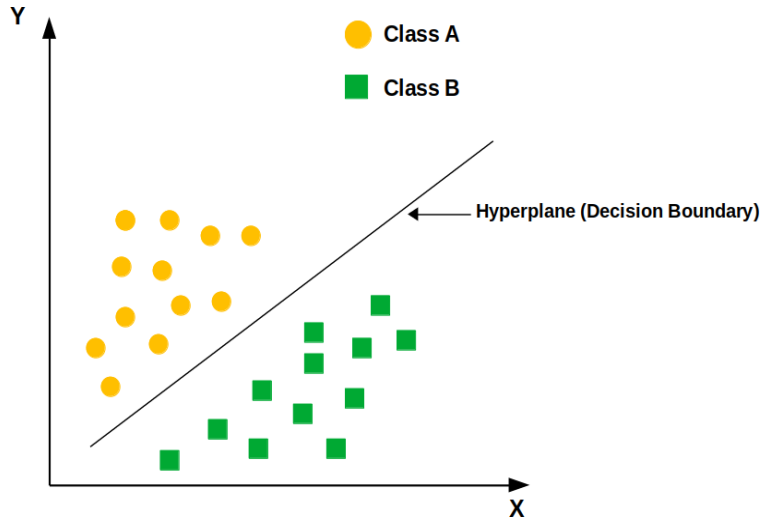
We consider a linear discriminant function:

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

Classification rule: sign($y(\mathbf{x})$)

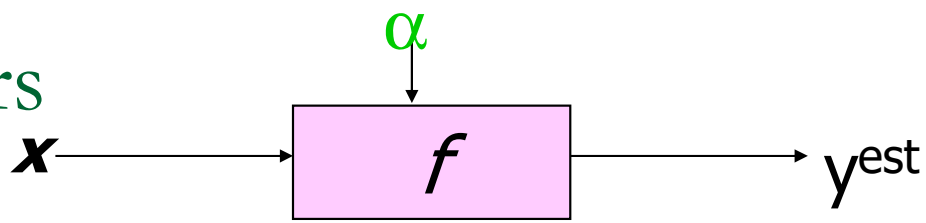
This means:

- If $y(\mathbf{x}) > 0 \Rightarrow$ Class +1
- If $y(\mathbf{x}) < 0 \Rightarrow$ Class −1



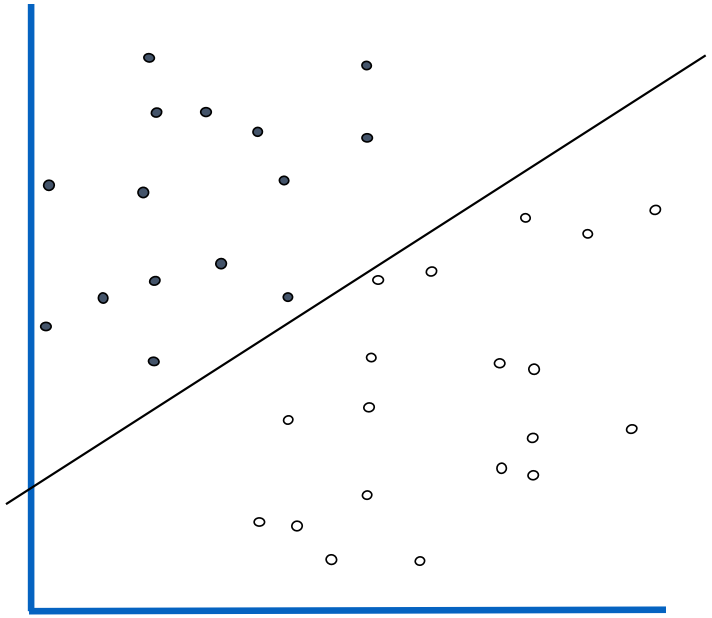
Two classes separated by a linear boundary

Linear Classifiers



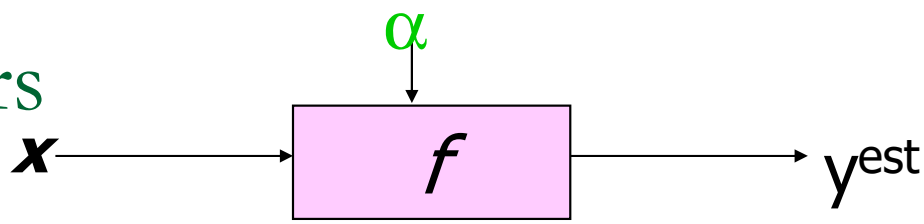
- denotes +1
- denotes -1

$f(\boldsymbol{x}, \boldsymbol{w}, \alpha) = sign(\boldsymbol{w}\,\boldsymbol{x} + \alpha)$

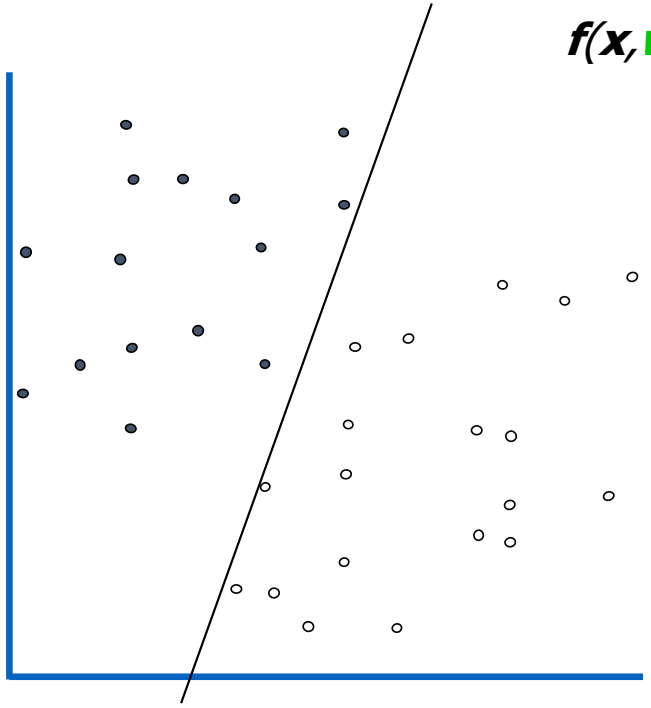


Easy to classify this data?

Linear Classifiers



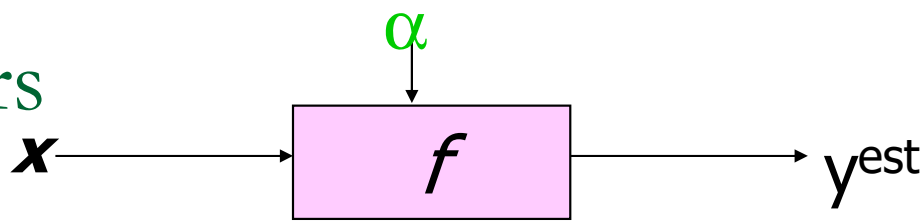
- denotes +1
- denotes -1



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

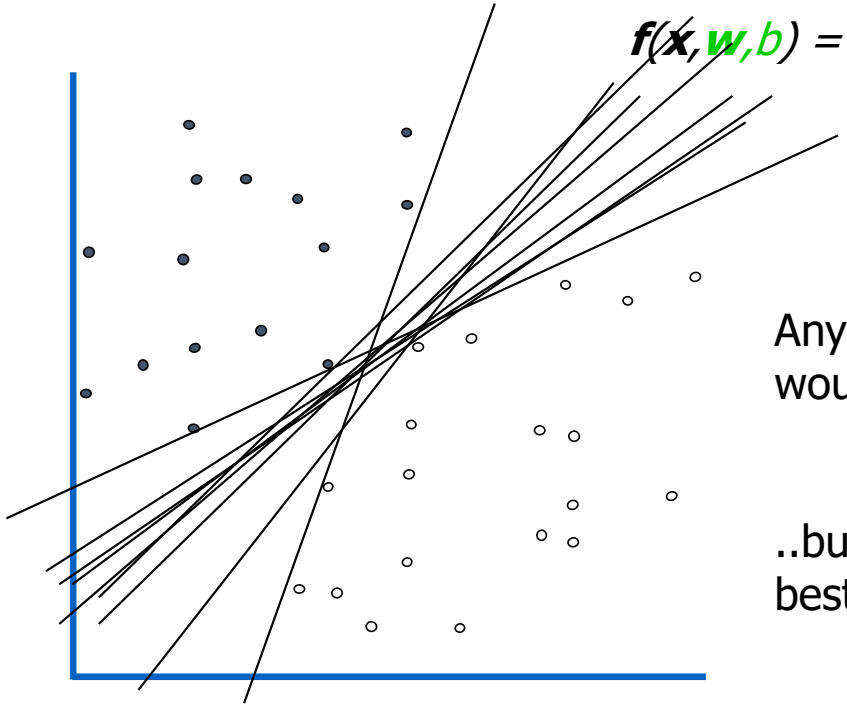
How would you classify this data?

Linear Classifiers



- denotes +1
- denotes -1

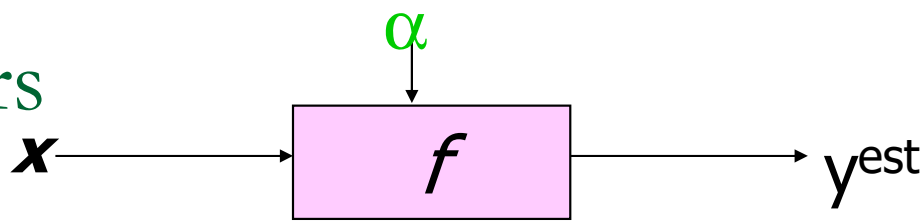
$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$



Any of these would be fine..

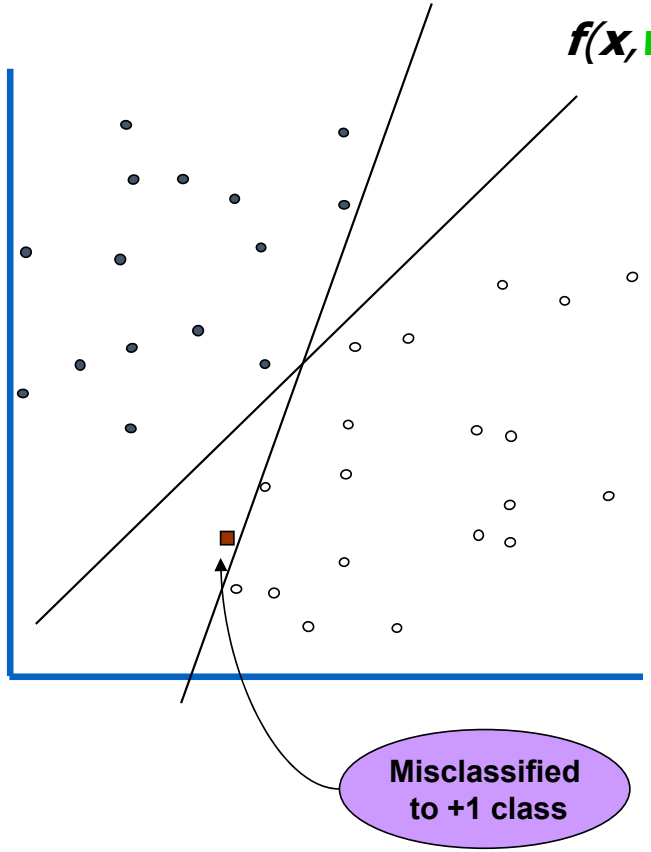
..but which is best?

Linear Classifiers



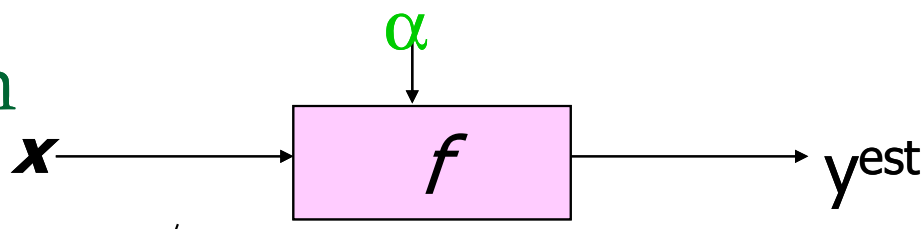
- denotes +1
- denotes -1

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$

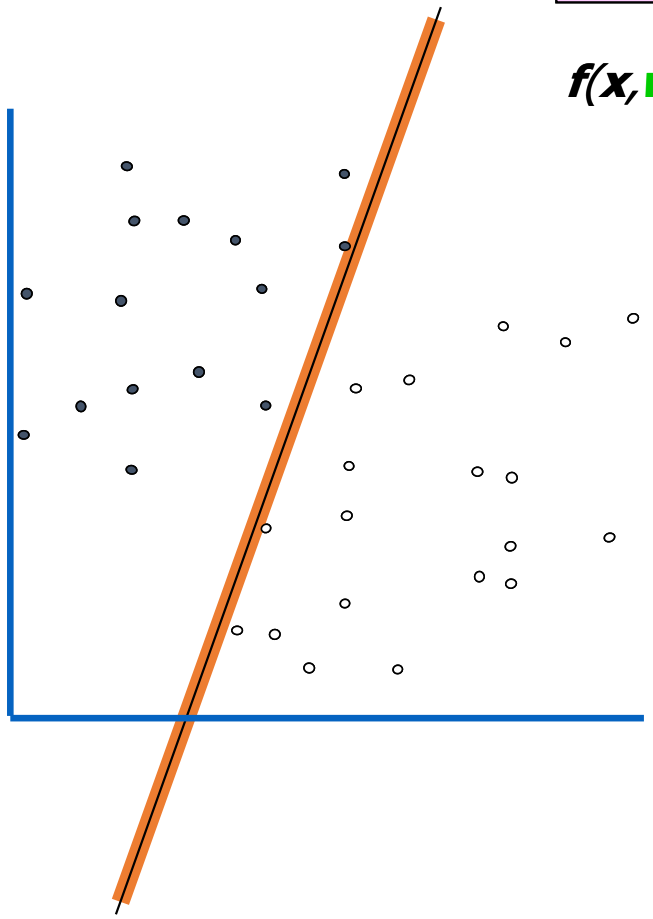


How would you classify this data?

Classifier Margin

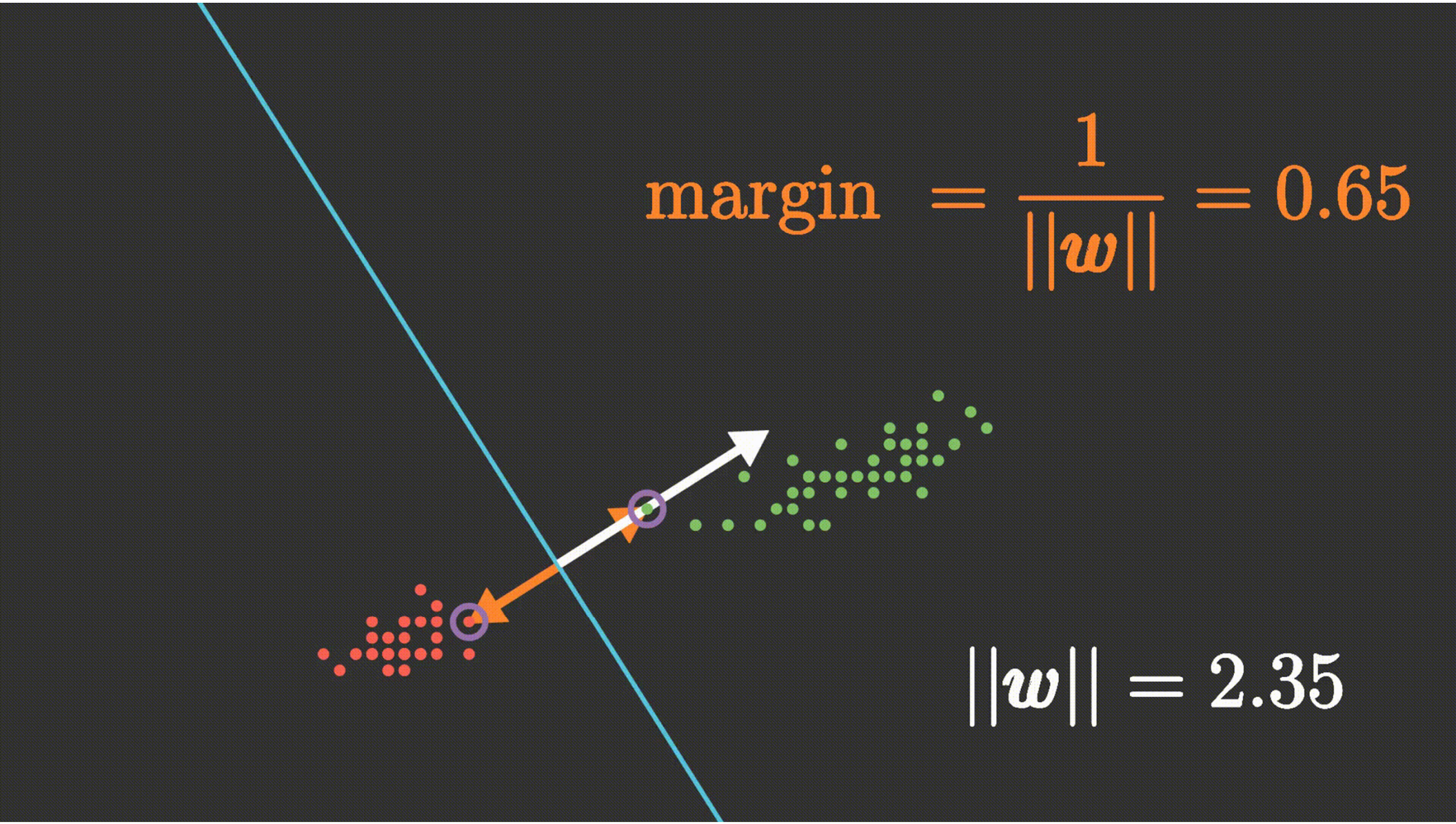


- denotes +1
- denotes -1

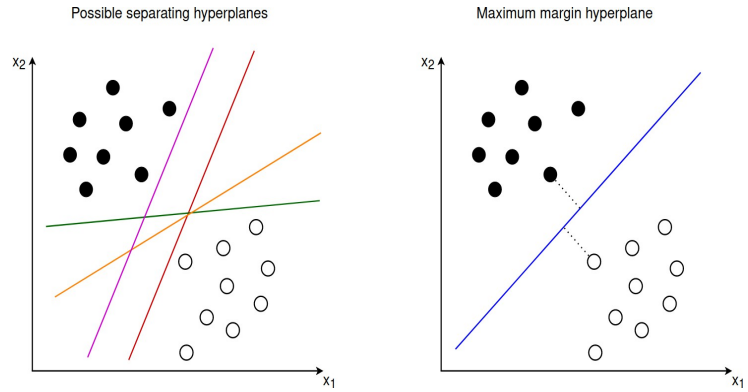


*f(**x**,**w**,b) = sign(**w** **x** + b)*

Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.


$\text{margin} = \frac{1}{||\boldsymbol{w}||} = 0.65$
$||\boldsymbol{w}|| = 2.35$

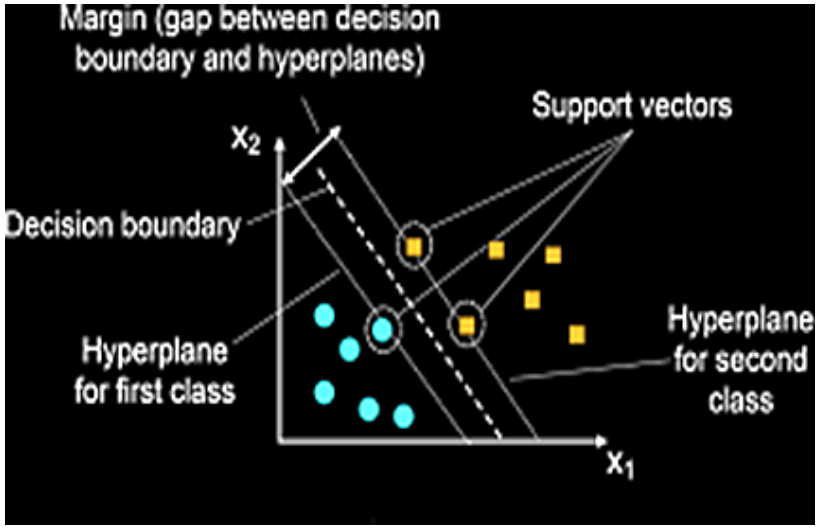
Motivation: Maximum Margin



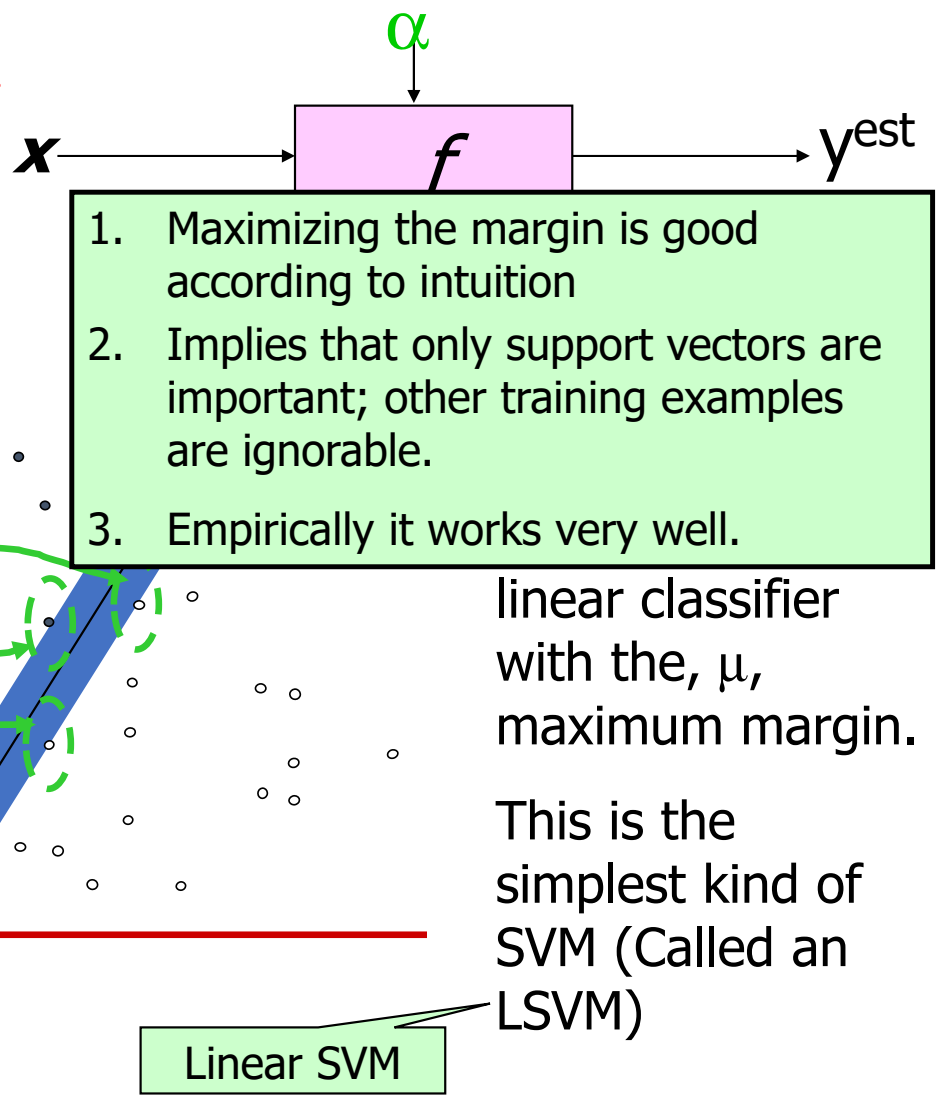
If the data are linearly separable:

- Infinitely many separating hyperplanes exist
- We want the one with the **largest margin**

Key idea: Choose the decision boundary that maximizes the distance to the closest data points. These closest points are called **support vectors.**



Maximum Margin



Properties of SVM

- **Flexibility in choosing a similarity function**
- **Sparseness of solution when dealing with large data sets**

  - only support vectors are used to specify the separating hyperplane
- **Ability to handle large feature spaces**

  - complexity does not depend on the dimensionality of the feature space
- **Overfitting can be controlled by soft margin approach**
- **Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution**
- **Feature Selection {Recursive Feature Elimination (SVM-RFE)}**

Functional Margin

For a data point ($\mathbf{x}_n$, t_n), define the functional margin:

$$\gamma_n = t_n\left(w^T X_n + w_0\right)$$

- $\gamma_n > 0$: correctly classified
- $\gamma_n < 0$: misclassified

The functional margin conveys two things: Direction:

- Whether the classification is correct.
- **Confidence:** How far the point is from the boundary in "score space."
  - Large positive γ_n $\Longrightarrow$ Confident correct prediction.
  - Small positive γ_n $\Longrightarrow$ Barely correct (near the boundary).
  - Negative γ_n $\Longrightarrow$ Wrong classification.

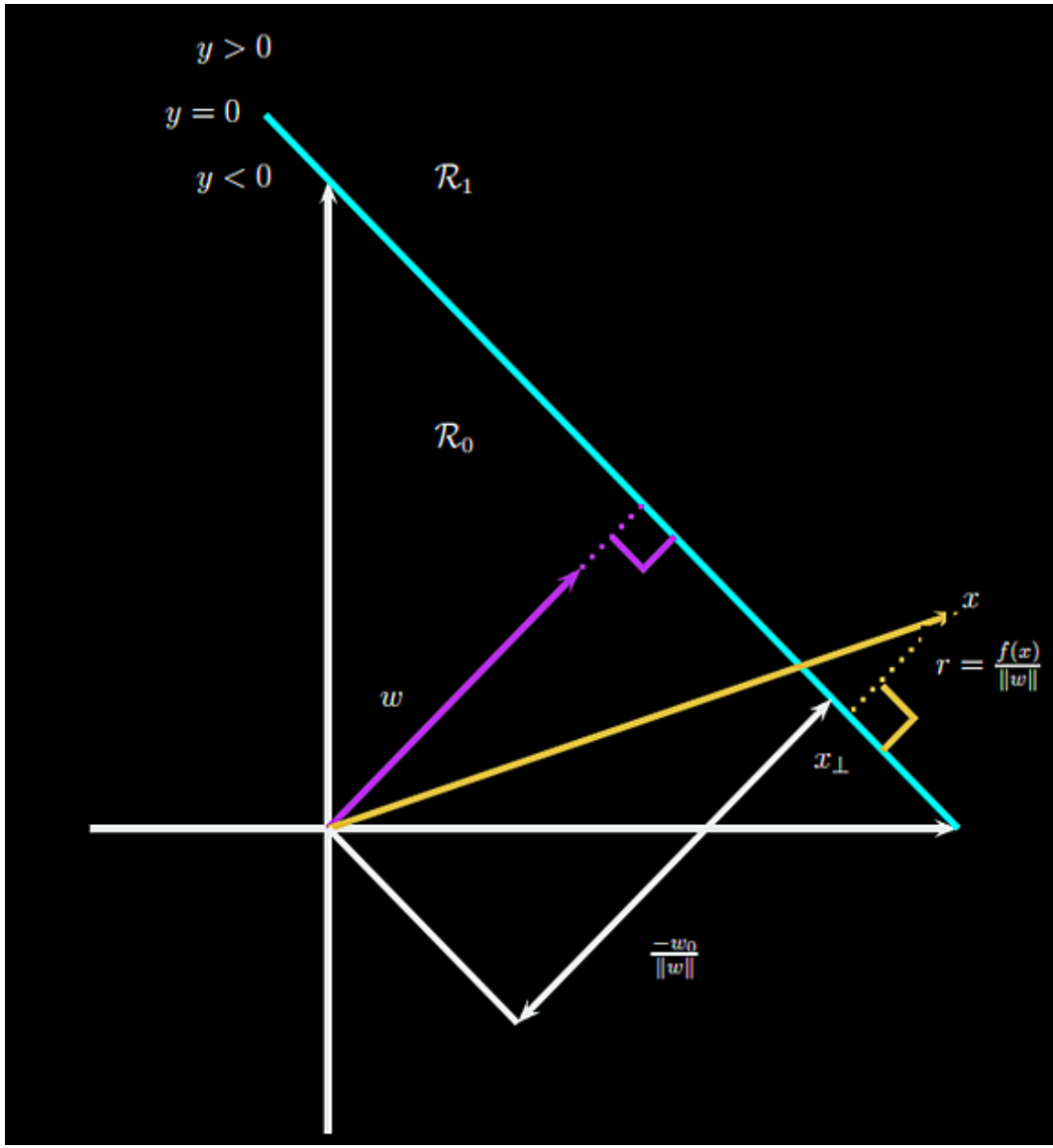
Note: The functional margin depends on the scaling of (w, w_0).

The large margin principle

Illustration of the geometry of a linear decision boundary in 2d. A point $\mathbf{x}$ is classified as belonging in decision region R_1 if $f(\mathbf{x}) > 0$, otherwise it belongs in decision region R_0; here $f(\mathbf{x})$ is known as a **discriminant function**.

The decision boundary is the set of points such that $f(\mathbf{x}) = 0$. $\boldsymbol{w}$ is a vector which is perpendicular to the decision boundary. The term w_0 controls the distance of the decision boundary from the origin. The signed distance of $\mathbf{x}$ from its orthogonal projection onto the decision boundary, $\boldsymbol{x}_\perp$, is given by $\boldsymbol{f}(\mathbf{x})/||\boldsymbol{w}||$.

Given X and W, find $x_\perp$



Geometric Margin

The perpendicular distance from a point $\mathbf{x}_n$ to the hyperplane is:

$$r = \frac{w^t x_n + w_0}{\|w\|}$$

The **geometric margin** is defined as:

$$\hat{y}_n = \frac{w^t x_n + w_0}{\| w \|}$$

Key Characteristics:

- **Real distance:** The perpendicular distance to the decision boundary.
- **Scale invariant:** Rescaling ($\mathbf{w}$, w_0) does not change $\hat{y}_n$.
- **Classification safety:**
  - Large value ⟹ Very safe classification.
  - Small value ⟹ Risky (near boundary).
  - Negative ⟹ Wrong classification.

Goal:

$$\max_{w, w_0} \min_{n} \hat{y}_n$$

In Support Vector Machines (SVM), the **functional margin** is a measure of both the correctness and confidence of a prediction for a given data point (x_i, y_i). Defined as $\hat{\gamma}_i = y_i(w^T x_i + b)$, it represents the signed distance scaled by the magnitude of the weight vector w, which is not necessarily normalized.

- **Definition & Formula:** The functional margin is computed as $\hat{\gamma}_i = y_i(w^T x_i + b)$, where y_i is the class label (± 1), w is the weight vector, x_i is the input data, and b is the bias.
- **Significance:** A positive value indicates the point is correctly classified, while a larger absolute value suggests greater confidence.
- **Limitations:** The functional margin can be arbitrarily increased by simply scaling the parameters w and b without changing the actual decision boundary.
- **Vs. Geometric Margin:** The functional margin is the raw score, whereas the **geometric margin** ($\gamma = \frac{\hat{\gamma}}{\|w\|}$) is the true geometric distance (normalized by $\|w\|$).
- **Optimization:** In SVM, the goal is to maximize the margin to achieve better generalization, typically by setting the functional margin of the closest points (support vectors) to 1 to manage the scaling issue.

Relationship Between Margins:

- **Functional Margin ($\hat{\gamma}$):** $y^{(i)}(w^T$
  can be scaled up by increasing
- **Geometric Margin (γ):** $\frac{\hat{\gamma}}{||w||}$. It
  to the hyperplane.

Ultimately, the goal is to maximize
performed using the **functional m**
stability.

Functional margin is used for SVM optimization because it simplifies the mathematical problem into a convex quadratic programming task, while geometric margin is harder to optimize directly due to non-convexity. By setting the functional margin to 1 (scaling w, b), we maximize the geometric margin indirectly by minimizing $\frac{1}{2}||w||^2$, avoiding the scaling invariance problem.

Why Functional Margin is Used Over Geometric Margin in Optimization:

- **Mathematical Convenience (Convexity):** The goal is to maximize the geometric margin $\gamma = \frac{\hat{\gamma}}{||w||}$ (where $\hat{\gamma}$ is functional margin). Directly maximizing this is complex. By fixing the functional margin of the support vectors to $\hat{\gamma} = 1$, the optimization objective becomes maximizing $\frac{1}{||w||}$, which is equivalent to minimizing $\frac{1}{2}||w||^2$.
- **Arbitrary Scaling Constraint:** Functional margin is not invariant to scaling; if we double the weight vector w and bias b, the functional margin doubles, but the actual decision boundary does not change. SVM uses this by fixing the functional margin of support vectors to 1, eliminating the scaling ambiguity and simplifying optimization.
- **Numerical Stability:** Minimizing $||w||^2$ is a convex optimization problem, which is numerically stable and guaranteed to find a global optimum, whereas optimizing for the raw geometric margin is more complex.

Why Maximum Margin is Powerful ?

The Intuition: Generalization and Robustness

Imagine two classifiers separating the same data:

- **Classifier A:** The boundary is very close to training points.
- **Classifier B (SVM):** The boundary is as far as possible from all points.

The "Noise" Test:

If noise or small perturbations appear in new, unseen data, which one survives?

Classifier B is more robust because it has a larger "safety buffer."

Key Takeaway:

Maximizing the geometric margin improves **generalization** to unseen data.

This is why SVM works effectively even with small datasets.

The large margin principle

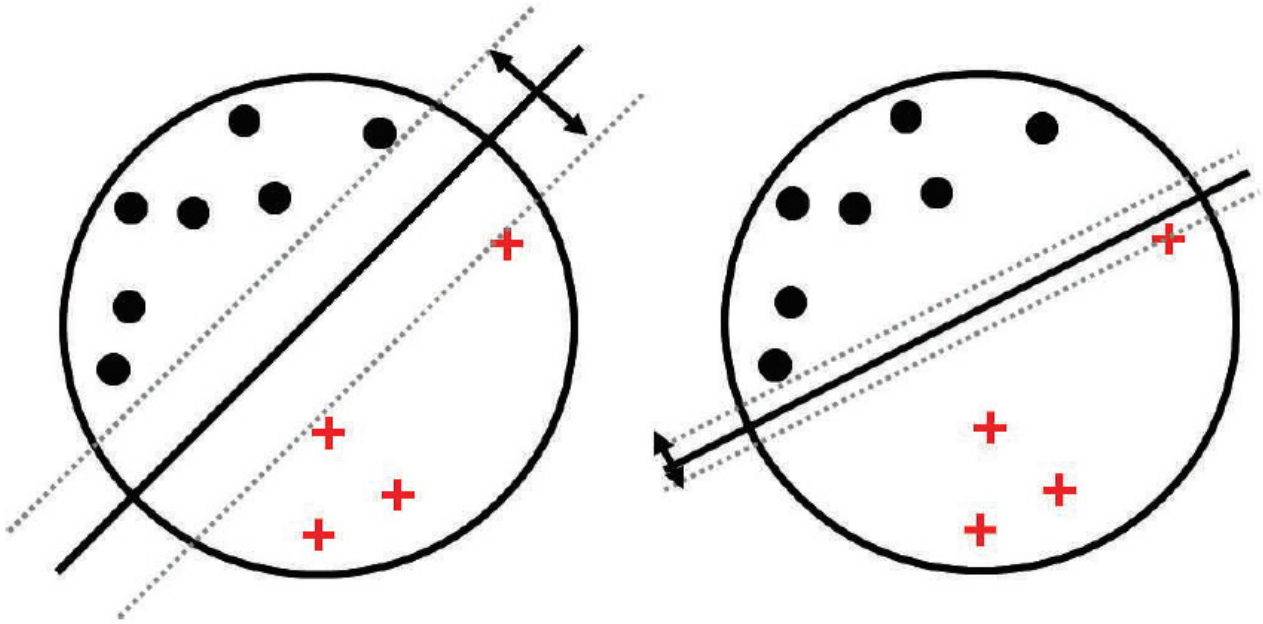


Illustration of the large margin principle
Left: a separating hyper-plane with large margin
Right: a separating hyper-plane with small margin

What is the Remaining issue ?

Our goal is now clear:

- We want to maximize the geometric margin.
- This ensures the best possible generalization to unseen data.

The Mathematical Problem: The objective function involves:

$$\max_{w,w_0} \left\{ \frac{1}{\|w\|} \min_n [t_n(w^T X_n + w_0)] \right\}$$

Why this is difficult:

- This is a non-convex optimization problem in its current form.
- The division by $\|\mathbf{w}\|$ makes the objective complex to differentiate and solve.
- The functional margin $\boldsymbol{t_n(\mathbf{w}^T\mathbf{x}_n + w_0)}$ is still sensitive to scaling.

Next: How can we use our "scaling freedom" to remove this fraction?

The Scaling Freedom

The decision boundary is

$$\boldsymbol{w^T x + w_0 = 0}$$

Now here is a key fact:

- If we multiply both **w** and w_0 by any positive constant *c*,

$$cw^T x + cw_0 = 0$$

- the hyperplane does not change.
- It is the exact same boundary.

So the classifier is scale invariant.

That means we are free to choose a convenient scaling.

Fix the Scale

We choose the scale so that the closest points satisfy:

$$t_n(w^T x_n + w_0) = 1$$

and all other points satisfy:

$$t_n(w^T x_n + w_0) \geq 1$$

This is called the **canonical representation**.

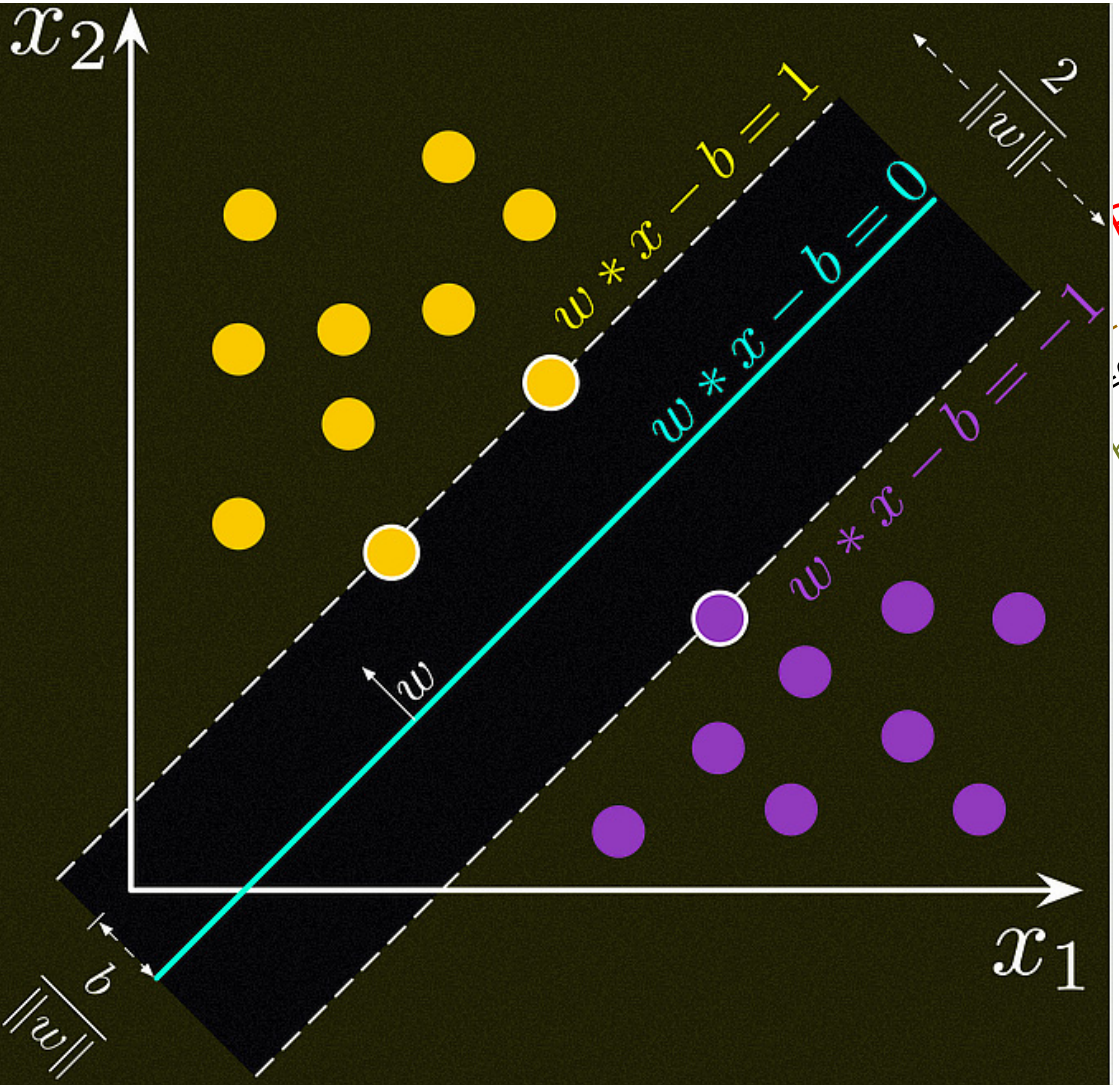
What did we just do? We forced the nearest points: the **support vectors** to lie exactly on two planes:

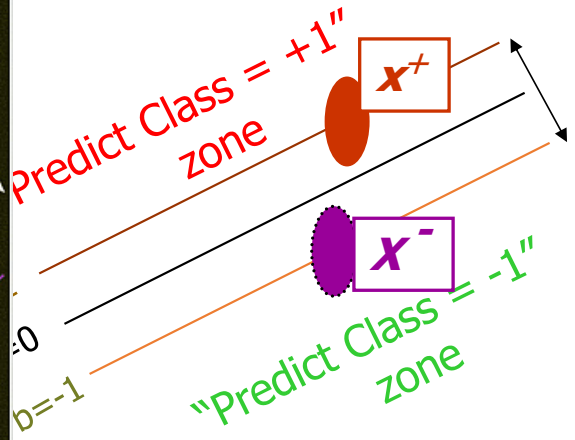
- $\boldsymbol{w^T x + w_0 = +1}$
- $\boldsymbol{w^T x + w_0 = -1}$

These are called **margin planes**. The decision boundary is exactly in the middle:

$$\boldsymbol{w^T x + w_0 = +0}$$

Linear SVM Mathematically





M=Margin Width

What we know:

- $\boldsymbol{w} \cdot \boldsymbol{x}^+ + b = +1$
- $\boldsymbol{w} \cdot \boldsymbol{x}^- + b = -1$
- $\boldsymbol{w} \cdot (\boldsymbol{x}^+ - \boldsymbol{x}^-) = 2$

$$M = \frac{(x^+ - x^-) \cdot w}{|w|} = \frac{2}{|w|}$$

Canonical Hyperplane

Thus we fix the scale by imposing

$$t_n(w^T x_n + w_0) \geq 1$$

Then the distance between the two margin planes is:

$$\textbf{Margin width} = \frac{2}{\| w \|}$$

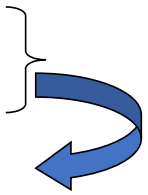
Maximizing the margin is equivalent to:

$$\min_{w} \frac{1}{2} \| w \|^2$$

Linear SVM Mathematically (*revisited – summary, till now*)

- Goal: **1) Correctly classify all training data**

$$wx_i + b \geq 1 \quad \text{if } y_i = +1$$
$$wx_i + b \leq 1 \quad \text{if } y_i = -1$$
$$y_i(wx_i + b) \geq 1 \quad \text{for all i}$$



2) Maximize the Margin $M = \frac{2}{|w|}$

same as minimize $\frac{1}{2} w^t w$

- **We can formulate a Quadratic Optimization Problem and solve for w and b**

- Minimize $\Phi(w) = \frac{1}{2} w^t w$

  subject to $y_i(wx_i + b) \geq 1 \quad \forall i$

Hard-Margin SVM: Primal Problem

The optimization problem becomes:

$$\underset{w,w_0}{\boldsymbol{min}} \quad \frac{\mathbf{1}}{\mathbf{2}} \| \boldsymbol{w} \|^{\mathbf{2}}$$

subject to:

$$\boldsymbol{t_n}\left(\boldsymbol{w^T x_n} + \boldsymbol{w_0}\right) \geq \mathbf{1}, \qquad \mathrm{n} = 1, \ldots, \mathrm{N}$$

This is in a convex quadratic programming problem domain, and is called the **primal problem of SVM**.

Constrained Optimization in SVMs

The hard-margin SVM is a constrained optimization problem:

$$\min_{w, w_0} \quad \frac{1}{2} \| w \|^2$$

subject to:

$$t_n\left(w^T x_n + w_0\right) \geq 1, \qquad n = 1, \ldots, N$$

Key idea:

- Objective: minimize model complexity (Small w: smoother boundary and Smooth boundary: better generalization)
- Constraints: enforce correct classification with margin

Why "Hard-Margin"?

- No mistakes are allowed.
- Every training point must satisfy the constraint.

So this works only when the dataset is perfectly linearly separable.

We solve this using **Lagrange multipliers**.

Lagrangian Formulation

Introduce Lagrange multipliers $\alpha_n \geq 0$ for each constraint.
The Lagrangian is:

$$L(w, w_0 \propto) = \frac{1}{2} \| w \|^2 - \sum_{n=1}^{N} \propto_n \left[t_n \left(w^T x_n + w_0 \right) - 1 \right]$$

Interpretation:

- α_n penalizes violations of the margin constraint
- Large α_n enforces strict satisfaction of the constraint

The Lagrangian, $L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i[y_i(w^T x_i + b) - 1]$, is minimized with respect to the primal variables (w, b) to find the best margin ar respect to the dual variables $(\alpha_i \geq 0)$ to enforce the classificat constraints.

At this point, the Lagrange multipliers are optimized to ensure strong duality, meaning the maximum of the dual problem equals the minimum of the primal problem.

Why it is a Saddle Point

1. **Opposing Optimization Goals (Mi**

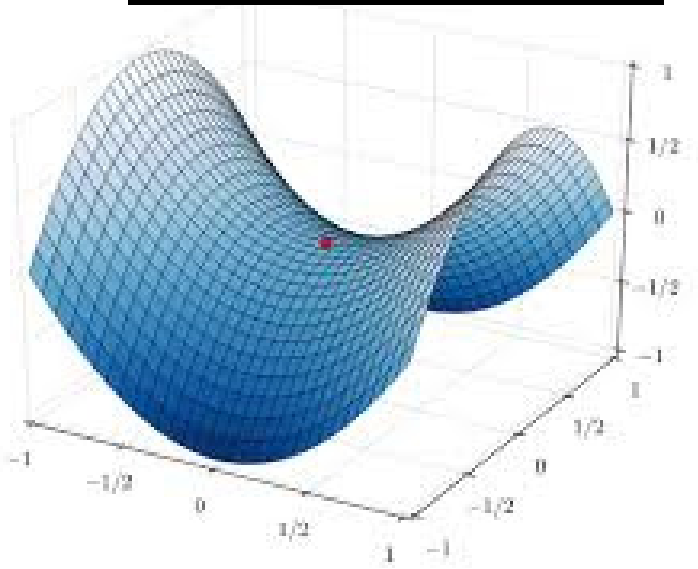
   - **Minimize w.r.t w, b:** To minimize adjusted to find the widest mar the cost function $\frac{1}{2}\|w\|^2$.
   - **Maximize w.r.t α:** The multipliers α_i act as penalties for v Lagrangian is maximized with respect to α to force the c ≥ 1 to be satisfied (i.e., making α_i large if a constraint is
   - **Result:** Because the objective is to find $min_{w,b} max_{\alpha} L(w$ is a minimum in one direction and a maximum in another, which is the definition of a saddle point.

- If a point is correctly classified and outside the margin $(y_i(w^T x_i + b) > 1)$, maximizing L w.r.t. α_i results in $\alpha_i = 0$.
- If a point is on the margin $(y_i(w^T x_i + b) = 1)$, α_i can be positive, and these are the support vectors.
- This "push and pull" ensures that only the support vectors (the "hardest" constraints) are used to define the decision boundary.

In summary, the saddle point structure arises directly from attempting to solve a constrained minimization problem by turning it into a mixed unconstrained optimization involving dual variables.

Saddle Point and Stationarity Conditions

$$z = x^2 - y^2$$



The solution is a saddle point:

- Minimize L w.r.t. $\mathbf{w}$, w_0
- Maximize L w.r.t. α_n

Setting derivatives to zero:

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^{N} \propto_n t_n x_n$$

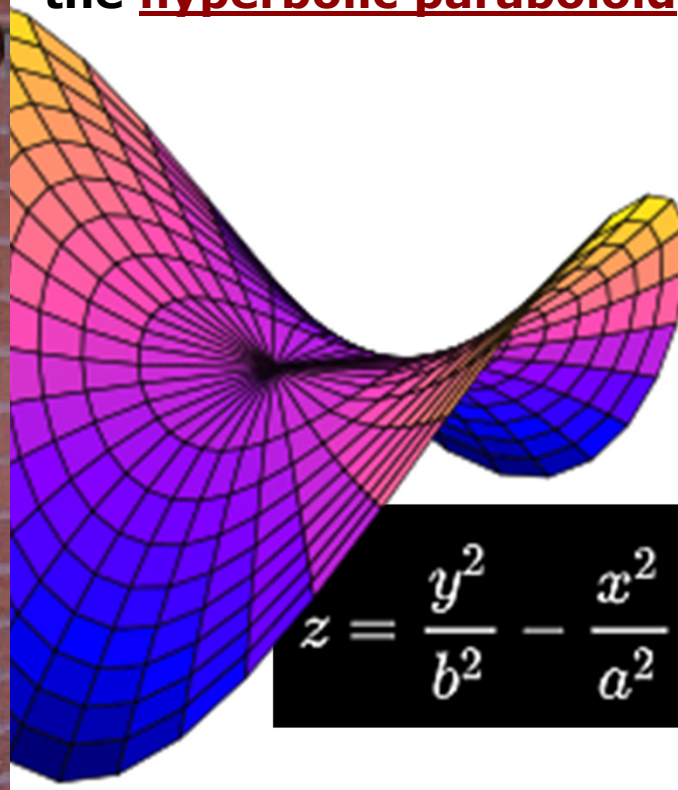
$$\frac{\partial L}{\partial w_0} = 0 \Rightarrow \sum_{n=1}^{N} \propto_n t_n = 0$$

Important consequence: The weight vector lies in the span of training data.

$f(x, y) = x^2 + y^3$ has a critical point at $(0, 0)$

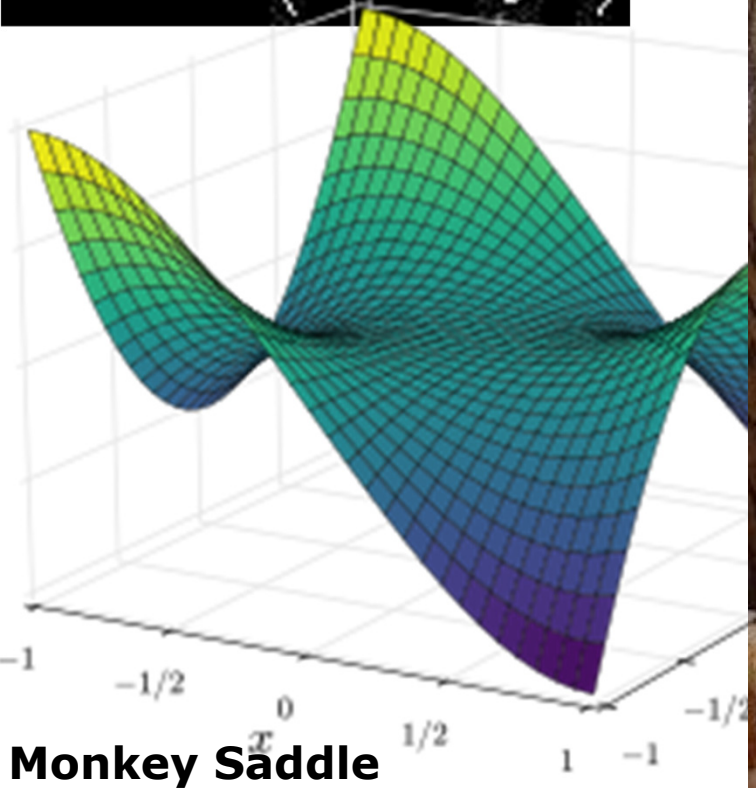
Elliptical hyperboloid

the [hyperbolic paraboloid]



$$z = \frac{y^2}{b^2} - \frac{x^2}{a^2}$$

$$a^2 z = x\left(x^2 - 3y^2\right)$$



Monkey Saddle

$$x = a\cosh v\cos\theta$$
$$y = b\cosh v\sin\theta$$
$$z = c\sinh v$$

Primal Problem objective	Dual problem		
	Objective	Constraints type	Variable sign
maximization	minimization	$\geq$	unrestricted
minimization	maximization	$\leq$	unrestricted

Key ideas

- Assign a dual variable for each primal (equality) constraint.
- Construct a dual constraint for each primal variable.
- The (column) constraint coefficients and the objective coefficient of the jth primal variable respectively define the left-hand an the right-hand sides of the jth dual constraint.
- The dual objective coefficients equal the right-hand sides of the primal constraint equations.

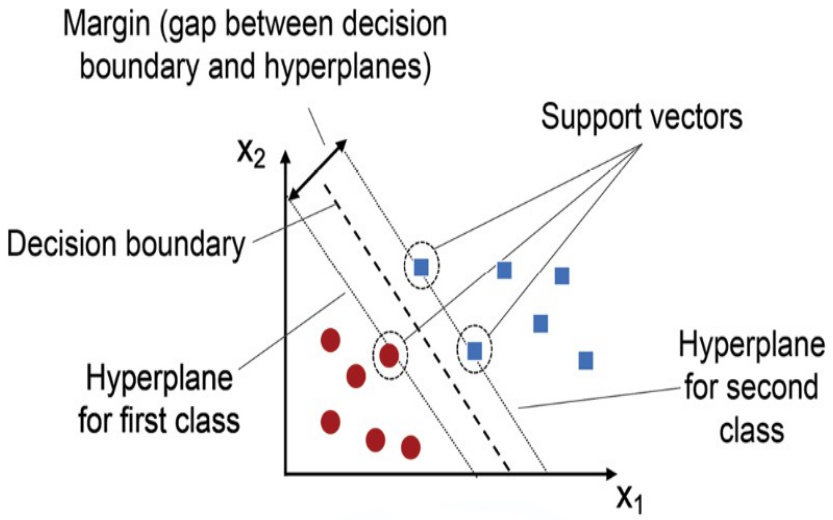
Support Vectors

- Data points with $\alpha_n > 0$ are **support vectors**
- Only support vectors contribute to the solution

Weight vector:

$$\boldsymbol{w} = \sum_{\boldsymbol{n \epsilon SV}} \boldsymbol{\alpha_n t_n x_n}$$

This leads to a **sparse representation.**



Only highlighted points define the boundary

The purpose of KKT conditions in SVM optimization is to convert the complex primal minimization problem (with inequality constraints) into a solvable dual problem, identifying the optimal Lagrange multipliers.

They are essential for defining support vectors, enabling the kernel trick, and identifying active constraints where the optimal hyperplane touches data points.

- **Identifying Support Vectors:** The KKT condition $\alpha_i \cdot (y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i) = 0$ identifies which data points are support vectors. If $\alpha_i > 0$, the corresponding data point is a support vector.
- **Dual Formulation Conversion:** KKT allows the primal SVM optimization (minimizing weight magnitude while satisfying constraint inequalities) to be converted to a dual problem that only depends on the inner product of input features, crucial for the "kernel trick".
- **Active Constraints Identification:** KKT conditions identify "active" constraints ($g(x) = 0$), meaning the margin constraints are met
- **Complementary Slackness:** The specific KKT condition that forces either the constraint to be active or its multiplier to be zero ($\alpha_i h_i(W) = 0$).
- **Lagrange Dual Formulation:** The reformulation of the SVM problem that KKT conditions make possible.

inactive constraints correspond

otimization, such as SVM, KKT

ufficient conditions to guarantee

ind.

Given the optimization pro

$$\min_{\mathbf{x}\in\mathbb{R}^2} f(\mathbf{x}$$

Define the **Lagrangian** as

$$\mathcal{L}(\mathbf{x},$$

Then $\mathbf{x}^*$ a local minimun

1. $\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*, \lambda^*) = \mathbf{0}$
2. $\lambda^* \geq 0$
3. $\lambda^* \, g(\mathbf{x}^*) = 0$
4. $g(\mathbf{x}^*) \leq 0$
5. Plus positive definite co

These are the **KKT condi**

KKT for multiple equality & inequality constraints

Given the constrained optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^2} f(\mathbf{x})$$

subject to

$$h_i(\mathbf{x}) = 0 \text{ for } i = 1, \ldots, l \quad \textbf{and} \quad g_j(\mathbf{x}) \leq 0 \text{ for } j = 1, \ldots, m$$

Define the **Lagrangian** as

$$\mathcal{L}(\mathbf{x}, \mu, \lambda) = f(\mathbf{x}) + \mu^t \, \mathbf{h}(\mathbf{x}) + \lambda^t \, \mathbf{g}(\mathbf{x})$$

Then $\mathbf{x}^*$ a local minimum $\iff$ there exists a unique λ^* s.t.

1. $\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*, \mu^*, \lambda^*) = \mathbf{0}$
2. $\lambda_j^* \geq 0$ for $j = 1, \ldots, m$
3. $\lambda_j^* \, g_j(\mathbf{x}^*) = 0$ for $j = 1, \ldots, m$
4. $g_j(\mathbf{x}^*) \leq 0$ for $j = 1, \ldots, m$
5. $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$
6. Plus positive definite constraints on $\nabla_{\mathbf{xx}}\,\mathcal{L}(\mathbf{x}^*, \lambda^*)$.

Karush–Kuhn–Tucker (KKT) Conditions

$$L(w, w_0 \propto) = \frac{1}{2} \| w \|^2 - \sum_{n=1}^{N} \alpha_n [t_n(w^T x_n + w_0) - 1]$$

The optimal solution must satisfy the KKT conditions:

1. **Primal feasibility:**

$$t_n(w^T x_n + w_0) \geq 1,$$

2. **Dual feasibility:**

$$\alpha_n \geq 0;$$

3. **Stationarity**

$$w = \sum_n \alpha_n t_n x_n; \quad \sum_n \alpha_n t_n = 0$$

4. **Complementary slackness**

$$\alpha_n [t_n(w^T x_n + w_0) - 1] = 0$$

$$\min_{w, w_0} \frac{1}{2} \| w \|^2$$

subject to:

$$t_n(w^T x_n + w_0) \geq 1$$

1. $\nabla_x \mathcal{L}(x^*, \lambda^*) = 0$
2. $\lambda^* \geq 0$
3. $\lambda^* g(x^*) = 0$
4. $g(x^*) \leq 0$
5. Plus positive definite constraints on $\nabla_{xx} \mathcal{L}(x^*, \lambda^*)$

Interpretation of KKT Conditions

$$\alpha_n[t_n(w^T x_n + w_0) - 1] = 0$$

$$t_n(w^T x_n + w_0) \geq 1; \qquad \alpha_n \geq 0;$$

The complementary slackness condition implies:

- $\alpha_n = 0$:

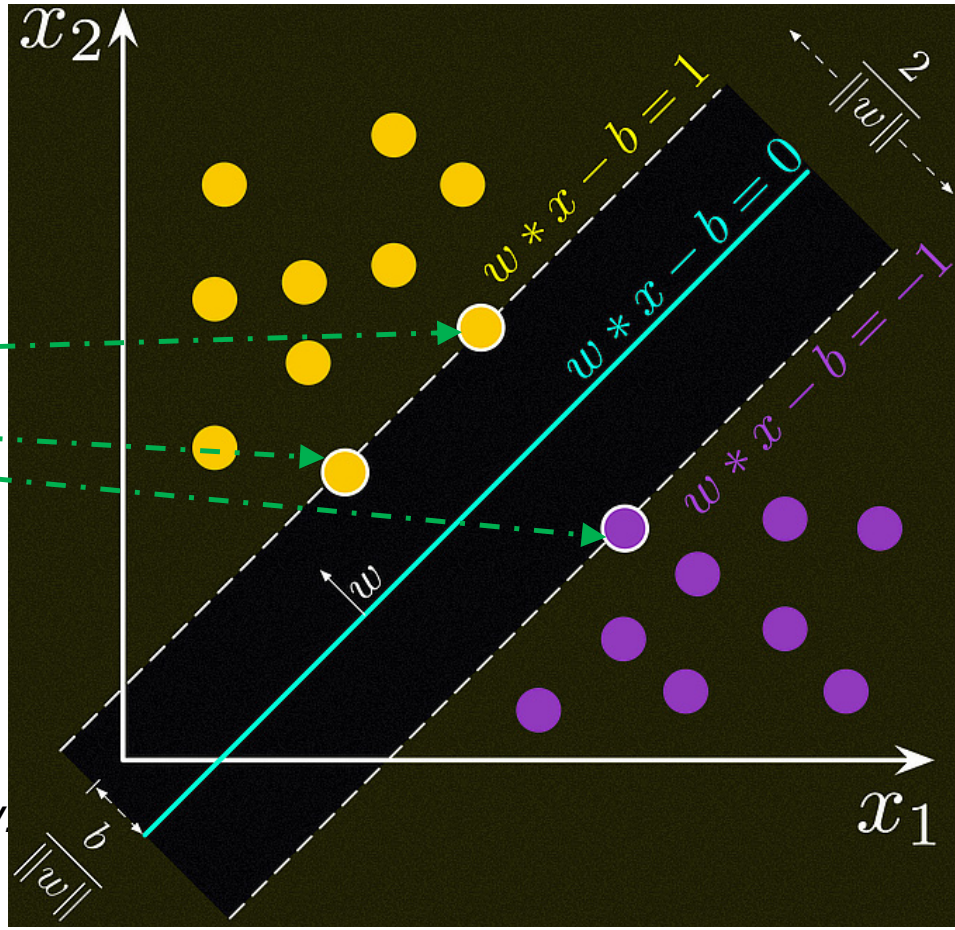
  $$t_n(w^T x_n + w_0) > 1,$$

  Point lies *outside* the margin (not a support vector)

- $\alpha_n > 0$:

  $$t_n(w^T x_n + w_0) = 1,$$

  Point lies *exactly on the margin (support vector)*



Key insight (Bishop):

Only support vectors influence the decision boundary.

Deriving the Dual Problem

Up to now, we have done three important things:

- We defined the maximum-margin classifier
- We constructed the Lagrangian
- We applied stationarity and KKT conditions

And from stationarity we obtained $\mathbf{w}$. We now substitute

$$\boldsymbol{w} = \sum_{\boldsymbol{n}=\mathbf{1}}^{\boldsymbol{N}} \boldsymbol{\alpha_n t_n x_n},$$

back into the Lagrangian and remove $\boldsymbol{w}$.

Dual Optimization Problem

$$w = \sum_{n=1}^{N} \alpha_n t_n x_n$$

After simplification, the dual optimization (derive yourself):

$$L(w, w_0, \alpha) = \frac{1}{2} \| w \|^2 - \sum_{n=1}^{N} \alpha_n \left[t_n \left(w^T x_n + w_0 \right) - 1 \right]$$

$$\max_{\alpha} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m x_n^T x_m$$

subject to:

$$\alpha_n \geq 0, \qquad \sum_n \alpha_n t_n = 0$$

$$\min_{w, w_0} \frac{1}{2} \| w \|^2$$

$$t_n \left(w^T x_n + w_0 \right) \geq 1, \qquad n = 1, \ldots, N$$

Observe that the above is "*w*" free

Important Insight: The data appears only through $x_n^T . x_m$ which is the dot product between two training points.

Primal problem	Dual problem
maximize $c^T x$	minimize $b^T y$
subject to $Ax = b$	subject to $A^T y \geq c$
$x \geq 0$	

Decision Function

$$w = \sum_{n=1}^{N} \alpha_n t_n x_n$$

The classifier is:

$$y(x) = \sum_{n=1}^{N} \alpha_n t_n x_n^T x + w_o$$

We compute the bias term w_0 using any support vector x_s. Since support vectors lie exactly on the margin:

$$t_s\left(w^T x_s + w_0\right) = 1$$

Rearranging,

$$w_0 =$$

Final Classification Rule: **sign** $(y(x))$

SVM PROBLEM SOLVED ?

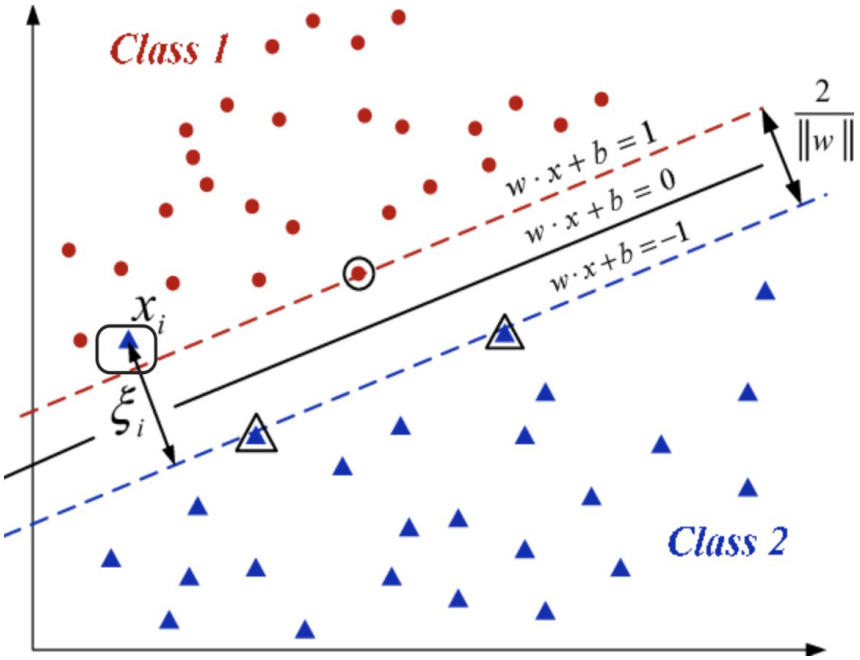
Nonseparable Data

In practice, perfect separation is rare.

Introduce slack variables $\xi_n \geq 0$:

$$t_n(w^T x_n + w_0) \geq 1 - \xi_n$$

- **Case 1:** $\xi_n = 0$
  Correct classification outside or on the margin. This is the **perfect case**.
  - **Case 2:** $0 < \xi_n < 1$
    Classification is **correct but risky**.
  - **Case 3:** $\xi_n \geq 1$
    Misclassified point.



Slack variables: margin violations and misclassification (w_0 represented by b)

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

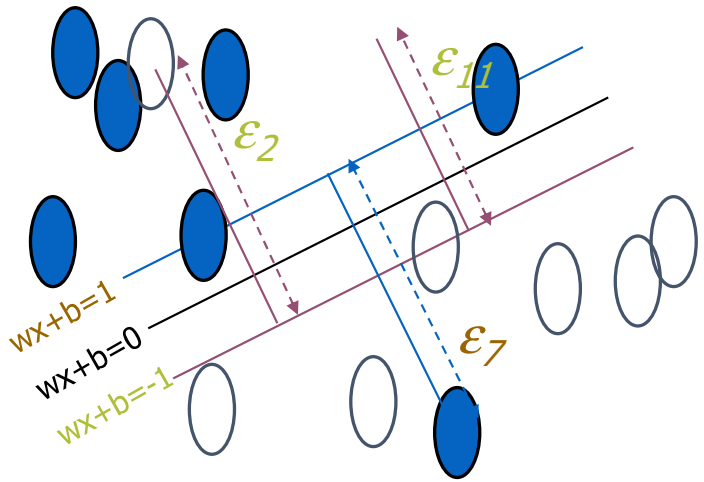
- **The new formulation incorporating slack variables:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- **Parameter *C* can be viewed as a way to control overfitting.**

Soft Margin Classification

Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples.



What should our quadratic optimization criterion be?

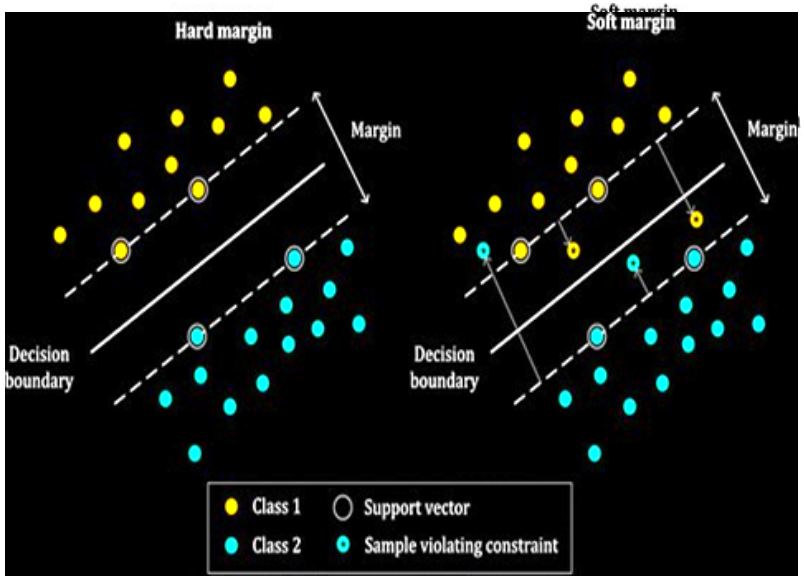
Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

Soft-Margin SVM

The optimization problem becomes:

$$\min_{w,w_0} \frac{1}{2} \| w \|^2 + C \sum_{n=1}^{N} \xi_n$$



subject to:

$$t_n(w^T x_n + w_0) \geq 1 - \xi_n$$

and,

$$\xi_n \geq 0$$

Small C: wide margin, more errors
Large C : narrow margin, fewer errors

The parameter *C* controls the trade-off between margin width and classification error.

Step 1: Express slack variable

From the constraint:

$$t_n . y(x_n) \geq 1 - \xi_n$$

Rearrange:

$$\xi_n \geq 1 - t_n . y(x_n)$$

Since $\xi_n \geq 0$, the smallest possible ξ_n is:

$$\xi_n = max(0, 1 - t_n y(x_n))$$

Step 2: Substitute into objective

$$\min_{w,w_0} \frac{1}{2} \| w \|^2 + C \sum_{n=1}^{N} \xi_n$$

Now substitute into: $C \quad \sum \xi_n$

We get:

$$C. \sum_{n=1}^{N} \max(0, (1 - t_n . y(x_n)))$$

So the SVM optimization becomes:

$$\min_{w} \frac{1}{2} \| w \|^2 + C \sum_{n=1}^{N} max(0, 1 - t_n y(x_n))$$

This is the final unconstrained SVM objective.

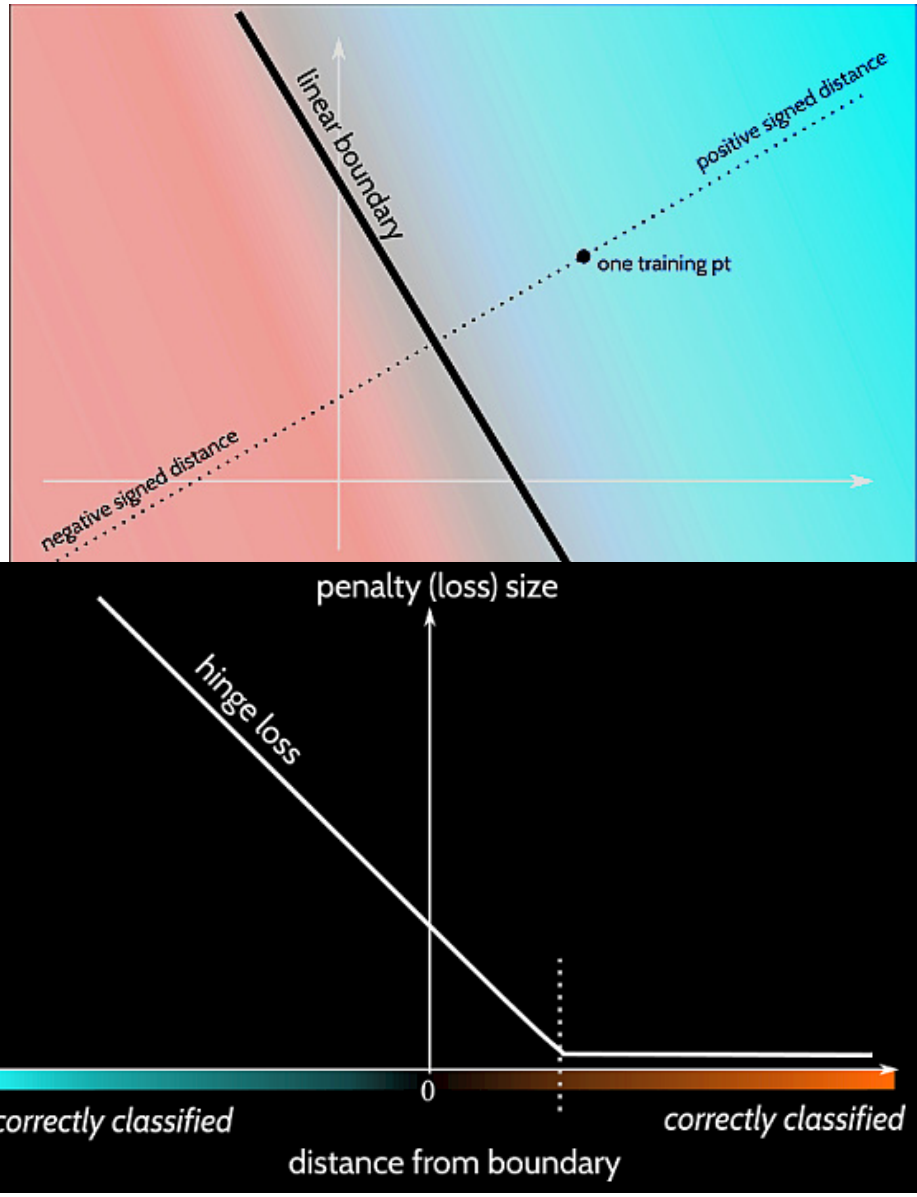
Hinge Loss Interpretation

Equivalent unconstrained objective:

$$\min_{w} \frac{1}{2} \| w \|^2 + C \sum_{n=1}^{N} \max(0, 1 - t_n y(x_n))$$

Hinge loss:

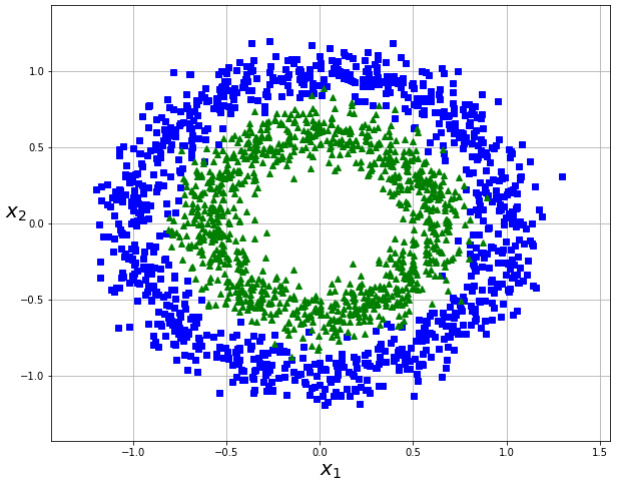
$$\ell(t, y) = \max(\ 0, \quad (1 - t.y)\ )$$



- **Case 1: Correct and Confident**
  If $ty \geq 1$, then $\ell = 0$. No penalty.
- **Case 2: Inside the Margin**
  If $0 < ty < 1$. Then loss is positive.
- **Case 3: Misclassification**
  If $ty < 0$. Loss becomes large. Heavy penalty.

What is the limitation now?

- So far SVM still produces a linear decision boundary.
- But what if the data is not linearly separable even with slack variables?
- Example: two concentric circles.
  No straight line can separate them.



Nonlinear data: no linear separator exists

Hard Margin v.s. Soft Margin (repeating)

- **The old formulation:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

- **The new formulation incorporating slack variables:**

Find $\mathbf{w}$ and b such that
$\mathbf{\Phi(w)} = \frac{1}{2}\,\mathbf{w}^T\mathbf{w} + C.\Sigma\xi_i$ is minimized and for all $\{(\mathbf{x_i}, y_i)\}$
$y_i\,(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

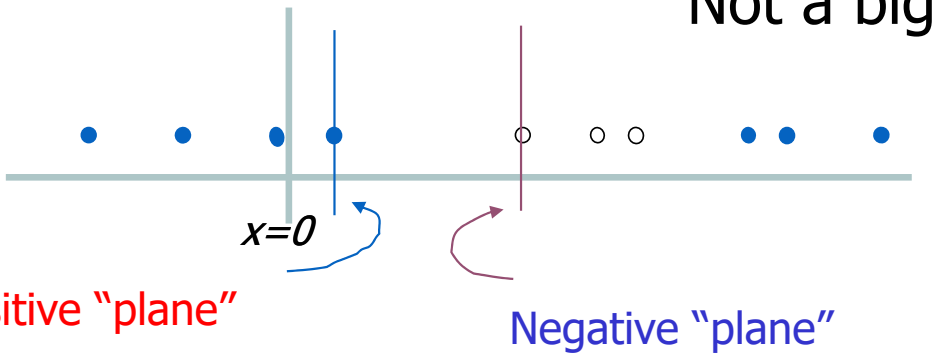
$$\min_w \frac{1}{2} \| w \|^2 + C \sum_{n=1}^{N} \max(0, 1 - t_n y(x_n))$$

- **Parameter *C* can be viewed as a way to control overfitting.**

Hard 1-dimensional Dataset

What would SVMs do with this data?

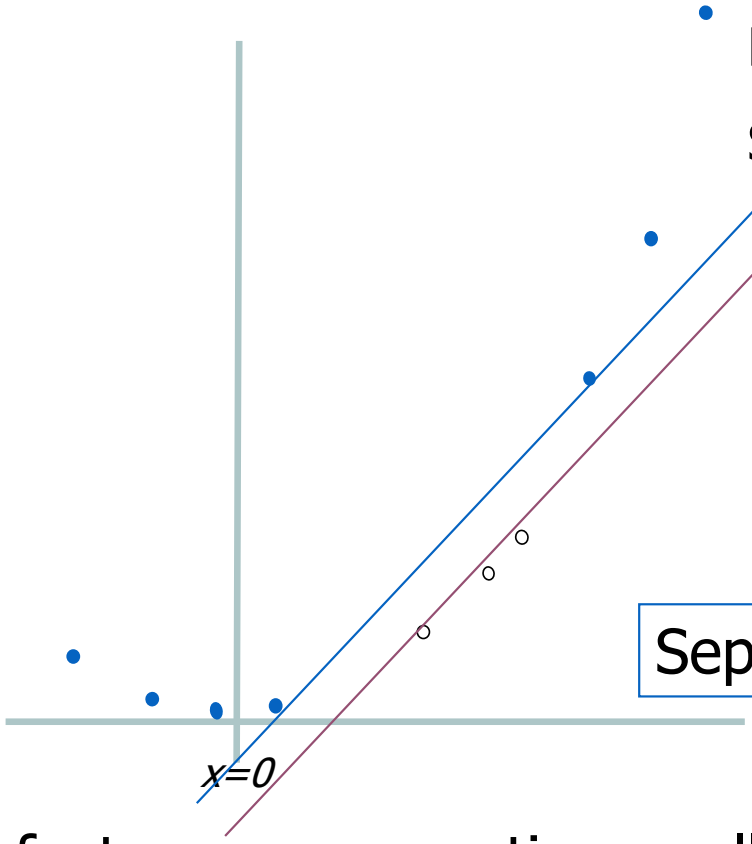
Not a big surprise



Doesn't look like slack variables will work/save us this time…

taken from Andrew W. Moore

Hard 1-dimensional Dataset



M*ake up a new feature!*

Sort of…

… computed from original feature(s)

$$\mathbf{z}_k = (x_k, x_k^2)$$

Separable! MAGIC!

New features are sometimes called *basis functions.*

Now drop this "augmented" data into our linear SVM.

taken from Andrew W. Moore

Kernels and Linear Classifiers

Let $\vec{x} = [\vec{x}_1, \vec{x}_2] \in \mathbb{R}^2$ be a vectorial represenation of object $x \in \mathcal{X}$

Let $\phi : \mathcal{X} \to \mathcal{K} \subset \mathbb{R}^3$ feature map be given by

$$\phi(\vec{x}) \doteq [\vec{x}_1, \vec{x}_2^2, \vec{x}_1\vec{x}_2]^T \in \mathcal{K} \subset \mathbb{R}^3$$

Def. Feature space: $\mathcal{K}$

We will use linear classifiers in this feature space.

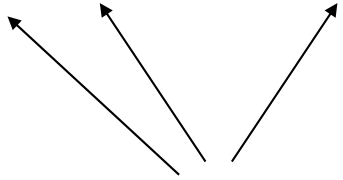
In the original space $\mathbb{R}^2$ for a given $\mathbf{w} \in \mathbb{R}^3$ the decision surface is:

$$\tilde{X}_0(\mathbf{w}) = \{\vec{x} \in \mathbb{R}^2 \mid w_1\vec{x}_1 + w_2\vec{x}_2^2 + w_3\vec{x}_1\vec{x}_2 = 0\}$$

- This is nonlinear in $\vec{x} \in \mathbb{R}^2$
- This is linear in the feature space $\phi(\vec{x}) \in \mathcal{K} \subset \mathbb{R}^3$

Kernels and Linear Classifiers

$$\phi(\vec{x}) \doteq [\phi_1(\vec{x}), \phi_2(\vec{x}), \phi_3(\vec{x})] \doteq [\vec{x}_1, \vec{x}_2^2, \vec{x}_1\vec{x}_2]^T$$



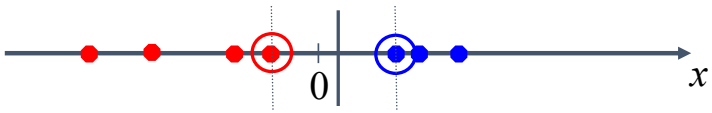
Feature functions

- We seek for a small set of basis vectors $\{\phi_i\}$ which allows perfect discrimination between the classes in $\mathcal{X}$ (**Feature selection**)

- If we have too many features $\Rightarrow$ overfitting can happen.

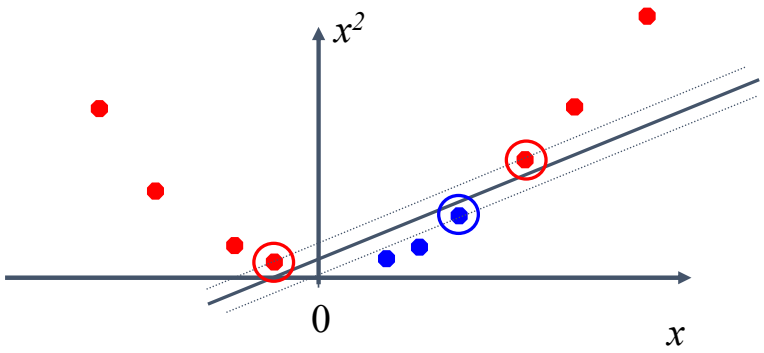
Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:



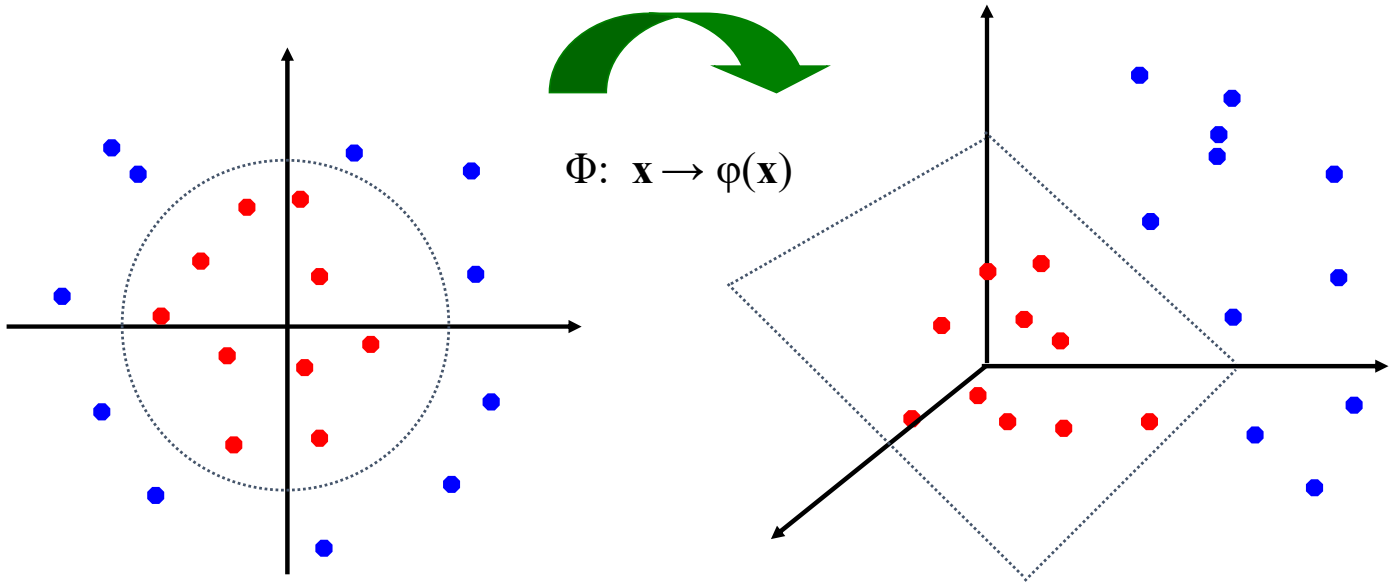
- But what are we going to do if the dataset is just too hard?

- How about… mapping data to a higher-dimensional space:

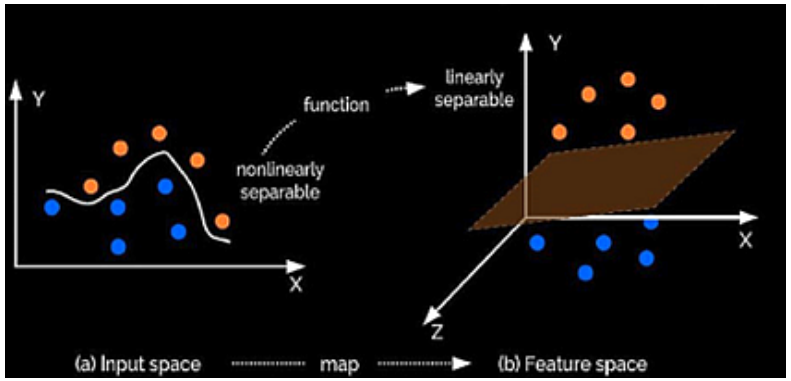


Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



The Idea of Feature Mapping



Suppose the original input is:

$$x = (x_1, x_2)$$

Now define a nonlinear feature map:

$$\boldsymbol{\varphi}(\boldsymbol{x}) = \left(\boldsymbol{x_1^2}, \quad \sqrt{\boldsymbol{2}}\boldsymbol{x_1 x_2}, \quad \boldsymbol{v x_2^2}\right)$$

Nonlinear → map → linearly separable

- Original input space: R^2
- Feature space: R^3
- We created **new nonlinear features** from the old coordinates

Key Idea

Linear in feature space ⇒ nonlinear in input space

The Key Observation

Look carefully at the prediction function:

$$y(x) = \sum_{n \epsilon SV} \alpha_n t_n \, X_n^T \, X + w_0$$

- We never directly manipulate the coordinates of *x*
- The classifier uses *x* only through terms of the form: $\boldsymbol{X_n^T . X}$
- This is simply the **dot product** (inner product) between two vectors

Crucial Point - ??

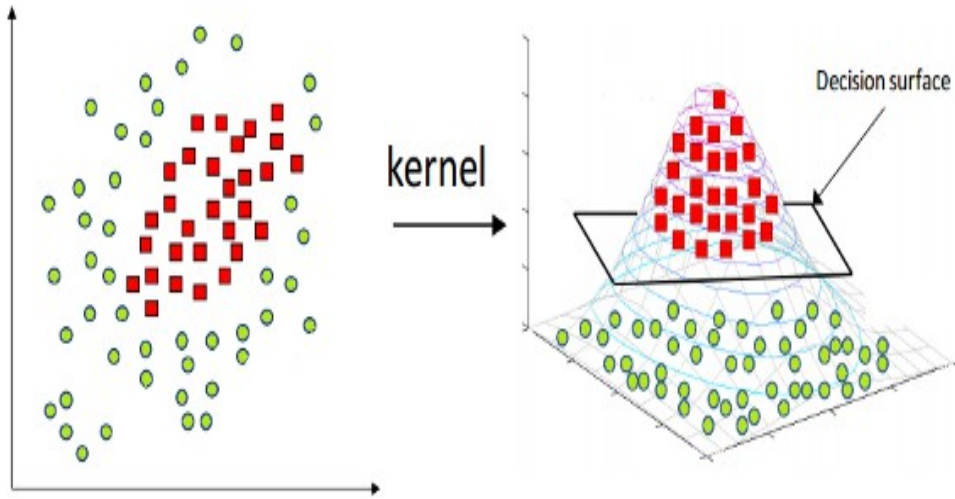
The Kernel Trick

Suppose we transform the input into a new feature space: $\phi(\mathbf{x})$
Instead of computing $\phi(\mathbf{x})$ directly as it is expensive operation, we compute only:

$$\boldsymbol{\phi}(\boldsymbol{x_n})^T \boldsymbol{\phi}(\boldsymbol{x_m})$$

and we define:

$$\boldsymbol{K}(\boldsymbol{x_n}, \boldsymbol{x_m}) = \boldsymbol{\phi}(\boldsymbol{x_n})^T \boldsymbol{\phi}(\boldsymbol{x_m})$$



Compute inner product without explicit mapping

This function *k* is called a **kernel function**. Now we replace:

$$\boldsymbol{X_n^T X_m} \Rightarrow \boldsymbol{k}(\boldsymbol{X_n}, \boldsymbol{X_m})$$

This allows nonlinear decision boundaries without explicit feature mapping.

The "Kernel Trick"

- **To produce linear separability in Higher Dimension, the linear classifier relies on dot product between vectors $K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$**
- **If every data point is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the dot product becomes:**

$$K(\mathbf{x}_i,\mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$$

- **A *kernel function* is some function that corresponds to an inner product in some expanded feature space.**
- **Example:**

  2-dimensional vectors $\mathbf{x} = [x_1 \; x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j) = (1 + \mathbf{x}_i^T\mathbf{x}_j)^2$,

  Need to show that $K(\mathbf{x}_i,\mathbf{x}_j) = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$:

$$K(\mathbf{x}_i,\mathbf{x}_j) = (1 + \mathbf{x}_i^T\mathbf{x}_j)^2,$$

$$= 1 + x_{i1}^2x_{j1}^2 + 2\,x_{i1}x_{j1}\,x_{i2}x_{j2} + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

$$= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\,x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^T \, [1 \;\; x_{j1}^2 \;\; \sqrt{2}\,x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}]$$

$$= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j), \quad \text{where } \varphi(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\,x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$$

What Functions are Kernels?

- **For some functions $K(x_i,x_j)$ checking that**

  $K(x_i,x_j) = \varphi(x_i)^T\varphi(x_j)$ can be cumbersome.

- **Mercer's theorem:**

  Every semi-positive definite symmetric function is a kernel

- **Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix *(skipping the theory)*:**

K=

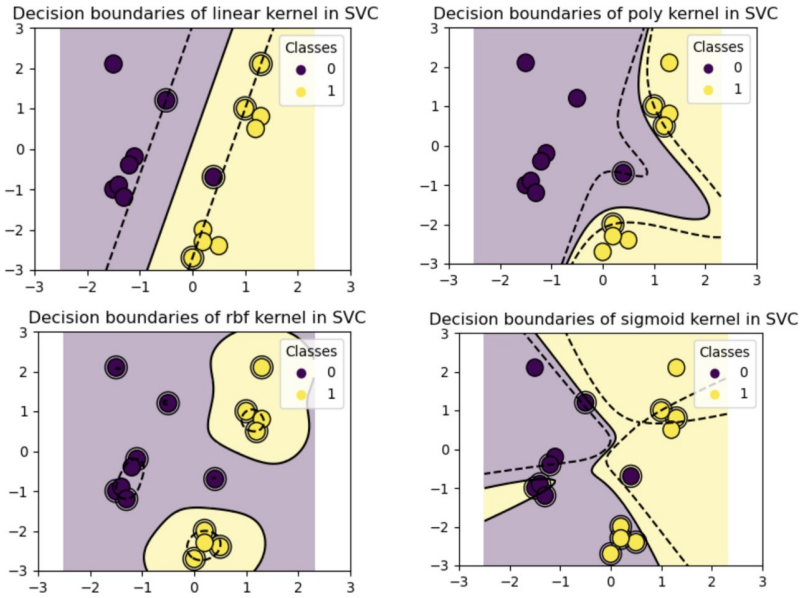
$K(x_1,x_1)$	$K(x_1,x_2)$	$K(x_1,x_3)$	…	$K(x_1,x_N)$
$K(x_2,x_1)$	$K(x_2,x_2)$	$K(x_2,x_3)$		$K(x_2,x_N)$
…	…	…	…	…
$K(x_N,x_1)$	$K(x_N,x_2)$	$K(x_N,x_3)$	…	$K(x_N,x_N)$

Examples of Kernel Functions

- Linear: $K(\mathbf{x_i},\mathbf{x_j})= \mathbf{x_i}^T\mathbf{x_j}$
- Polynomial of power p: $K(\mathbf{x_i},\mathbf{x_j})= (1+ \mathbf{x_i}^T\mathbf{x_j})^p$
- Gaussian (radial-basis function network):

$$K(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2})$$

- Sigmoid: $K(\mathbf{x_i},\mathbf{x_j})= \tanh(\beta_0\mathbf{x_i}^T\mathbf{x_j} + \beta_1)$



Different kernels ⇒ different decision boundaries

Non-linear SVMs Mathematically

- **Dual problem formulation:**

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(x_i, x_j)$ is maximized
and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

- **The solution is:**

$$f(x) = \Sigma\alpha_i y_i K(x_i, x_j) + b$$

- **Optimization techniques for finding α_i's remain the same!**

To Summarize:

$$\max_{\alpha} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m x_n^T x_m$$

$$w = \sum_{n=1}^{N} \alpha_n t_n x_n$$

The objective is to maximize $W(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$ over the search space, where K is the kernel function.

Dimensionality:

The search space is N-dimensional (not feature dimension in x), where N is the number of training samples.

$$b = w_0$$

$$= t_s - \sum_{n} \alpha_n t_n x_n^T x_s$$

- **Soft Margin SVM**: The search space is a bounded "box" in n-dimensional space, defined by:
  - $0 \leq \alpha_i \leq C$ for all i, where C is the regularization hyperparameter.
  - The same equality constraint: $\sum_{i=1}^{n} \alpha_i y_i = 0$.

$$f(\mathbf{x}) = \Sigma \alpha_i y_i K(\mathbf{x_i}, \mathbf{x_j}) + b$$

The Gram matrix K=Xi.Xj

- X1·X1 = 2 × 2 + 2 × 2 = 4+4 = 8
- X1·X2 = 2 × 1 + 2 ×(−1) = 2−2 = 0
- X1·X3 = 2 ×(−2) + 2 ×(−2) = −4−4 = −8
- X2·X2 = 1 × 1 + (−1)×(−1) = 1+1 = 2
- X2·X3 = 1 ×(−2) + (−1)×(−2) = −2+2 = 0
- X3·X3 = (−2) 2 + (−2) 2 = 4+4 = 8

	X_i	y_i
	(2, 2)	+1
	(1, −1)	−1
	(−2, −2)	−1

$$K = \begin{bmatrix} 8 & 0 & -8 \\ 0 & 2 & 0 \\ -8 & 0 & 8 \end{bmatrix}$$

$$Q_{ij} = y_i y_j K_{ij}$$

$$Q = \begin{bmatrix} 1 \times 1 \times 8 & 1 \times (-1) \times 0 & 1 \times (-1) \times (-8) \\ (-1) \times 1 \times 0 & (-1) \times (-1) \times 2 & (-1) \times (-1) \times 0 \\ (-1) \times 1 \times (-8) & (-1) \times (-1) \times 0 & (-1) \times (-1) \times 8 \end{bmatrix} = \begin{bmatrix} 8 & 0 & 8 \\ 0 & 2 & 0 \\ 8 & 0 & 8 \end{bmatrix}$$

$$\mathcal{D}(\alpha) = \alpha_1 + \alpha_2 + \alpha_3 - \frac{1}{2}\left[8\alpha_1^2 + 2\alpha_2^2 + 8\alpha_3^2 + 2(8\alpha_1\alpha_3)\right]$$

αi ≥ 0 and α1−α2−α3 = 0

SELF-STUDY (not part of exams) – latest SVM models

- **Fuzzy-SVM**
- **Latent SVM**
- **MI-SVM**
- **Fuzzy Twin Proximal SVM (2026): A variant of the Twin SVM (TWSVM)**
- **DeepF-SVM, CNN-SVM**
- **QSVM**

Also, SVM for regression and outlier detection