Lecture 14

Support Vector Machines

SVM is a supervised learning algorithm used for classification and regression tasks.

It finds the optimal boundary (hyperplane) that best separates classes in the feature space.

The goal is to maximize the margin between the closest points (support vectors) of the classes and the decision boundary.

Hyperplane: A decision boundary that separates classes. In 2D, it's a line; in higher dimensions, a hyperplane.

Margin: The distance between the hyperplane and the closest training samples from both classes. SVM maximizes this margin to improve generalization.

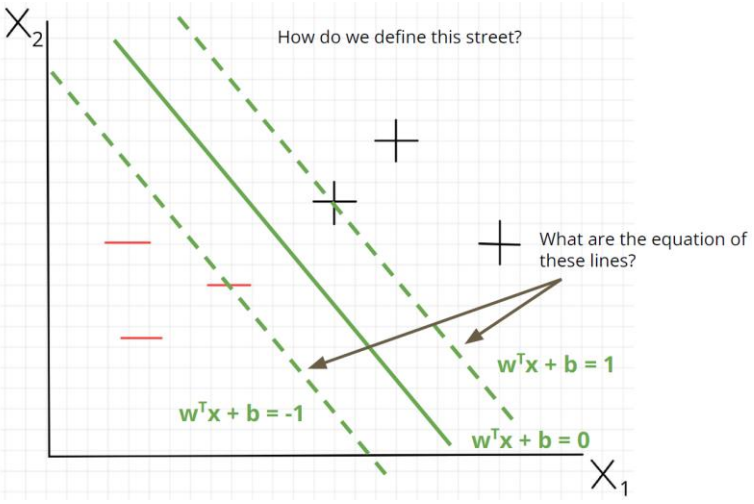
Support Vectors: The data points closest to the hyperplane that define the margin. These points are critical to determining the position of the hyperplane.

Now,

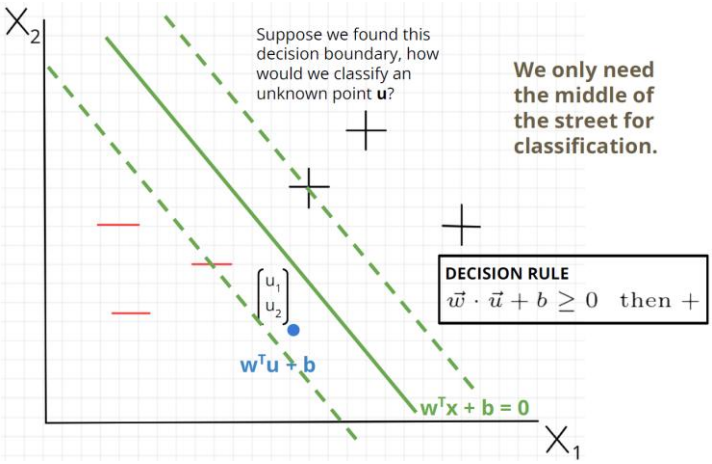
1. Data points $X = (x_1, x_2)$ live in feature space.
2. A hyperplane in 2D is a line, defined as: $W^T X + b = 0$

   where:

   - W is a weight vector normal (perpendicular) to the hyperplane,
   - b is the bias term (offset).
3. Points for which $W^T X + b > 0$ lie on one side of the hyperplane (e.g. class +1), and those with < 0 on the other side (e.g. class -1).
4. Points exactly on the hyperplane satisfy $W^T X + b = 0$

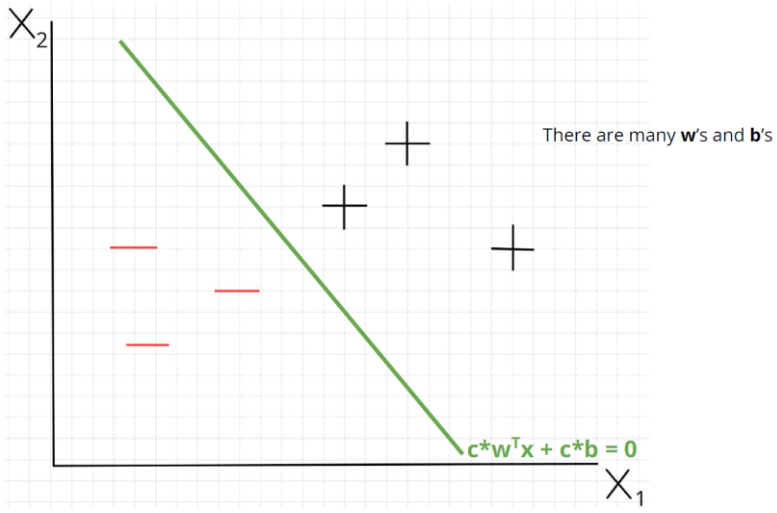


The middle solid green line is the **decision boundary**: $W^T X + b = 0$
The two dashed lines on either side define the **margin boundaries**: $W^T X + b = \pm 1$
The **margin** is the <u>distance</u> between these two dotted lines or the "street" separating the two classes.



Given the decision boundary $W^T X + b = 0$, a new point $u = (u_1, u_2)$ is classified according to: <u>If</u> $\boldsymbol{W^T u + b \geq 0 \rightarrow Class = 1}$, <u>else</u> $\boldsymbol{Class = -1}$

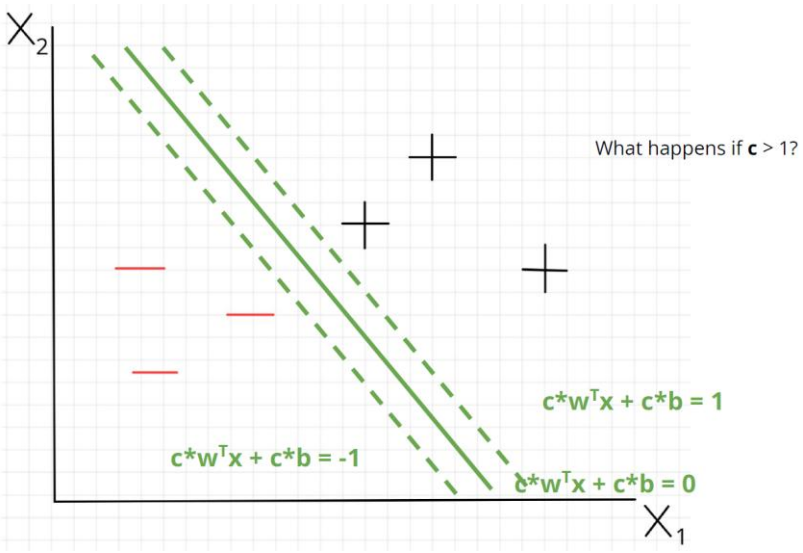
The dashes and crosses representing classes are separated by this linear boundary.
Only the sign of $\boldsymbol{W^T u + b}$ is needed, not its magnitude, to assign class labels.



Multiplying both W and b (both are not unique) by a positive scalar c produces infinitely many equivalent hyperplanes:

$$cW^T X + cb = 0$$

Such hyperplanes represent the same decision boundary (same line) but scaled differently. Choice of c relates directly to margin size and normalization of W.

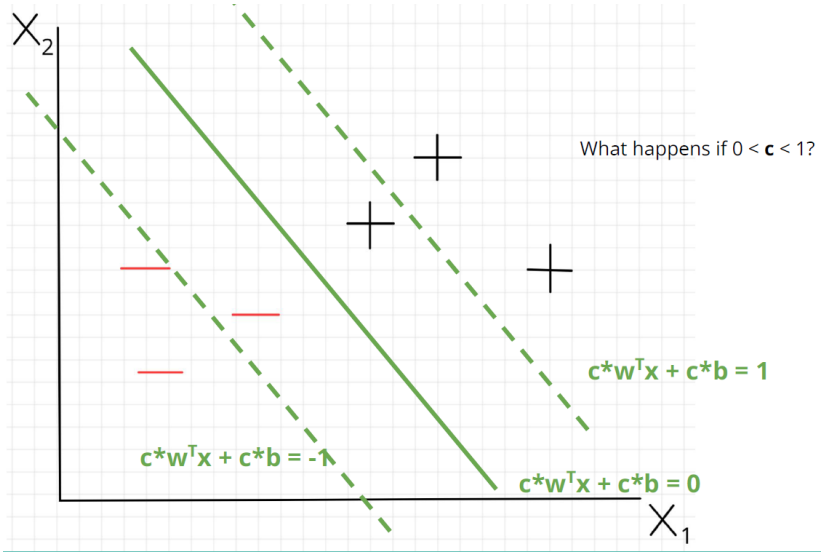


For $c > 1$, the margin boundaries:

$$cW^T X + cb = \pm 1$$

become closer to the decision boundary because:

Since c is larger, the set of points satisfying $cW^T X + cb = \pm 1$ moves inward. Margin width decreases i.e. the "street" narrows.



When c is between 0 and 1, margin boundaries move away from the decision boundary. This increases the margin width i.e. the street becomes wider.
SVM optimizes to maximize the margin, adjusting W and b accordingly.

Two Goals of SVM:

1. Data Points Must Be Classified Correctly:
    a. The SVM aims to find a decision boundary (a hyperplane) that correctly separates the training data points by their class labels.
    b. This means:
        i. All positive class points should lie on one side of the hyperplane.
        ii. All negative class points should lie on the other side.
    c. Formally, for each training point X_i with label $y_i \in \{+1, -1\}$ the constraint is:
    $$y_i(W^T X_i + b) \geq 1$$
    d. This ensures no points are misclassified (i.e., no point is on the wrong side of the margin).
2. Data Points Cannot Be in the Street (Margin):
    a. The SVM maximizes the margin, which is the distance between the decision boundary and the closest data points from any class.
    b. The "street" is the space (area) between the two dashed green lines:
    $$W^T X + b = +1 \; and \; W^T X + b = -1$$
    c. No training points should lie between these two boundaries (points exactly on these lines are the support vectors).
    d. The margin width M is:
    $$M = \frac{2}{\|W\|}$$
    where $\|W\|$ is the Euclidean norm of the weight vector.
    e. Maximizing this margin increases the generalization ability of the classifier thus leading to better performance on unseen data.

Learning W and b:

As defined earlier:

- W is the weight vector normal (perpendicular) to the decision hyperplane.
- b is the bias or intercept term, which shifts the hyperplane away from the origin.
- The decision boundary equation: $W^T X + b = 0$
- Goal 1: Separate the data: correctly classify all points (or accept some errors in non-separable cases).
- Goal 2: Maximize margin: maximize the distance between the decision boundary and the closest data points (support vectors).

Primal Form to Dual Form

We will formulate is as an optimization problem:

Given labelled training samples (X_i, y_i) with $y_i \in \{+1, -1\}$:

$$\begin{cases} y_i\,(W^T X_i + b) \geq 1 \\ maximixe\ M = \dfrac{2}{\|W\|} \end{cases} , ensure\ correct\ classification\ with\ margin$$

Reformulate maximizing margin:

Maximizing $\frac{2}{\|W\|}$ is equivalent to minimizing $\frac{1}{2}\|W\|^2$ for mathematical convenience:

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

Subject to:

$$y_i\,(W^T X_i + b) \geq 1, \qquad \forall i$$

Which means Each training point must be classified correctly with the above margin.

Solving the Optimization:

The above problem is **Convex Quadratic Programming** Problem

It can be solved using:

- Lagrange multipliers to form dual problem
- Sequential Minimal Optimization (SMO) algorithm for efficiency.

How to enforce constraints during optimization?

One way: Penalize any violation of constraints by adding a penalty term to the objective.

If a point violates its constraint (e.g., margin condition is not met), add a penalty proportional to how much it violates.

Lagrange multipliers α_i play the role of these "penalty weights":

- If constraint is satisfied, penalty is zero or small, so the term doesn't reduce the objective.
- If constraint is violated, penalty grows to force the optimizer to fix the violation.

Forming the Lagrangian for SVM:

The Lagrangian L combines:

- The original objective function:

$$\frac{1}{2}\left|\left|W\right|\right|^2$$

- Minus the sum over all training points of multipliers times the constraint residuals:

$$L(w, b, \alpha) = \frac{1}{2}\left|\left|W\right|\right|^2 - \sum_i^n \alpha_i \left[y_i \left(W^T X_i + b\right) - 1\right]$$

Intuition is:

- If the constraint $y_i \left(W^T X_i + b\right) \geq 1$ is satisfied $y_i \left(W^T X_i + b\right) - 1 \geq 0$, this term is non-negative.
- For $\alpha_i \geq 0$, a positive violation term reduces L, pushing the solution towards satisfying the constraint.
- This way, the optimization problem balances:
    - Finding the smallest $\left|\left|w\right|\right|^2$ (largest margin)
    - While strictly respecting the margin constraints.

Conditions for Optimality

We find the saddle point by taking derivatives and setting to 0:

$$\frac{\partial L}{\partial W} = 0$$

$$\frac{\partial \left(\frac{1}{2}||W||^2 - \sum_i^n \alpha_i \left[y_i \left(W^T X_i + b\right) - 1\right]\right)}{\partial W} = \frac{\partial (W^T W)}{\partial W} - \frac{\partial (\sum_i^n \alpha_i y_i W^T x_i)}{\partial W} + \dots$$

(Terms not involved with W are represented by …)

$$\frac{\partial L}{\partial W} = W - \sum_i^n \alpha_i y_i x_i = 0$$

$$W = \sum_i^n \alpha_i y_i x_i$$

- W can be expressed as a combination of support vectors.
- Here α_i are Lagrange multipliers, zero for non-support vectors.

Now w.r.t bias

$$\frac{\partial L}{\partial b} = 0$$

The only term is $-\sum_i \alpha_i y_i b_i$

$$= \frac{\partial (-\sum_i^n \alpha_i y_i b_i)}{\partial b}$$

$$\frac{\partial L}{\partial b} = -\sum_i^n \alpha_i y_i = 0$$

$$\sum_i^n \alpha_i y_i = 0$$

Therefore, the conditions for optimality are:

$$\begin{cases} W = \sum_i^n \alpha_i y_i x_i \\ \sum_i^n \alpha_i y_i = 0 \end{cases}$$

Plug back into Lagrangian to get the dual problem:

$$\max_{a} \sum_{i} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

Subject to:

$$\alpha_i \geq 0, \quad \sum_{i}^{n} \alpha_i y_i = 0$$

- The solution α_i is nonzero only for support vectors i.e. the training points closest to the decision boundary.
- This means only support vectors influence the decision boundary.

Final classification decision rule:

Once the α_i values are found, the decision function for any input x is:

$$f(x) = \sum_{i}^{n} \alpha_i y_i \langle x_i, x \rangle + b$$

and classification is:

$$Predict\ y = +1\ if\ f(x) \geq 0, \qquad -1\ otherwise$$

Summary:

- The weights W of the SVM are expressed as a linear combination of support vectors weighted by $\alpha_i y_i$.
- The training reduces to solving the dual problem using Lagrange multipliers.
- Only those training points that lie on or inside the margin (support vectors) matter.
- The decision rule depends on computing inner products between test points and support vectors plus bias b.

Soft Margin SVM

When data is not perfectly separable, introduce slack variables $\xi_i \geq 0$ to allow margin violations:

$$y_i\ (W^T\ X_i + b) \geq 1 - \xi_i$$

Optimize:

$$\min_{w,b,\xi} \frac{1}{2}\left|\left|W\right|\right|^2 + C\sum_i \xi_i$$

Where C controls the trade-off between margin size and penalty for misclassification

- Large C means the model penalizes violations heavily, striving for fewer errors, possibly reducing the margin.
- Small C allows more margin violations and a wider margin, tolerating some misclassifications for better generalization.

ξ_i measures how much the i-th point violates the margin constraints:

- $\xi_i = 0$ if the point is correctly classified and outside the margin.
- $0 < \xi_i \leq 1$ if it lies inside the margin but on the correct side.
- $\xi_i > 1$ if it is misclassified.

So, forming the Lagrangian:

$$L(w, b, \xi, \alpha, \mu) = \frac{1}{2}\left|\left|W\right|\right|^2 + C\sum_i^n \xi_i - \sum_i^n \alpha_i[y_i\ (W^T\ X_i + b) - 1 + \xi_i] - \sum_i^n \mu_i\xi_i$$

Taking Partial Derivatives and setting to 0:

Done for optimality conditions

w.r.t W

$$\frac{\partial L}{\partial W} = W - \sum_i^n \alpha_i\ y_i x_i = 0$$

$$W = \sum_i^n \alpha_i\ y_i x_i$$

w.r.t b

$$\frac{\partial L}{\partial b} = -\sum_i^n \alpha_i\ y_i = 0; \quad \therefore \sum_i^n \alpha_i\ y_i = 0$$

w.r.t ξ_i

$$\frac{\partial L}{\partial \xi_i} = \frac{\partial (C \sum_i^n \xi_i - \sum_i^n \alpha_i [y_i \, (W^T \, X_i + b) - 1 + \xi_i] - \sum_i^n \mu_i \xi_i)}{\partial \xi_i} = \; C - \alpha_i - \mu_i = 0$$

$$\alpha_i - \mu_i = C$$

Since $\mu_i \geq 0$ and $\alpha_i \leq C$

Substituting and writing Dual Problem:

Using above conditions:

$$\max_a \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i x_j \rangle$$

Subject to:

$$0 \leq \alpha_i \leq C, \qquad \sum_i^n \alpha_i y_i = 0$$

Key Difference:

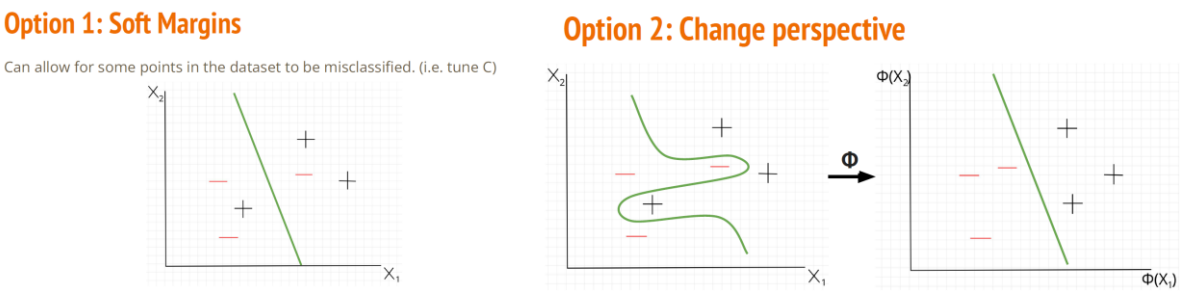
- Larger C restricts α_i to be close to hard margin (less tolerance for misclassification).
- Smaller C allows larger slack, more margin violations, and potentially better generalization.

The Limitation of Linear SVM

- Linear SVM finds a linear decision boundary in the original input space.
- But many real-world problems have nonlinear boundaries: no line or hyperplane can separate classes.
- How can SVM classify in such cases?

Ways to Solve:

1. Soft Margins (As just mentioned above)
    a. Allows some misclassification or margin violations.
    b. Helps when classes overlap slightly but still linearly separable with some slack
2. Change the Perspective
    a. *Idea:* Map the original data into a higher-dimensional space (called feature space) via a transformation $\phi(x)$
    b. In that transformed space, data might be linearly separable.
    c. However, explicitly transforming data can be computationally expensive or infeasible.



The Idea of Change of Perspective:

- Let $\phi: \mathbb{R}^d \to \mathbb{R}^D$ be a transformation that maps original feature vectors X in low-dimensional space to a high (possibly infinite) dimensional feature space.
- In this new space, the data might become linearly separable, allowing SVM to find a linear hyperplane.
- Example: mapping 2D points onto a 3D paraboloid to separate them linearly.

The Challenge:

- Computing $\phi(x)$ can be:
  - Computationally expensive, especially when D is very large or infinite.
  - Memory intensive, needing to store large vectors.
- So, explicitly transforming data is often infeasible.

Kernel Trick

Instead of computing $\phi(x)$ explicitly, leverage the fact that SVM relies on dot products between data points in the feature space.

Recall SVM decision function:

$$f(x) = \sum_{i}^{n} \alpha_i y_i \langle x_i, x \rangle + b$$

It can be written as

$$f(x) = \sum_{i}^{n} \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b$$

Now it depends **only on dot products** $\langle \phi(x_i), \phi(x) \rangle$ in the transformed space.

If we can compute these dot products directly without explicitly computing $\phi(X)$ we avoid the high-dimensional mapping cost.

Formal Definition of a Kernel Function:

A kernel function K corresponds to the inner product in feature space:

$$K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

K is defined on the original input space $\mathbb{R}^d$, but behaves like an inner product in $\mathbb{R}^D$.
By using $K(x_i, x_j)$, we implicitly compute similarity in the high-dimensional feature space.

Usefulness:

- *Computational efficiency*: Compute K in input space without ever constructing $\phi(X)$.
- *Infinite dimensional embeddings*: Some kernels correspond to infinite-dimensional ϕ, e.g., Gaussian RBF kernel.
- *Flexibility:* Allows SVM to handle complex nonlinear relations by choosing different kernels.

Some Common Kernels and their Transformations:

Kernel Name	**Formula**	**Intuition**
Linear	$K(x_i, x_j) = x_i^T x_j$	No mapping; standard dot product (linear SVM)
Polynomial	$K(x_i, x_j) = (x_i^T x_j + 1)^n$	Maps into polynomial feature space of degree n
Gaussian RBF/ Radial	$K(x_i, x_j) = \exp\left(\frac{\left\|\left\|x_i^T x_j\right\|\right\|^2}{2\sigma^2}\right)$	Infinite-dimensional mapping weighting similarity by distance