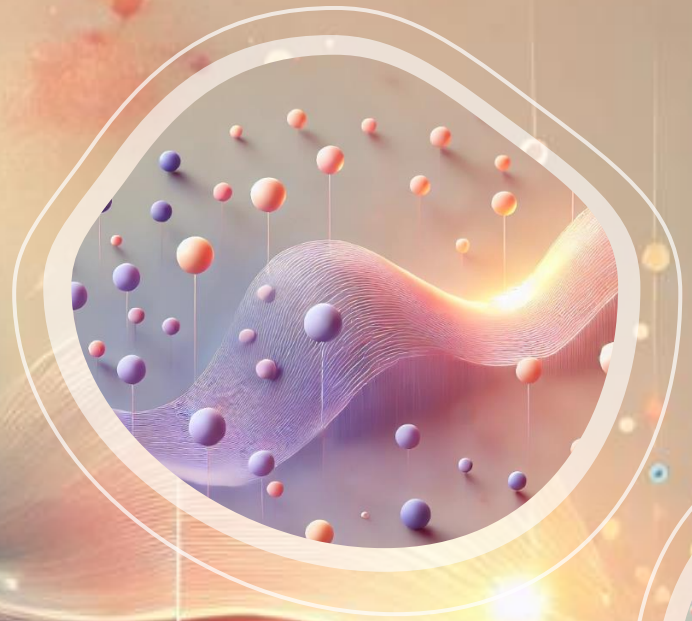
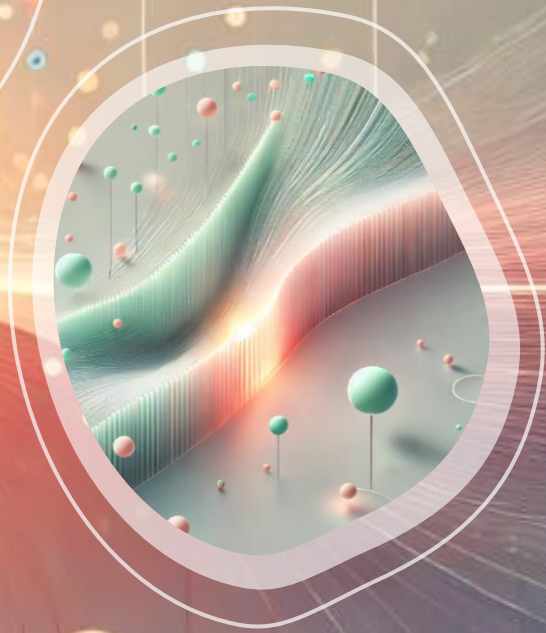
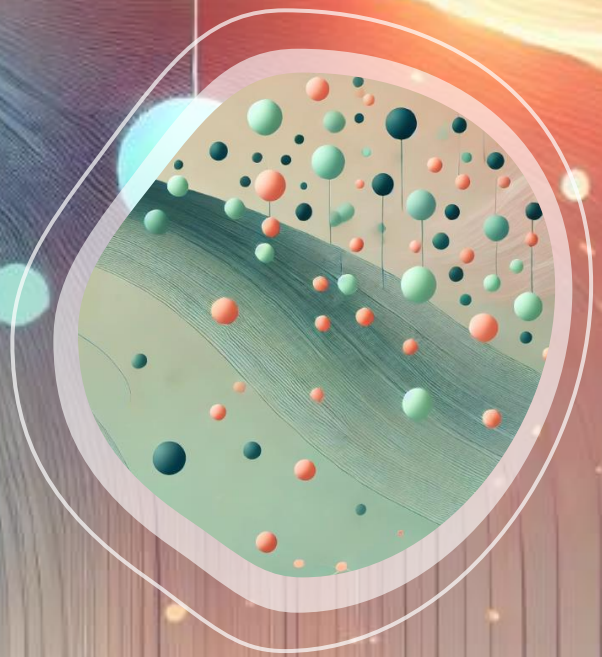
Support Vector Machines

- ISE – 364 / 464
- Dept. of Industrial and Systems Engineering
- Griffin Dean Kent

LEHIGH UNIVERSITY







Overview of Support Vector Machines

SVMs

Support Vector Machines (SVM) are one of the most studied supervised discriminative learning models in the field of machine learning and statistics.

At a high-level view, SVMs aim at learning the parameters of a **separating hyperplane** that distinguish between different classes of data. In its most basic form, this would amount to learning a **linear boundary** between target classes; however, SVMs are also able to learn **nonlinear decision boundaries** between target classes by applying a technique that is known as the "**kernel trick**", which we will discuss later.

- SVMs are typically applied in solving classification problems and this is the context under which we will mostly be discussing them; although, it bears mentioning that there are ways to use SVM in a regression setting.

- SVMs are also one of the most mathematically heavy ML techniques and which also requires a fair amount of knowledge in constrained optimization. Please review the material on **hyperplanes** in the Linear Algebra Review Slides if need be.

Initial Setup

- Consider a dataset of m feature-label pairs (x, y) of datapoints given by $\mathcal{D} \coloneqq \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{m}$, where $x^{(i)} \in \mathbb{R}^n$ and $y^{(i)} \in \{-1,1\}$, for all $i \in \{1,2, \dots, m\}$.

- The **Goal of an SVM model** is to learn the parameters of a hyperplane that separates the two classes.

- Further, we will begin our discussion under a **strong assumption**: that our data are **linearly separable** (this means that the two target classes in our dataset can be separated by a linear classifier without any datapoint being misclassified).
  - As one can imagine, this is a relatively unreasonable assumption to make about any problem in reality, but we will use this formulation as a starting point from which more sophisticated formulations can be derived.

- The formulation of SVM that comes from this assumption is referred to as **Hard-SVM** (short for "hard-margin" SVM) and can be contrasted with the more sophisticated **Soft-SVM** (short for "soft-margin" SVM) which does indeed allow overlap in target classes.

Illustration of Separating Hyperplanes

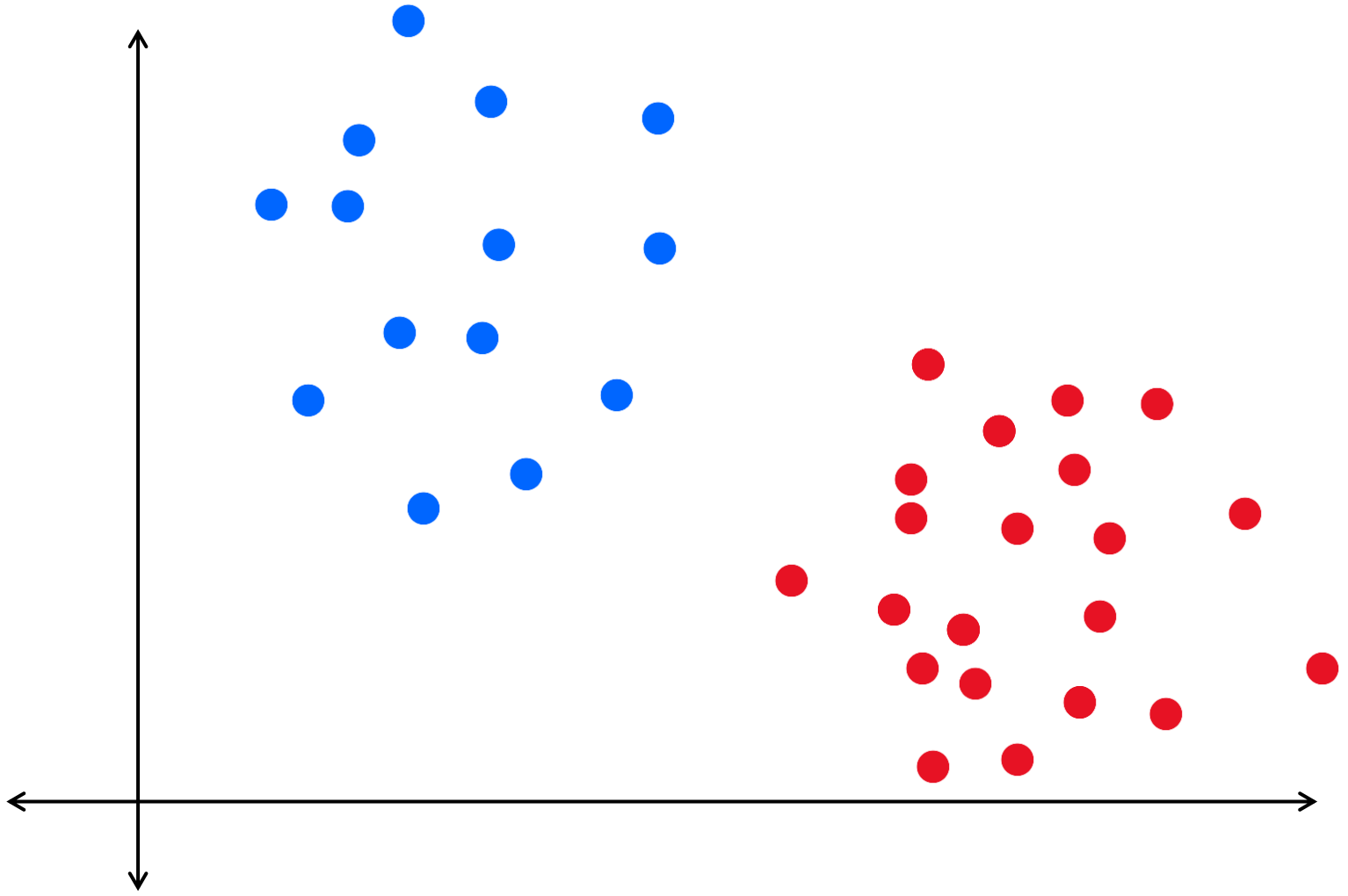
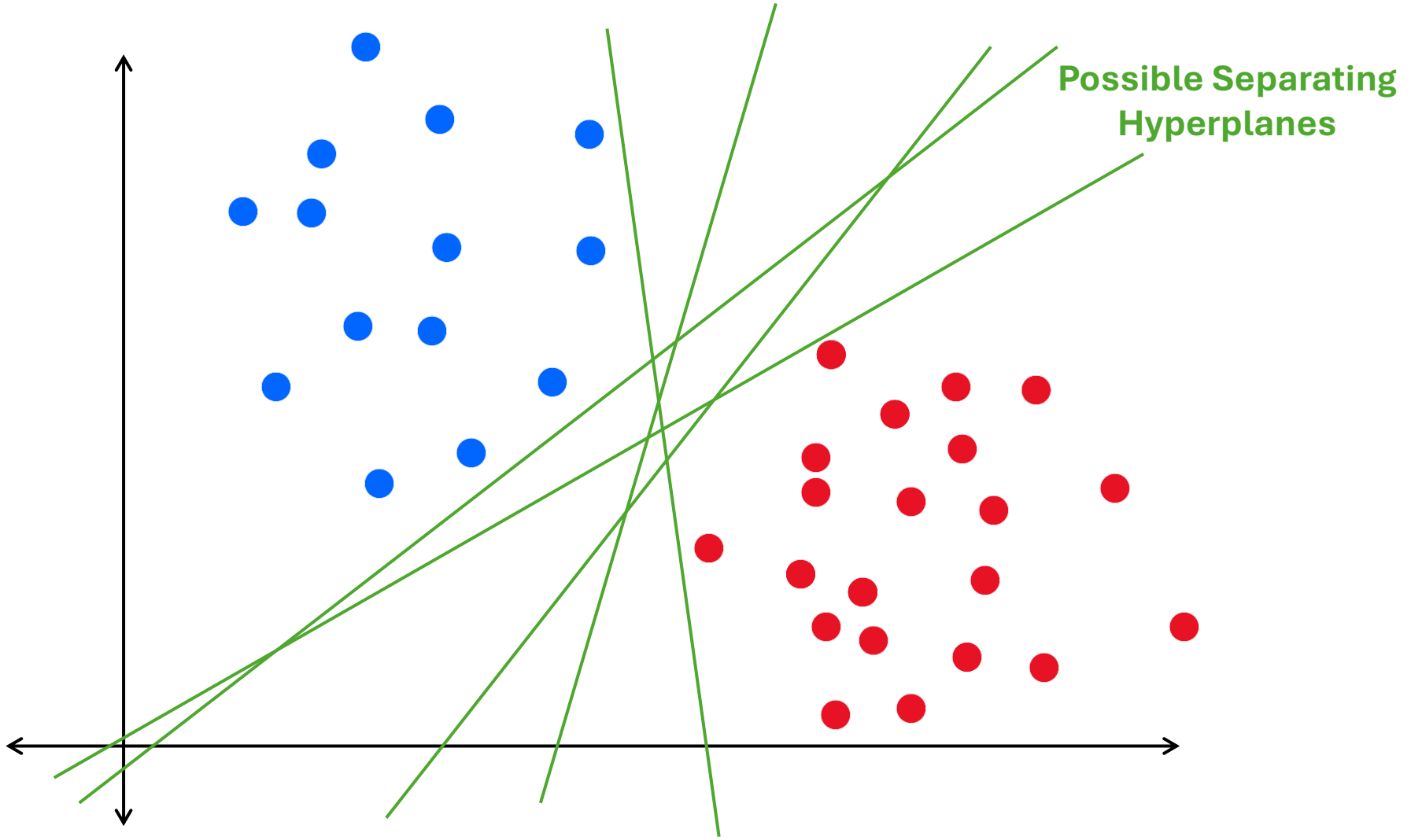


Illustration of Separating Hyperplanes



Hyperplanes Revisited

Recall that a hyperplane $\mathcal{H} \subseteq \mathbb{R}^n$ is a set of points that satisfy a given linear equation, and is defined explicitly as

$$\mathcal{H} \coloneqq \{x \in \mathbb{R}^n : w^T x - b = 0\},$$

where $w \in \mathbb{R}^n$ is the normal vector and $b \in \mathbb{R}$ is the intercept.

Further, we can define a predictive model h_θ to represent this hyperplane by denoting the function

$$h_\theta(x) \coloneqq \theta^T x,$$

where $\theta \coloneqq [-b, w^T] \in \mathbb{R}^{n+1}$ is the vector of parameters that define the hyperplane (the normal and intercept) and x has been augmented to have a 1 appended to the front of the vector, i.e., $x \coloneqq [1, x_1, x_2, \dots, x_n] \in \mathbb{R}^{n+1}$ (in this way, we can express the hyperplane with the more-condensed dot-product form).

Therefore, we can state the goal of our SVM model mathematically as learning the parameters θ of the hyperplane $h_\theta(x) \coloneqq \theta^T x = w^T x - b = 0$ subject to the **constraint** that all the datapoints are correctly classified (this comes from our **assumption that the datapoints are linearly separable**), which we can express as the constraint

$$y^{(i)}\left(w^T x^{(i)} - b\right) > 0.$$

Notice that this constraint helps highlight why we chose the binary target $y^{(i)}$ to have values of $\{-1,1\}$ (as opposed to $\{0,1\}$). This constraint implies that a datapoint $\left(x^{(i)}, y^{(i)}\right)$, where $y^{(i)} = 1$, will lie on the side of the hyperplane that satisfies $y^{(i)}\left(w^T x^{(i)} - b\right) > 0$, whereas a datapoint $\left(x^{(i)}, y^{(i)}\right)$, where $y^{(i)} = -1$, will lie on the side of the hyperplane that satisfies $y^{(i)}\left(w^T x^{(i)} - b\right) < 0$.

“Best-Fit” Hyperplanes

It should be obvious that there could very well be **multiple choices of the parameters** $\theta = [-b, w^T]$, each defining different hyperplanes, which could correctly classify all the datapoints. Thus, the question becomes:

Which hyperplane should we choose?

Answer: We want to choose the hyperplane that will yield the **largest margin**.

Margins & Support Vectors

Given some **decision boundary** $\mathcal{B}$ (defined by some mathematical model), the “**margin**” is defined as the distance between the closest points of the target classes and the decision boundary $\mathcal{B}$. It bears mentioning that the points that are closest to the decision boundary are called the “**support vectors**”.

Illustration of Support Vectors

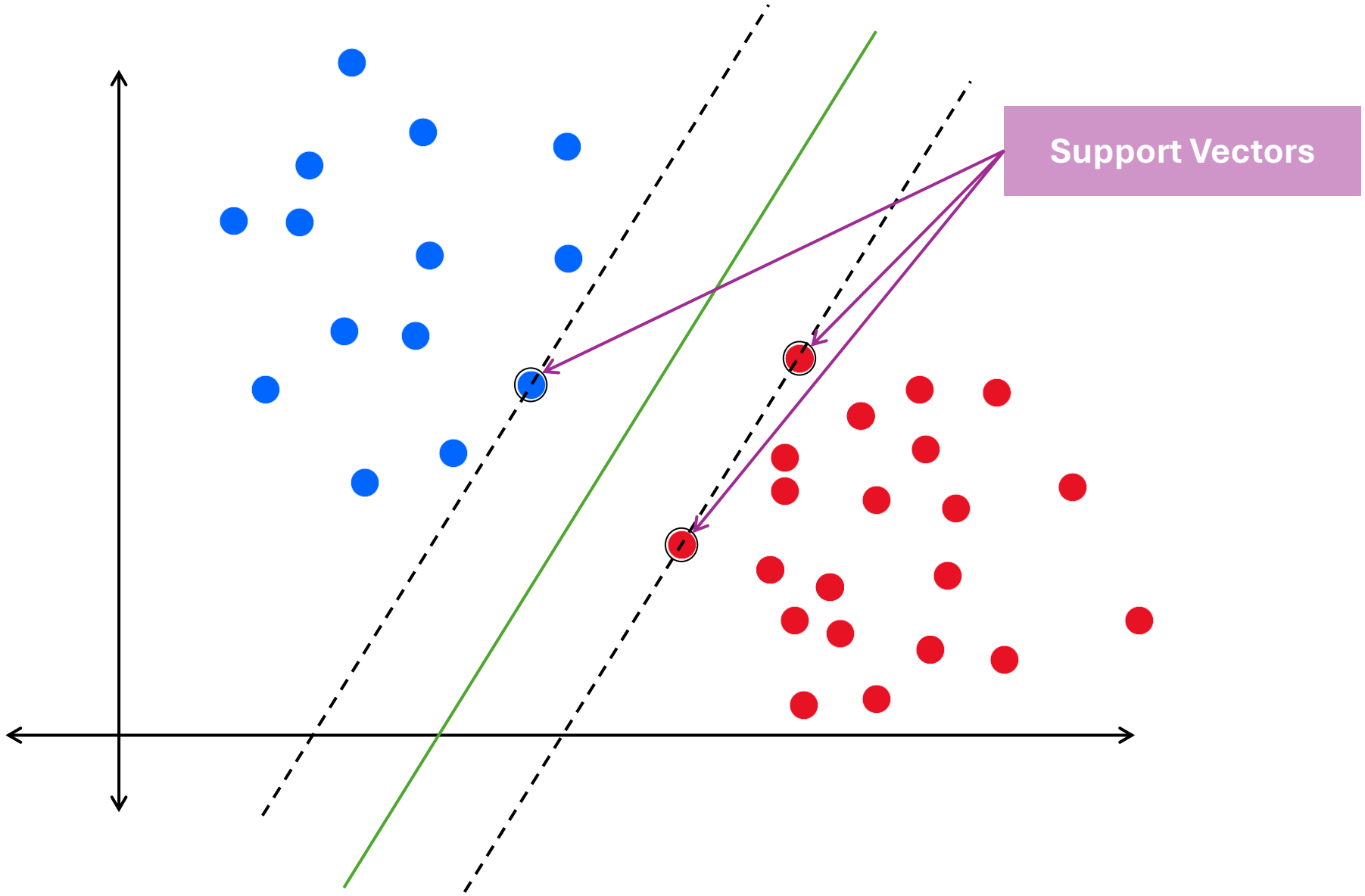
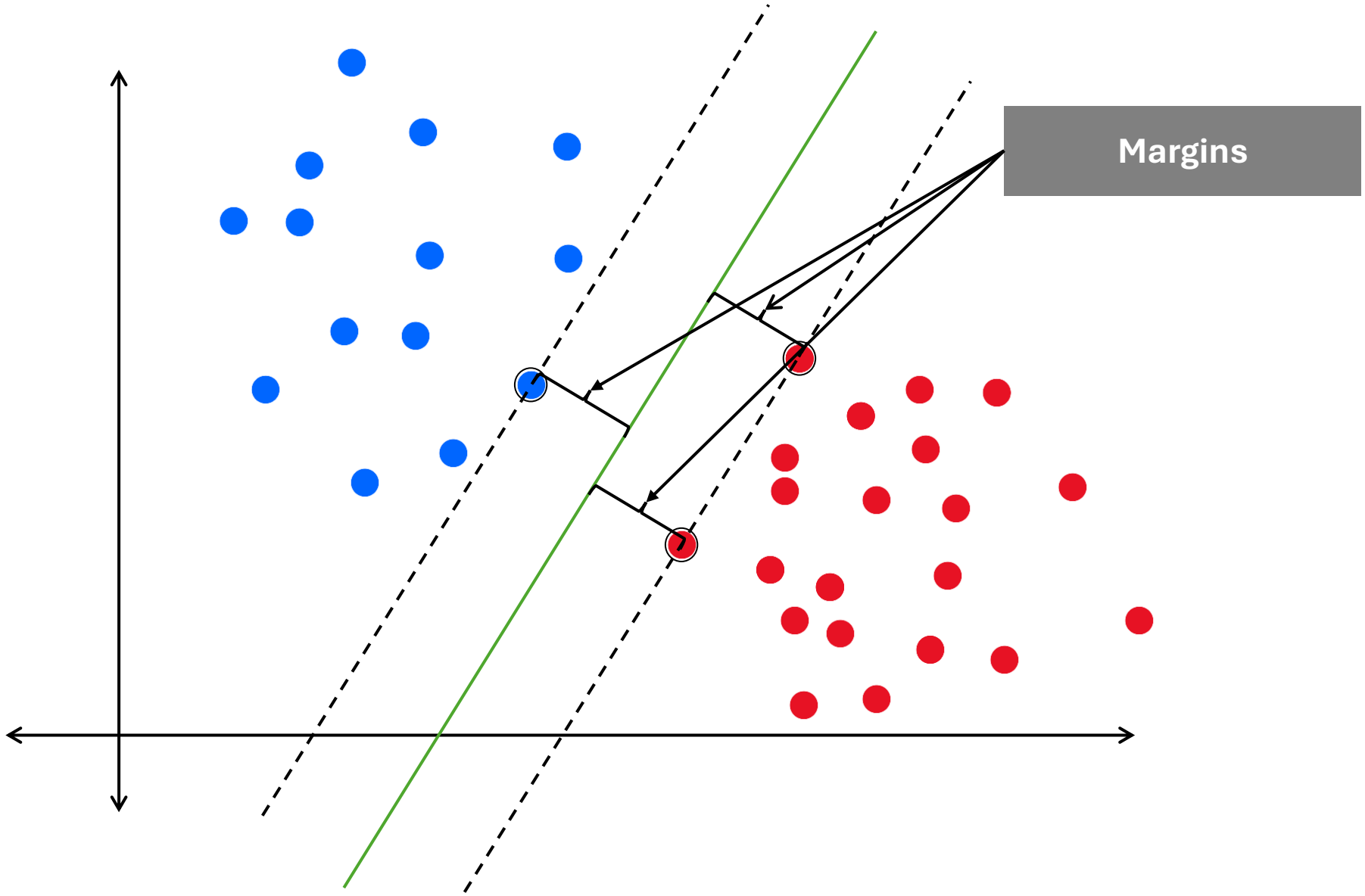


Illustration of Support Vectors



“Best-Fit” Hyperplanes

It should be obvious that there could very well be **multiple choices of the parameters** $\theta = [-b, w^T]$, each defining different hyperplanes, which could correctly classify all the datapoints. Thus, the question becomes:

Which hyperplane should we choose?

Answer: We want to choose the hyperplane that will yield the **largest margin**.

Margins & Support Vectors

Given some **decision boundary** $\mathcal{B}$ (defined by some mathematical model), the “**margin**” is defined as the distance between the closest points of the target classes and the decision boundary $\mathcal{B}$. It bears mentioning that the points that are closest to the decision boundary are called the “**support vectors**”.

We can begin by defining the “**functional margin**” $\hat{\gamma}^{(i)} \in \mathbb{R}_+$, for every datapoint $\left(x^{(i)}, y^{(i)}\right)$, as the constraint that we introduced on the previous slide, i.e.,

$$\hat{\gamma}^{(i)} := y^{(i)}\left(w^T x^{(i)} - b\right).$$

- Notice that every function margin is guaranteed to be a positive value by our assumption of linear separability.
- Further, $\hat{\gamma}^{(i)}$ is a measure of how confident the hyperplane $h_\theta(x) = \theta^T x = 0$ is about correctly classifying the datapoint $\left(x^{(i)}, y^{(i)}\right)$.
- Notice that if $\left(x^{(i)}, y^{(i)}\right)$ is very close to the hyperplane, then $\hat{\gamma}^{(i)}$ will be small and near 0 (with a value of 0 if the datapoint is on the hyperplane). Conversely, $\hat{\gamma}^{(i)}$ will be large if the datapoint is far from the hyperplane.
- The larger the “distance” (as measured by the functional margin) a datapoint is from the hyperplane the more confidant we will be that the point is classified correctly.

Functional-Margin Formulation of SVM

Thus, since the **numerical value of the functional margins** are directly tied to the level of **confidence in the classification**, we intuitively wish to **choose the parameters of the hyperplane that maximize the smallest of the functional margins**.

- Notice that we do not want to maximize over all the functional margins; just the margins corresponding to the datapoints that are closest to the hyperplane.

Functional-Margin Formulation of SVM (FM-SVM)

Let's denote the smallest of the functional margins as

$$\hat{\gamma} := \min_{1 \le i \le m} \hat{\gamma}^{(i)}.$$

Using this, we can define our first formulation of SVM as an optimization problem where we wish to make the smallest functional margin $\hat{\gamma}$ as large as possible, which will in-turn ensure that the other functional margins are relatively large. This formulation can be stated as:

$$\max_{w,b,\hat{\gamma}} \quad \hat{\gamma}$$
$$\text{s.t.} \quad y^{(i)}\left(w^T x^{(i)} - b\right) \ge \hat{\gamma}, \qquad \text{for all } i \in \{1,2,\dots,m\}.$$

- However, there is a problem with this functional margin formulation of SVM. What is it?

Projected Distance as a Superior Metric Over the Functional Margin

The **issue** with the functional margin formulation of SVM is that of **hyperplane identification** and the **sensitivity of the functional margin to scaling**. Specifically, this is a problem since one can scale the parameters w and b by some factor $t > 1$, which will not actually change the orientation of the plane (since tw will still have the same orientation of w) nor will it change the distance from the plane to the origin (since $\frac{|tb|}{\|tw\|} = \frac{t|b|}{t\|w\|} = \frac{|b|}{\|w\|}$), but it will indeed alter the value of the functional margin.

- This means that the ***functional margin is not scale-invariant***.
- This scale dependency can lead to **ambiguity** in defining the largest functional margin since increasing the norm of w will artificially inflate the functional margin without changing the actual distance between the datapoint $\left(x^{(i)}, y^{(i)}\right)$ and the hyperplane $h_\theta\left(x^{(i)}\right)$.

A natural way to handle solving this problem of hyperplane identification is to utilize an equivalent reformulation. This is typically done by imposing a normalization constraint on the normal vector (i.e., $\|w\| = 1$) and then normalizing the functional margins as $\gamma^{(i)} := \frac{\hat{\gamma}^{(i)}}{\|w\|} = \frac{y^{(i)}\left(w^T x^{(i)} - b\right)}{\|w\|}$, which is referred to as the "**geometric margin**" of the datapoint $\left(x^{(i)}, y^{(i)}\right)$. This will eliminate the problem of hyperplane identification.

- Under the normalization constraint $\|w\| = 1$, the functional margin and the geometric margin are equivalent, i.e., $\gamma^{(i)} = \hat{\gamma}^{(i)}$.
- Also notice that the **geometric margin** is **equivalent** to the **projected distance of the vector $x^{(i)}$ to the hyperplane** h_θ (when the data are linearly separable), i.e., $\gamma^{(i)} = \frac{y^{(i)}\left(w^T x^{(i)} - b\right)}{\|w\|} = \frac{|w^T x^{(i)} - b|}{\|w\|}$.

Geometric-Margin Formulation of SVM

We can now use the geometric margin to reformulate the FM-SVM optimization problem.

Geometric-Margin Formulation of SVM (GM-SVM)

To eliminate the hyperplane ambiguity introduced in the FM-SVM formulation, one can impose the added normalization constraint $\|w\| = 1$, which will allow us to reformulate the FM-SVM formulation in an equivalent form by utilizing the geometric margins $\gamma^{(i)} \coloneqq \frac{\hat{\gamma}^{(i)}}{\|w\|} = \frac{y^{(i)}(w^T x^{(i)} - b)}{\|w\|}$. Denoting the smallest of the geometric margins as

$$\gamma \coloneqq \frac{\hat{\gamma}}{\|w\|} = \min_{1 \le i \le m} \gamma^{(i)},$$

we can state the equivalent **geometric-margin formulation of SVM** (**GM-SVM**) reformulation that eliminates the hyperplane ambiguity as:

$$\begin{aligned} \max_{w,b,\gamma} \quad & \gamma \\ \text{s.t.} \quad & y^{(i)}\left(w^T x^{(i)} - b\right) \ge \gamma, \qquad \text{for all } i \in \{1,2,\dots,m\}, \\ & \|w\| = 1. \end{aligned}$$

However, there is also an issue with this formulation…

- The added normalization constraint $\|w\| = 1$ artificially restricts the solution space and will limit the flexibility of our optimization.

Reformulating GM-SVM to Eliminate the Normalization Constraint

Derivation:

Let $\theta^* := [-b^*, w^{*T}]$ denote the solution to the GM-SVM optimization problem defined on the previous slide. Then, we can define the **largest minimal geometric margin** obtained as

$$\gamma^* := \underset{w,b,\gamma}{\operatorname{argmax}}\, \gamma = \min_{1 \le i \le m} \frac{y^{(i)}\left(w^{*T} x^{(i)} - b^*\right)}{\|w^*\|}.$$

From the constraints in GM-SVM, for all $i \in \{1,2, \dots, m\}$, we have that

$$y^{(i)}\left(w^{*T} x^{(i)} - b^*\right) \ge \gamma^*.$$

Dividing both sides by γ^* and denoting $\hat{w} := \frac{w^*}{\gamma^*}$ and $\hat{b} := \frac{b^*}{\gamma^*}$, we obtain

$$y^{(i)}\left(\left(\frac{w^*}{\gamma^*}\right)^T x^{(i)} - \frac{b^*}{\gamma^*}\right) \ge 1 \qquad \rightarrow \qquad y^{(i)}\left(\hat{w}^T x^{(i)} - \hat{b}\right) \ge 1.$$

Now, from the normalization constraint, we know that $\|w^*\| = 1$, which implies that $\|\hat{w}\| = \left\|\frac{w^*}{\gamma^*}\right\| = \frac{\|w^*\|}{|\gamma^*|} = \frac{1}{\gamma^*}$.

Therefore, we have that $\gamma^* = \frac{1}{\|\hat{w}\|}$. Notice that this implies that maximizing the geometric margin γ is equivalent to maximizing the quantity $\frac{1}{\|\bar{w}\|}$ over the parameters $\bar{w}$, such that $\frac{1}{\|\hat{w}\|} = \underset{\bar{w}}{\operatorname{argmax}} \frac{1}{\|\bar{w}\|}$.

We can formally state this newly derived formulation as an optimization problem over the parameters of the "new" hyperplane defined by the equation $\bar{w}^T x^{(i)} - \bar{b}$, where $\hat{w}$ and $\hat{b}$ denote the new optimal parameters.

Hard-Margin Formulation of SVM

We can now officially state what is referred to as the **Hard-Margin formulation of SVM** (HM-SVM) using the derivation on the previous slide.

- It bears mentioning that in this new formulation, we do indeed use an **abuse of notation** by simply redefining the parameters of the "new" hyperplane $\widehat{w}$ and $\hat{b}$ as the same variables that we used for the parameters of the "old" hyperplane w and b (simply for ease of notation), i.e., we have $w \coloneqq \widehat{w}$ and $b \coloneqq \hat{b}$.

Hard-Margin Formulation of SVM (HM-SVM)

The HM-SVM formulation, which successfully eliminates the normalization constraint of the GM-SVM formulation, is stated as

$$\begin{aligned} \max_{w,b} \quad & \frac{1}{\|w\|} \\ \text{s.t.} \quad & y^{(i)}\left(w^T x^{(i)} - b\right) \geq 1, \qquad \text{for all } i \in \{1,2,\dots,m\}. \end{aligned}$$

Although this is a correct form of the HM-SVM problem, the actual used-in-practice form (which is more efficient and cleaner to solve) is given by the equivalent problem

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y^{(i)}\left(w^T x^{(i)} - b\right) \geq 1, \qquad \text{for all } i \in \{1,2,\dots,m\}. \end{aligned}$$

This optimization problem is also referred to as the "**Primal SVM problem**" for reasons that will be discussed.

The second form of HM-SVM is a **convex optimization problem**.

Illustration of HM-SVM

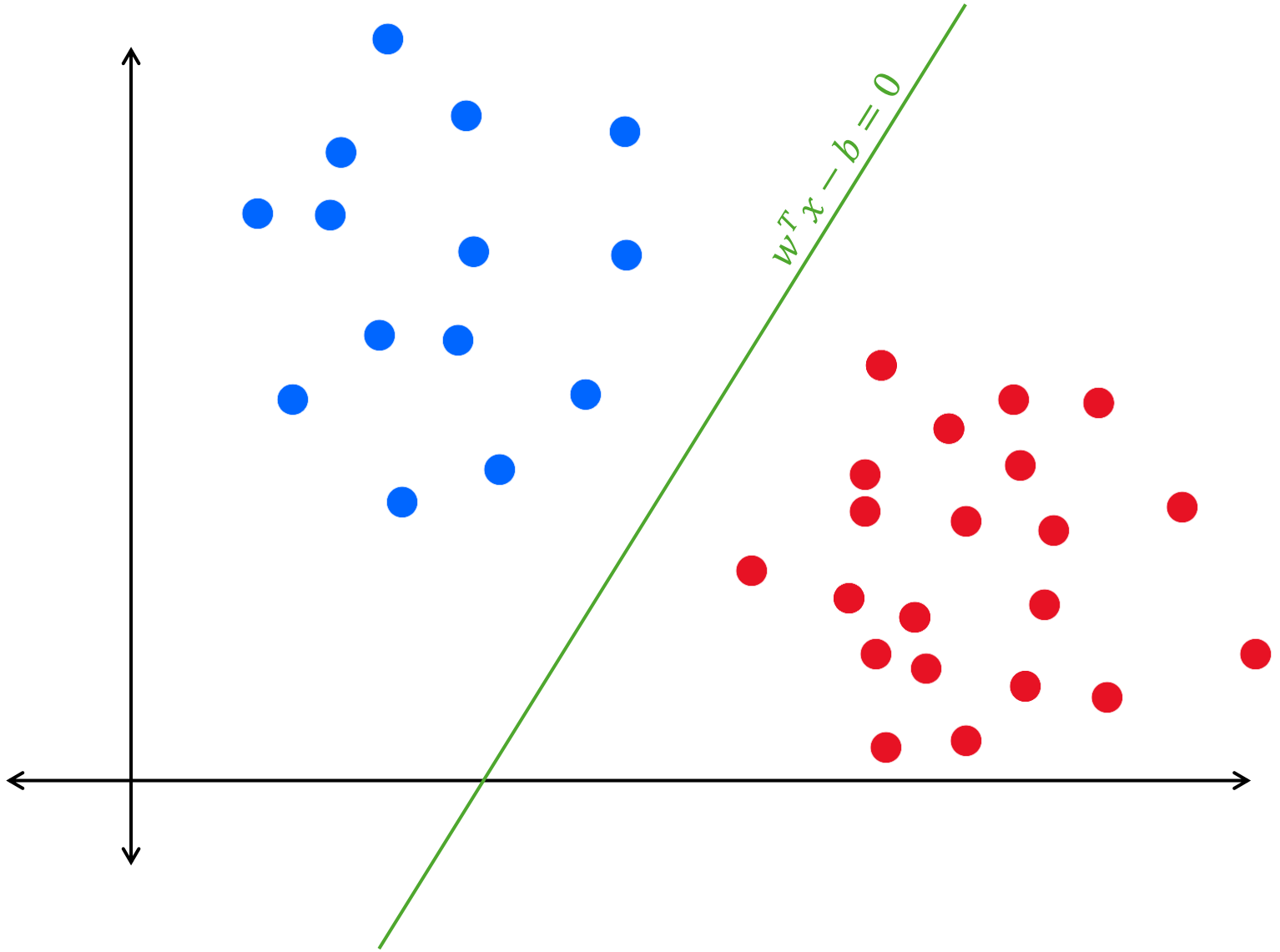


Illustration of HM-SVM

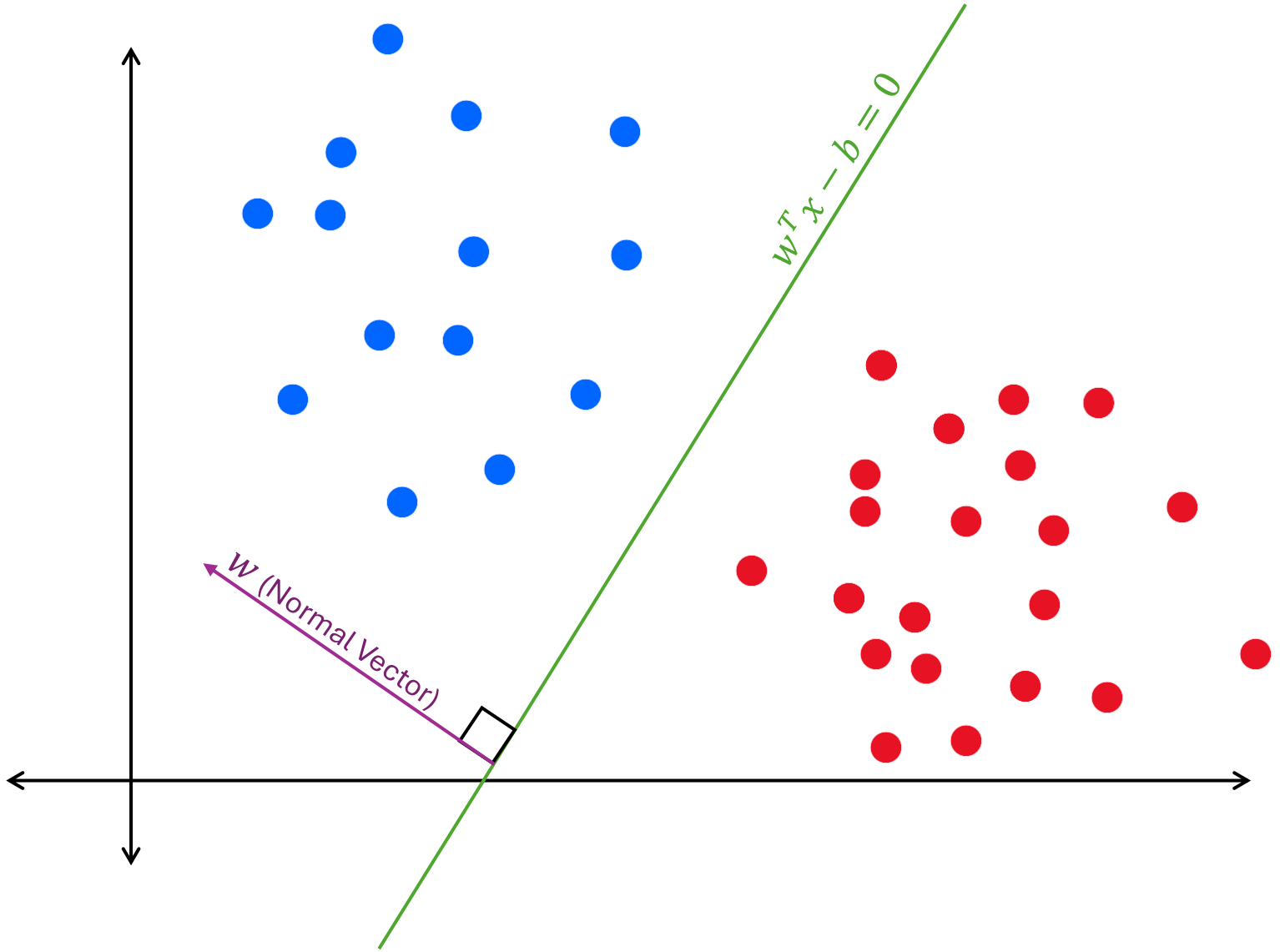


Illustration of HM-SVM

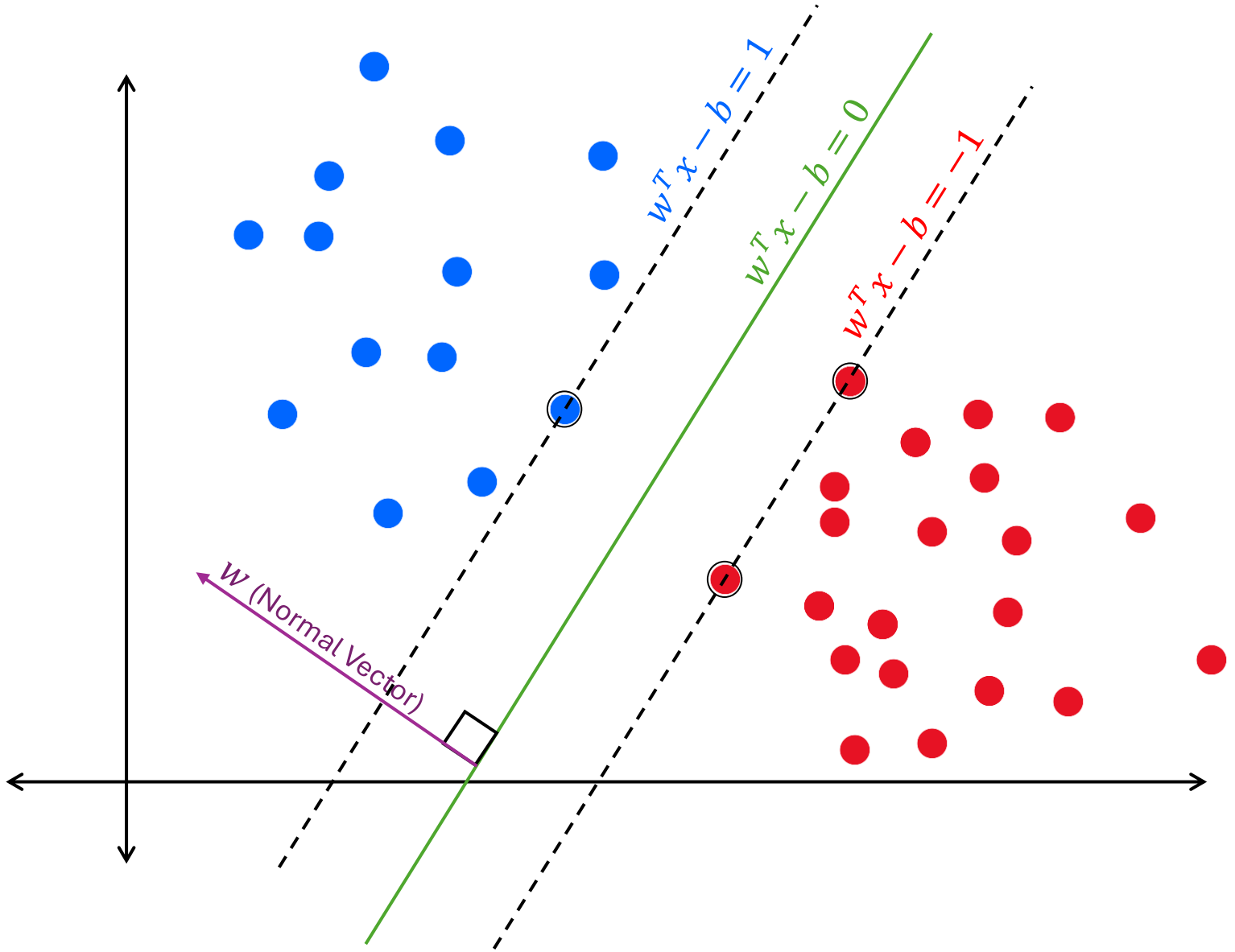


Illustration of HM-SVM

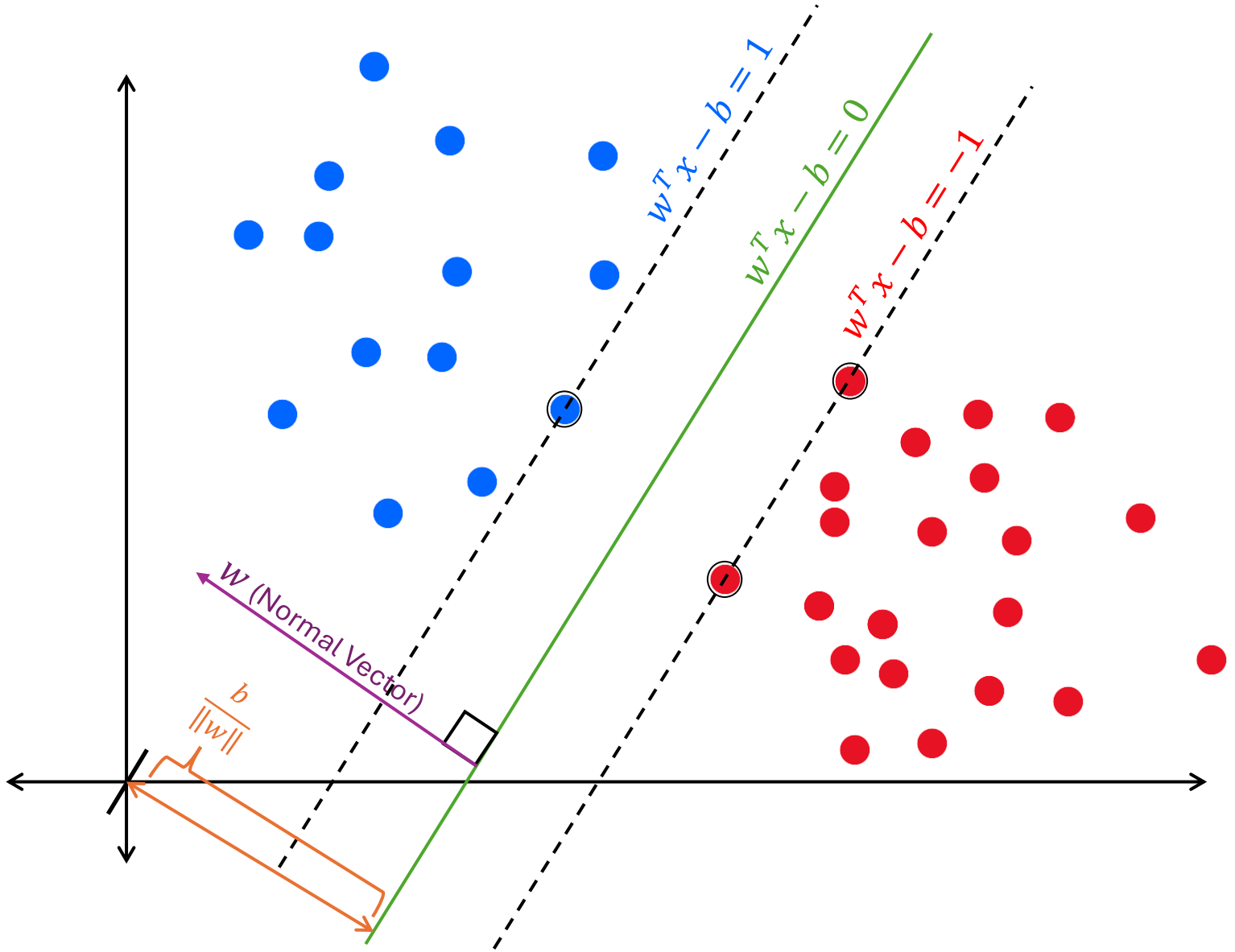


Illustration of HM-SVM

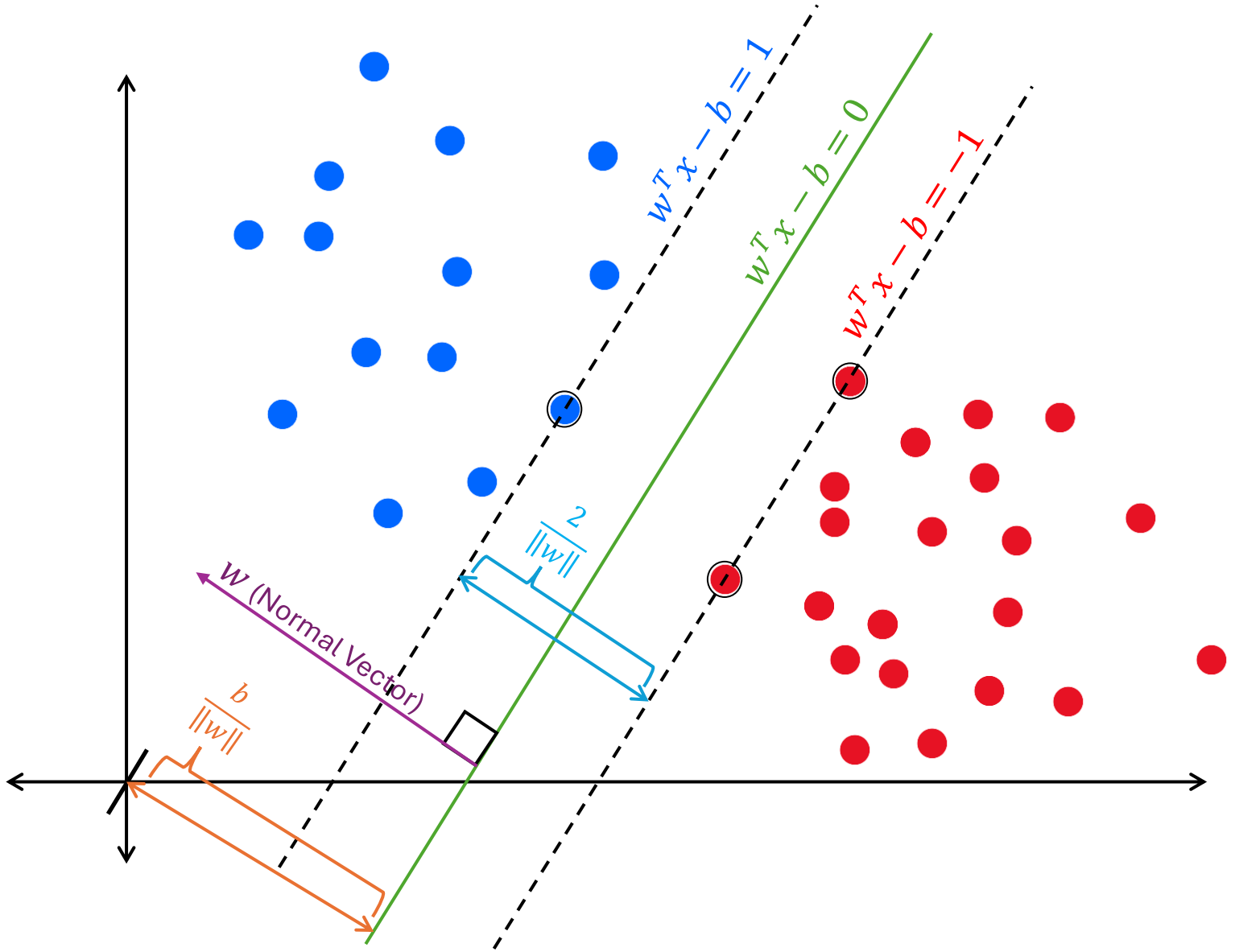
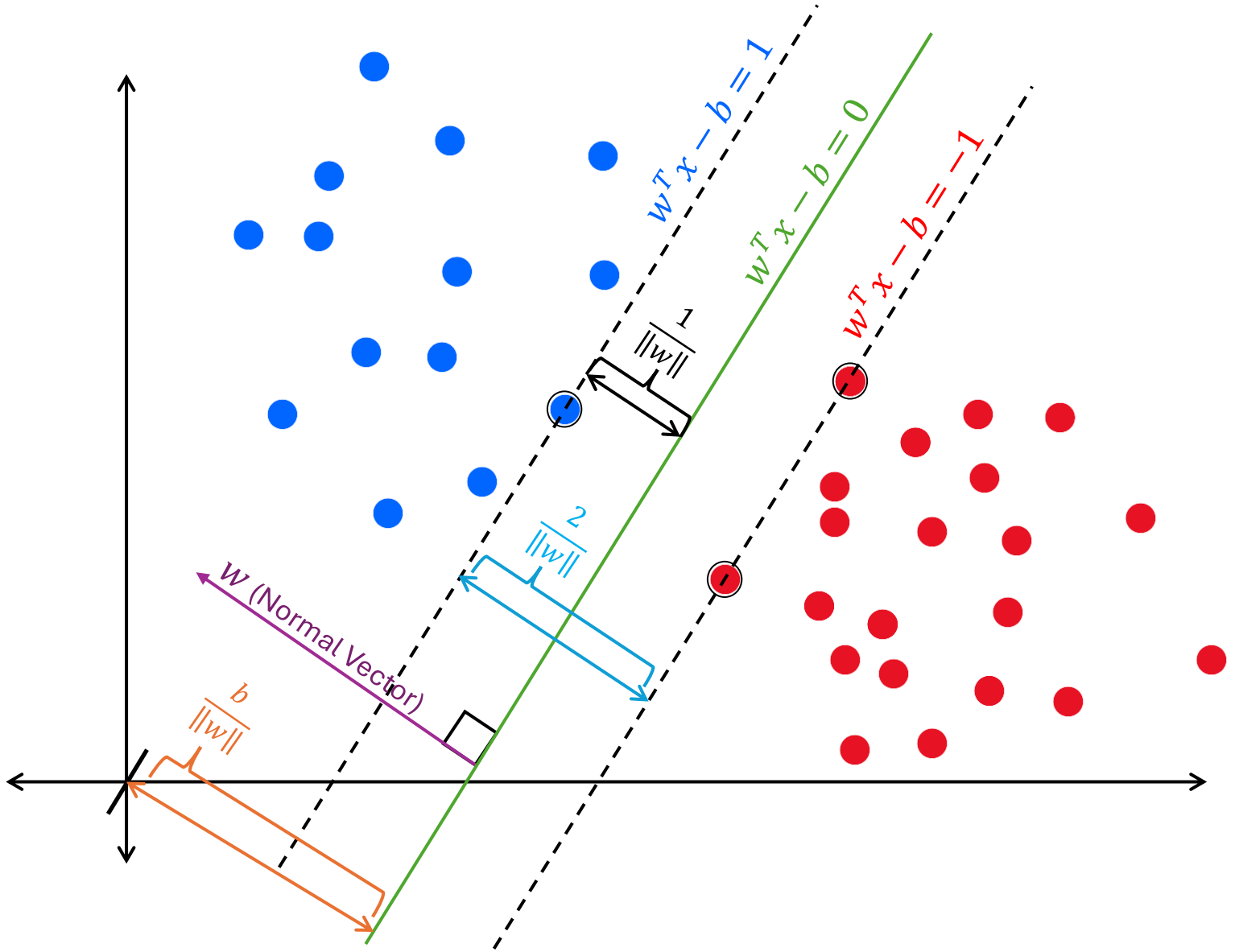


Illustration of HM-SVM



Addressing The Problem of Non-Separable Data

- Until this point, we have derived formulation for SVM under the strong assumption that the target classes of our dataset are linearly separable (ultimately leading to the Hard-SVM formulation).
- However, this is a very strong assumption and one which is not usually satisfied with real data.
- As such, we wish to derive an SVM formulation that will **tolerate small portions of misclassifications** to occur to still determine a general decision boundary between target classes of data. This is referred to as **Soft-SVM**.

- To do this, we can introduce a new vector of **"slack" variables** $\xi \in \mathbb{R}^m$, such that $\xi_i \geq 0 \; \forall i \in \{1,2,\dots,m\}$. The variables ξ_i define the **proportional amount** by which the **prediction** of the datapoint $\left(x^{(i)}, y^{(i)}\right)$ is on **the wrong side of its margin**. The variables ξ_i can be thought of as the **residuals** of datapoints to their margins.
- Thus, if $\xi_i > 0$ then the datapoint $\left(x^{(i)}, y^{(i)}\right)$ is on the **wrong side of its margin**, if $\xi_i > 1$ then the datapoint $\left(x^{(i)}, y^{(i)}\right)$ is on the **wrong side of the decision boundary** (as such, this point would be misclassified by the model), and if $\xi_i = 0$ then the datapoint $\left(x^{(i)}, y^{(i)}\right)$ is **correctly classified** and is on the **correct side of its margin**.

- Therefore, we can augment the Hard-SVM formulation by incorporating these slack variables along with the constraints (for all $i \in \{1,2,\dots,m\}$) $\xi_i \geq 0$, $y^{(i)}\left(w^T x^{(i)} - b\right) \geq 1 - \xi_i$, and $\sum_{i=1}^{m} \xi_i \leq K$ for some $K \in \mathbb{R}_+$.
- The constraints $\xi_i \geq 0$ ensure the slack variables are **nonnegative**. The constraints $y^{(i)}\left(w^T x^{(i)} - b\right) \geq 1 - \xi_i$ allow for the datapoints to **lie on the wrong side of the decision boundary** by the amounts ξ_i. Lastly, the constraint $\sum_{i=1}^{m} \xi_i \leq K$ bounds the amount of error that is incurred by datapoints falling on the wrong side of their margins; this will help **ensure** that the **best margins are correctly identified**.

Soft-Margin Formulation of SVM

Including the slack variables as well as the corresponding constraints that were discussed on the previous slide, we can now define the Soft-Margin formulation of SVM (Soft-SVM) as the problem

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y^{(i)}\left(w^T x^{(i)} - b\right) \geq 1 - \xi_i, && \text{for all } i \in \{1,2,\dots,m\}, \\ & \xi_i \geq 0, && \text{for all } i \in \{1,2,\dots,m\}. \\ & \sum_{i=1}^{m} \xi_i \leq K\,. \end{aligned}$$

Soft-Margin Formulation of SVM (Soft-SVM)

However, this can be reformulated as the more computationally efficient-to-solve formulation of Soft-SVM by introducing the Lagrange multiplier $C \geq 0$, yielding the problem

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i \\ \text{s.t.} \quad & y^{(i)}\left(w^T x^{(i)} - b\right) \geq 1 - \xi_i, && \text{for all } i \in \{1,2,\dots,m\}, \\ & \xi_i \geq 0, && \text{for all } i \in \{1,2,\dots,m\}. \end{aligned}$$

Notice that in the case where the data are indeed linearly separable, the Hard-SVM formulation can be reobtained by setting $C = \infty$, which will drive $\xi = 0$. Further, this formulation is also a **convex optimization problem**.

Illustration of SVM for Non-Separable Data

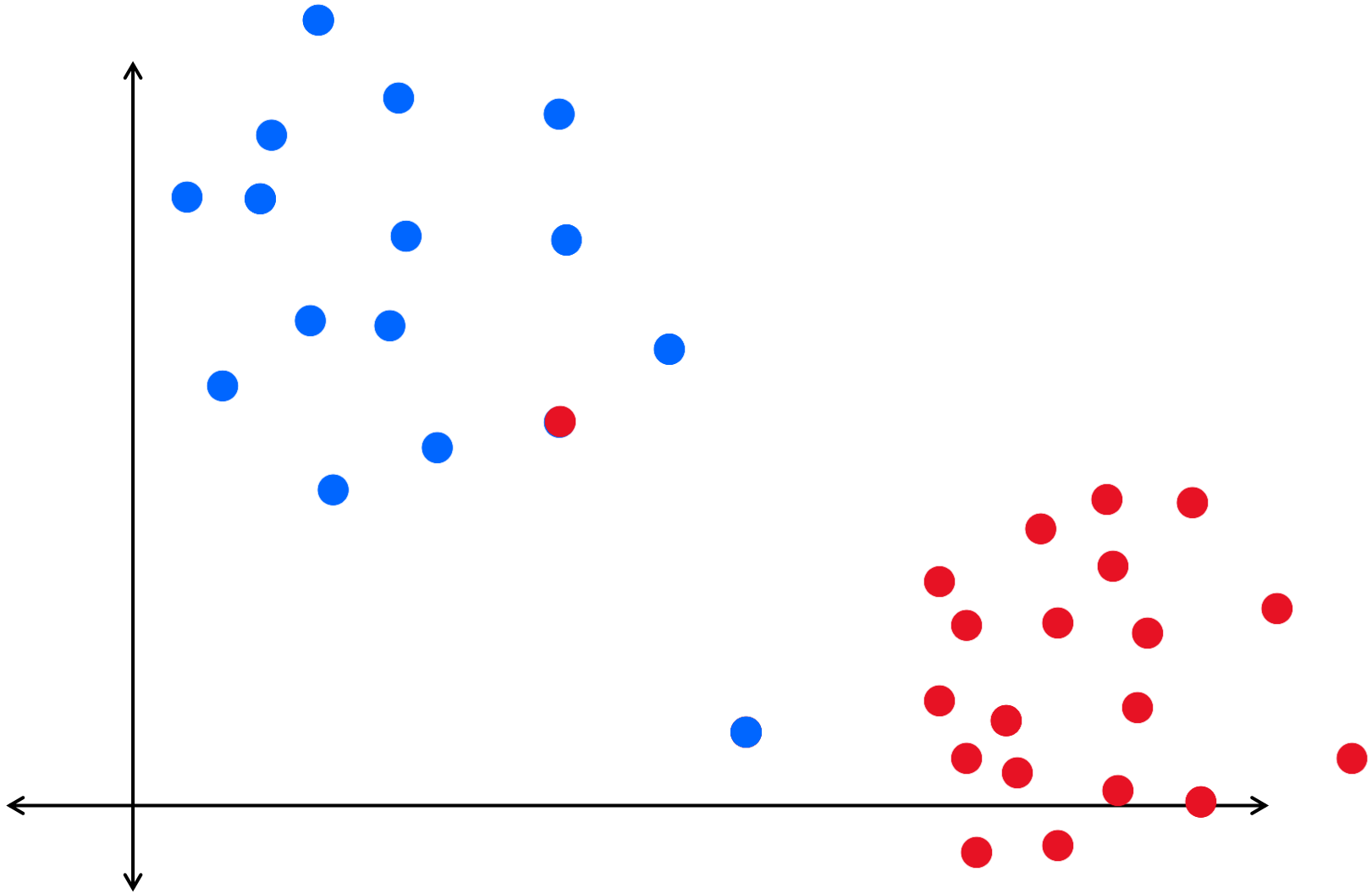


Illustration of SVM for Non-Separable Data

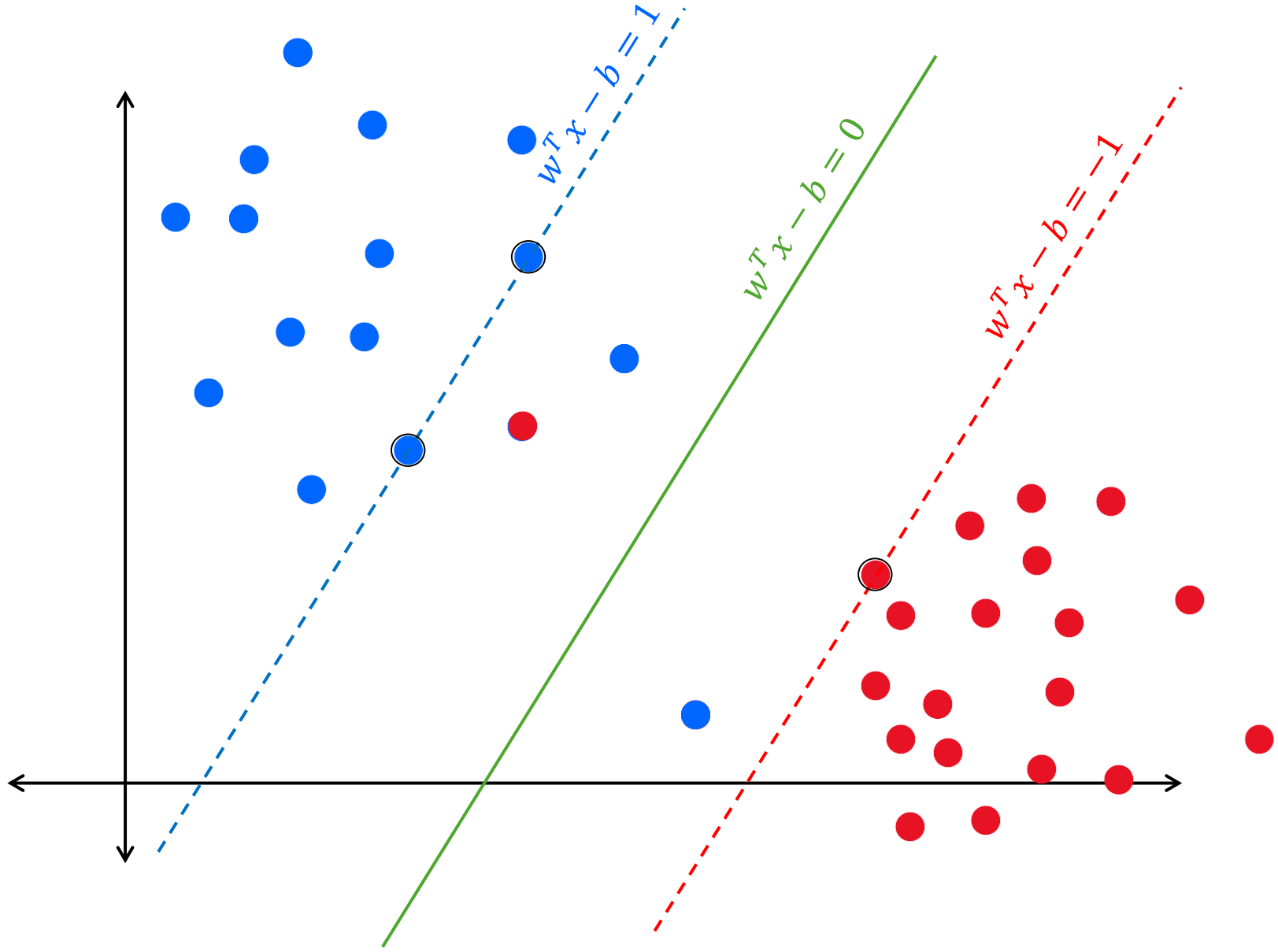
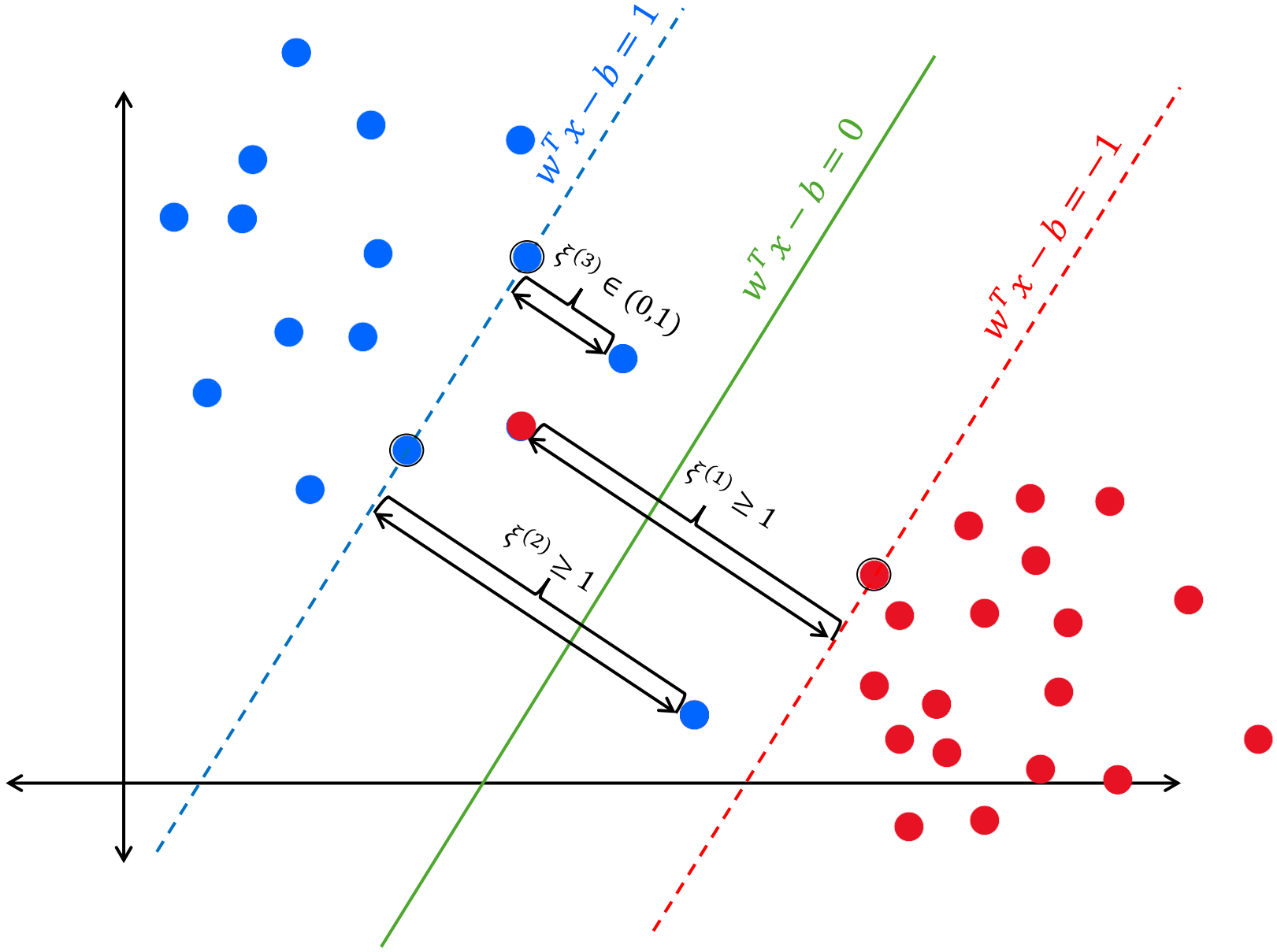
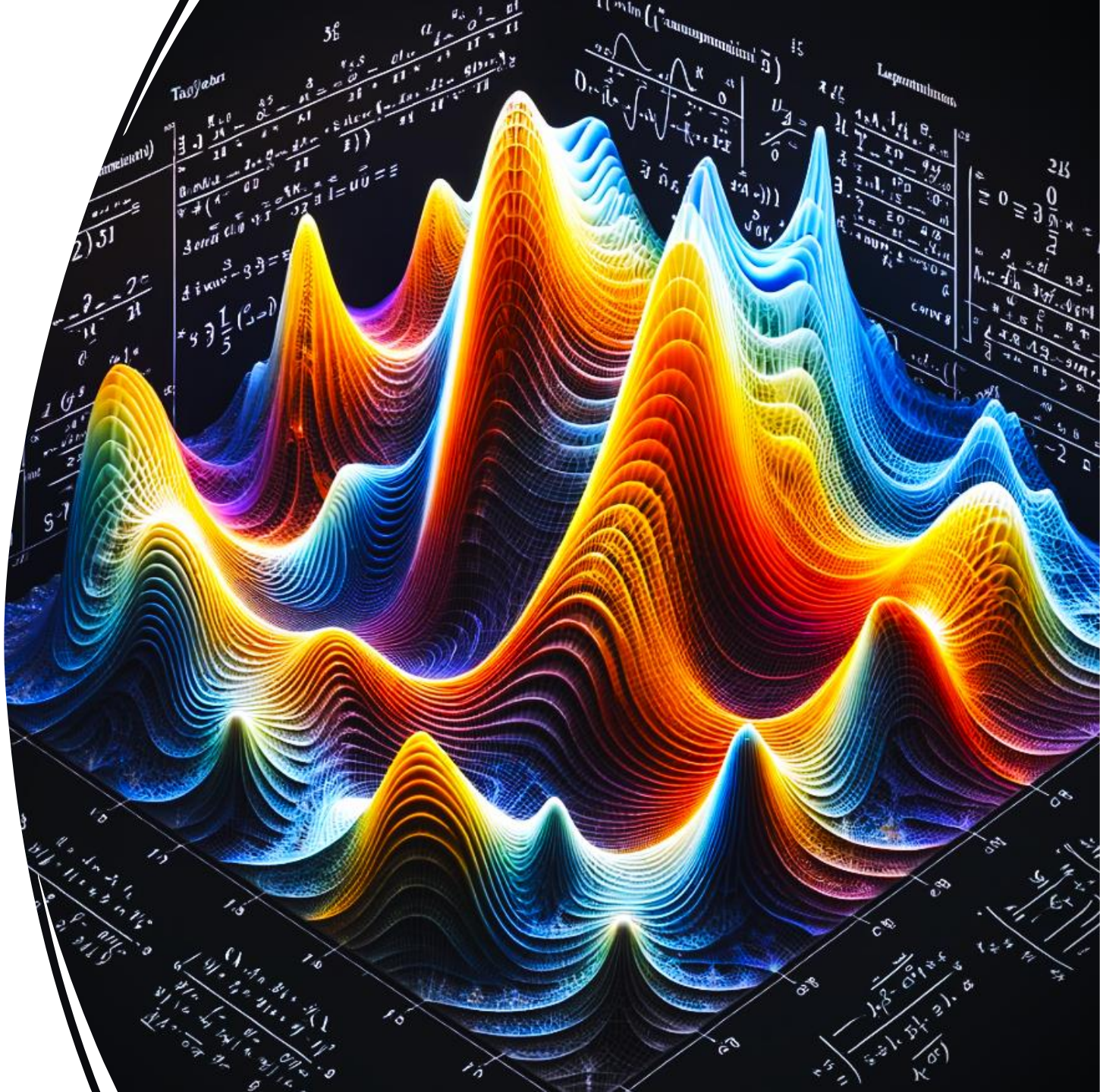


Illustration of SVM for Non-Separable Data



Overview of Constrained Optimization: The Primal & Dual



Overview of Primal Optimization

Consider the **Primal optimization problem** given by

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \geq 0, \qquad \text{for all } i \in \{1,2,\dots,m\}. \end{aligned}$$

This problem has the corresponding **Lagrangian function**, given some vector of Lagrange multipliers $\lambda \in \mathbb{R}_+^n$, defined as

$$\mathcal{L}(x,\lambda) \coloneqq f(x) - \sum_{i=1}^{m} \lambda_i g_i(x) = f(x) - \lambda^T g(x),$$

where $g(x) \coloneqq [g_1(x), g_2(x), \dots, g_m(x)]^T$. Using this, we can rewrite the primal problem as the unconstrained problem

$$\hat{\mathcal{L}}(\lambda) \coloneqq \min_{x \in \mathbb{R}^n} \mathcal{L}(x,\lambda) = f(x) - \lambda^T g(x).$$

Assuming there exists an optimal solution to this problem, it will satisfy what are known as the **KKT (Karush-Kuhn-Tucker) conditions**, which are given by the equations

$$\begin{aligned} \nabla_x \mathcal{L}(x,\lambda) = \nabla_x f(x) - \lambda^T \nabla_x g(x) = 0, \\ \lambda_i \geq 0, \qquad \text{for all } i \in \{1,2,\dots,m\}, \\ g_i(x) \geq 0, \qquad \text{for all } i \in \{1,2,\dots,m\}, \\ \lambda_i g_i(x) = 0, \qquad \text{for all } i \in \{1,2,\dots,m\}, \end{aligned}$$

The first condition states that the gradient of the Lagrangian evaluated at the point is equal to 0 (**stationarity**), the next two conditions state that the point will satisfy the constraints (**feasibility**), and the last ensures **complementary slackness**.

Overview of Dual Optimization

The Primal problem has a corresponding **Dual optimization problem** defined as

$$\begin{aligned} \max_{\lambda \in \mathbb{R}^m} \quad & \hat{\mathcal{L}}(\lambda) \\ \text{s. t.} \quad & \lambda_i \geq 0, \qquad \text{for all } i \in \{1,2,\dots,m\}. \end{aligned}$$

- Thus, the optimal solution to the dual problem is obtained by maximizing the dual objective function $\hat{\mathcal{L}}(\lambda) \coloneqq \min_{x \in \mathbb{R}^n} \mathcal{L}(x, \lambda)$ over the "dual variables" λ (which are the Lagrange multipliers) in the Lagrangian function.

Primal & Dual Properties

- **Strong Duality:** A property that guarantees that the optimal value of the primal problem $p^* \coloneqq \min_{x \in \mathbb{R}^n} f(x)$ is equivalent to the optimal value of the dual problem $d^* \coloneqq \max_{\lambda \in \mathbb{R}^m} \hat{\mathcal{L}}(\lambda)$, i.e., $p^* = d^*$. Strong Duality can be ensured under certain conditions, but specifically for primal problems that are convex, and which have strictly feasible points.
- **Weak Duality:** A property that guarantees that the optimal value of the dual problem is a lower bound for the optimal value of the primal problem, i.e., $d^* \leq p^*$.

Why do we care about the dual problem?

- Every primal optimization problem has a corresponding dual problem associated with it.
- If the primal problem satisfies certain conditions, then strong duality will hold, which means that solving the dual problem is equivalent to solving the primal problem.
- **Solving the dual problem** can be **more computationally efficient and easier** than the primal problem.

Illustration of Primal Vs. Dual Optimization

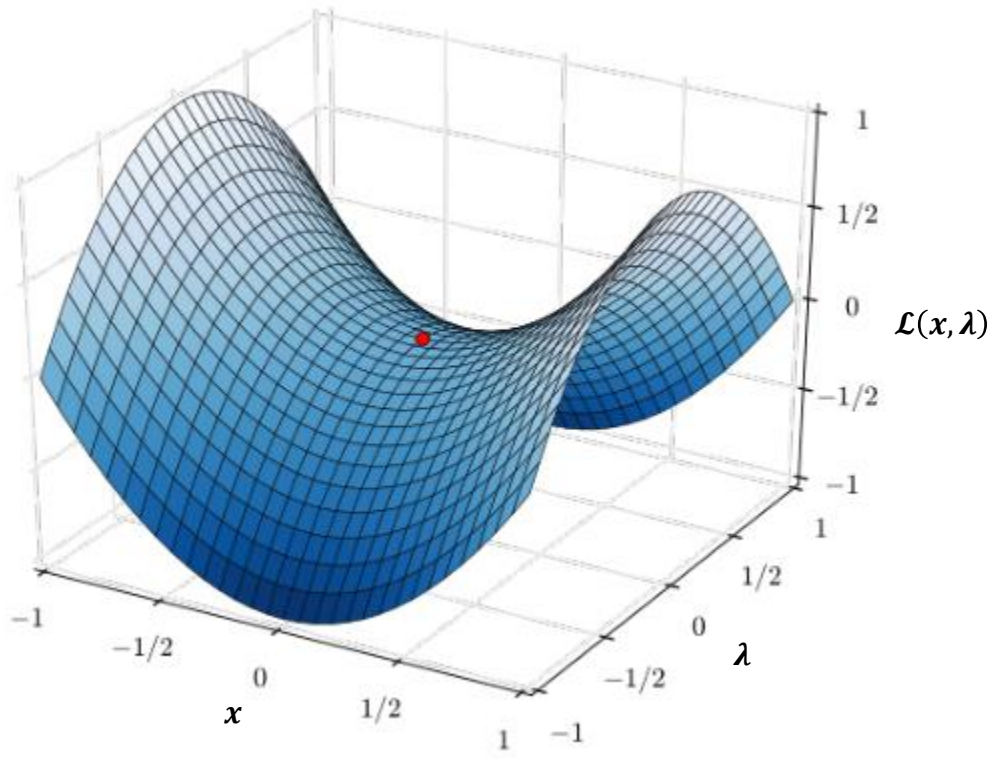


Illustration of the surface of the Lagrangian function $\mathcal{L}(x, \lambda)$ for scalars $x \in \mathbb{R}$ and $\lambda \in \mathbb{R}$. The red dot is a stationary point (a saddle point) where $\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial \lambda} = 0$. Therefore, fixing a value of λ and minimizing $\mathcal{L}(x, \lambda)$ over x is equivalent to solving the primal problem. Similarly, fixing a value of x and maximizing $\mathcal{L}(x, \lambda)$ over λ is equivalent to solving the dual problem. Thus, the saddle point (x^*, λ^*) for the Lagrangian is a point such that x^* is the optimal solution to the primal problem and λ^* is the optimal solution to the dual problem.

Dual Formulations for Hard & Soft SVMs

Deriving the Dual of Hard-SVM

Recall the Hard-SVM formulation derived earlier. Let $\alpha_i \geq 0$ denote Lagrange multipliers associated with the m inequality constraints given by $y^{(i)}\left(w^T x^{(i)} - b\right) - 1 \geq 0$, for all $i \in \{1,2,\dots,m\}$. Then, we can define the **Lagrangian function corresponding to the Hard-SVM** formulation as

$$\mathcal{L}(w,b,\alpha) \coloneqq \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i \left(y^{(i)}\left(w^T x^{(i)} - b\right) - 1\right).$$

Further, letting $\hat{\mathcal{L}}(\alpha) \coloneqq \min_{w,b} \mathcal{L}(w,b,\alpha)$, we can define the corresponding **dual optimization problem** as

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^m} \quad & \hat{\mathcal{L}}(\alpha) \\ \text{s. t.} \quad & \alpha_i \geq 0, \qquad \text{for all } i \in \{1,2,\dots,m\}. \end{aligned}$$

Now, to derive a "usable" expression for the dual objective function $\hat{\mathcal{L}}(\alpha)$. Notice that for any fixed α, the function $\mathcal{L}(w,b,\alpha)$ is convex in the primal variables w and b (since $\frac{1}{2}\|w\|^2$ is convex and the terms corresponding to the constraints are linear). Thus, we can determine the optimal primal values w^* and b^* by solving for them when setting their corresponding partial derivatives equal to 0. In this way, we have

$$\frac{\partial \mathcal{L}(w,b,\alpha)}{\partial w} = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0 \quad \rightarrow \quad w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}.$$

$$\frac{\partial \mathcal{L}(w,b,\alpha)}{\partial b} = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0. \qquad \text{(Balancing Equation)}$$

Dual-Formulation of Hard-SVM

Using the derivations on the previous slide, one can show (Proof as Homework) that the dual objective function is given by

$$\hat{\mathcal{L}}(\alpha) \coloneqq \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} \left({x^{(i)}}^T x^{(j)} \right).$$

Using this along with the balancing equation, we can formally state the dual problem of Hard-SVM.

Dual-Formulation of Hard-SVM

The **dual form of the Hard-Margin formulation of SVM** is defined as

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^m} \quad & \hat{\mathcal{L}}(\alpha) \\ \text{s.t.} \quad & \alpha_i \geq 0, \qquad \text{for all } i \in \{1,2,\dots,m\}, \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

Where $\hat{\mathcal{L}}(\alpha)$ is defined as

$$\hat{\mathcal{L}}(\alpha) \coloneqq \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} \left({x^{(i)}}^T x^{(j)} \right).$$

Dual-Formulation of Soft-SVM

Using the same process that we used to derive the dual problem for Hard-SVM, we can derive the dual problem for Soft-SVM by first defining the Lagrangian as

$$\mathcal{L}(w, b, \xi, \alpha, \mu) := \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i\left(y^{(i)}\left(w^T x^{(i)} - b\right) - (1 - \xi_i)\right) - \sum_{i=1}^{m}\mu_i\xi_i\,,$$

where $\alpha_i \geq 0$ are the Lagrange multipliers associated with the constraints $y^{(i)}\left(w^T x^{(i)} - b\right) - (1 - \xi_i) \geq 0$ and $\mu_i \geq 0$ are the Lagrange multipliers associated with the constraints $\xi_i \geq 0$. The following result can be shown (Proof as Homework).

Dual Formulation of Soft-SVM

The **dual form of the Soft-Margin formulation of SVM** is defined as

$$\begin{aligned}
\max_{\alpha\in\mathbb{R}^m,\,\mu\in\mathbb{R}^m}\ & \hat{\mathcal{L}}(\alpha) \\
\text{s.t.}\quad & \alpha_i \geq 0, \mu_i \geq 0, && \text{for all } i \in \{1,2,\dots,m\}, \\
& \sum_{i=1}^{m}\alpha_i y^{(i)} = 0, \\
& C - \alpha_i - \mu_i = 0, && \text{for all } i \in \{1,2,\dots,m\},
\end{aligned}$$

Where $\hat{\mathcal{L}}(\alpha)$ is defined as

$$\hat{\mathcal{L}}(\alpha) := \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y^{(i)}y^{(j)}\left(x^{(i)^T}x^{(j)}\right).$$

Training SVMs

Sequential Minimal Optimization:

A Fast Algorithm for Training Support Vector Machines

John C. Platt
Microsoft Research
jplatt@microsoft.com
Technical Report MSR-TR-98-14
April 21, 1998
© 1998 John Platt

ABSTRACT

This paper proposes a new algorithm for training support vector machines: *Sequential Minimal Optimization*, or *SMO*. Training a support vector machine requires the solution of a very large quadratic programming (QP) optimization problem. SMO breaks this large QP problem into a series of smallest possible QP problems. These small QP problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop. The amount of memory required for SMO is linear in the training set size, which allows SMO to handle very large training sets. Because matrix computation is avoided, SMO scales somewhere between linear and quadratic in the training set size for various test problems, while the standard chunking SVM algorithm scales somewhere between linear and cubic in the training set size. SMO's computation time is dominated by SVM evaluation, hence SMO is fastest for linear SVMs and sparse data sets. On real-world sparse data sets, SMO can be more than 1000 times faster than the chunking algorithm.

1. INTRODUCTION

In the last few years, there has been a surge of interest in Support Vector Machines (SVMs) [19] [20] [4]. SVMs have empirically been shown to give good generalization performance on a wide variety of problems such as handwritten character recognition [12], face detection [15], pedestrian detection [14], and text categorization [9].

However, the use of SVMs is still limited to a small group of researchers. One possible reason is that training algorithms for SVMs are slow, especially for large problems. Another explanation is that SVM training algorithms are complex, subtle, and difficult for an average engineer to implement.

We have just spent a decent amount of effort deriving the dual optimization problems of the corresponding primal forms for both Hard and Soft SVM. **Why?**

Reason (1): When it comes to "training" an SVM (i.e., solving these specific optimization problems), the **dual problems are much easier to solve** than their corresponding primal counterparts.

- Specifically, the dual formulations that we have derived are known as **convex quadratic programming** problems, for which many standard optimization algorithms can be utilized to solve them efficiently.

Reason (2): The dual problems for SVM have a useful formulation that can efficiently implement what is referred to as the **"kernel trick"** which allows SVMs to learn nonlinear decision boundaries (to be discussed).

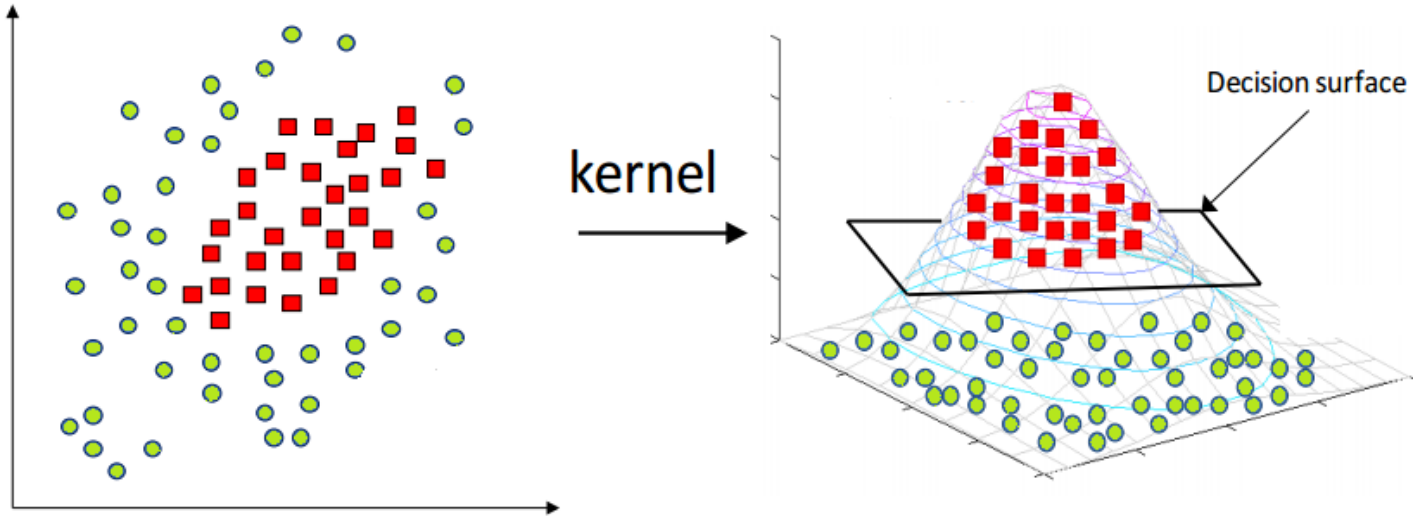
The Sequential Minimal Optimization (SMO) Algorithm

The SMO Algorithm was developed in 1998 by John Platt and is the preferred method for solving the SVM optimization formulations. The main idea is that instead of solving a very large quadratic program (QP), SMO breaks this QP into a series of smaller QPs that can be solved analytically, which drastically reduces the memory and time required to solve these problems.

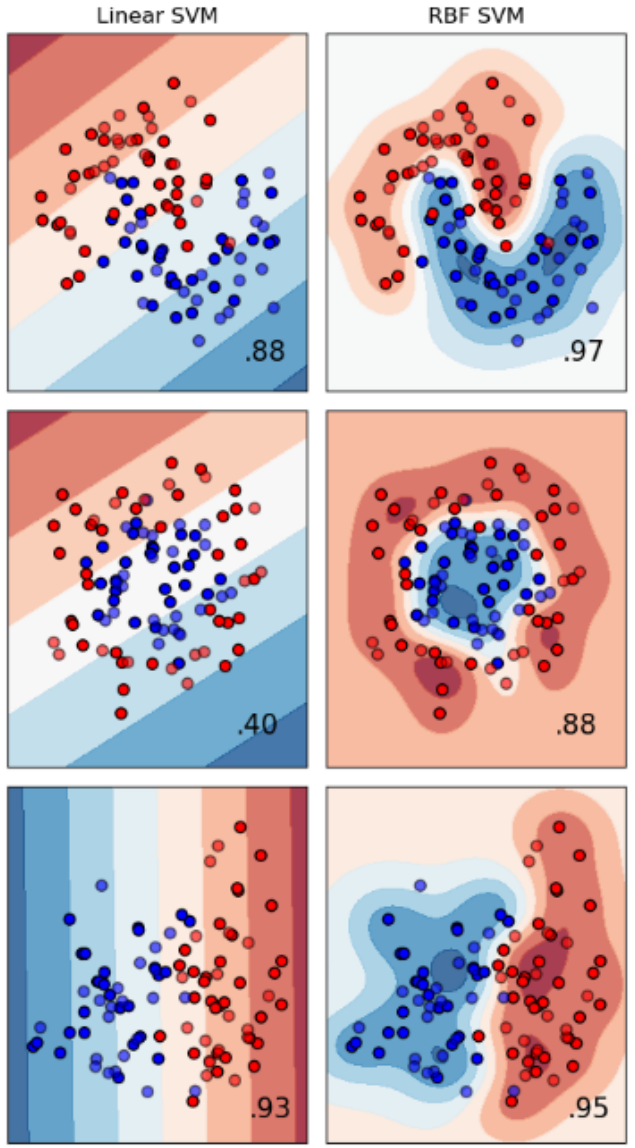
Nonlinear Decision Boundaries with SVMs

Nonlinear Boundaries Via Higher Dimensions

- The SVMs we have developed so far have only been learning linear decision boundaries in the feature space. However, like what polynomial regression does, once can make this procedure more flexible by **enlarging the feature space** into **higher dimensions**.
- The idea is that generally linear decision boundaries (defined via hyperplanes) in the enlarged space can identify more **sophisticated** target **class separations** which, when projected back to the original feature space, will yield **nonlinear decision boundaries**.



See Reference Slide

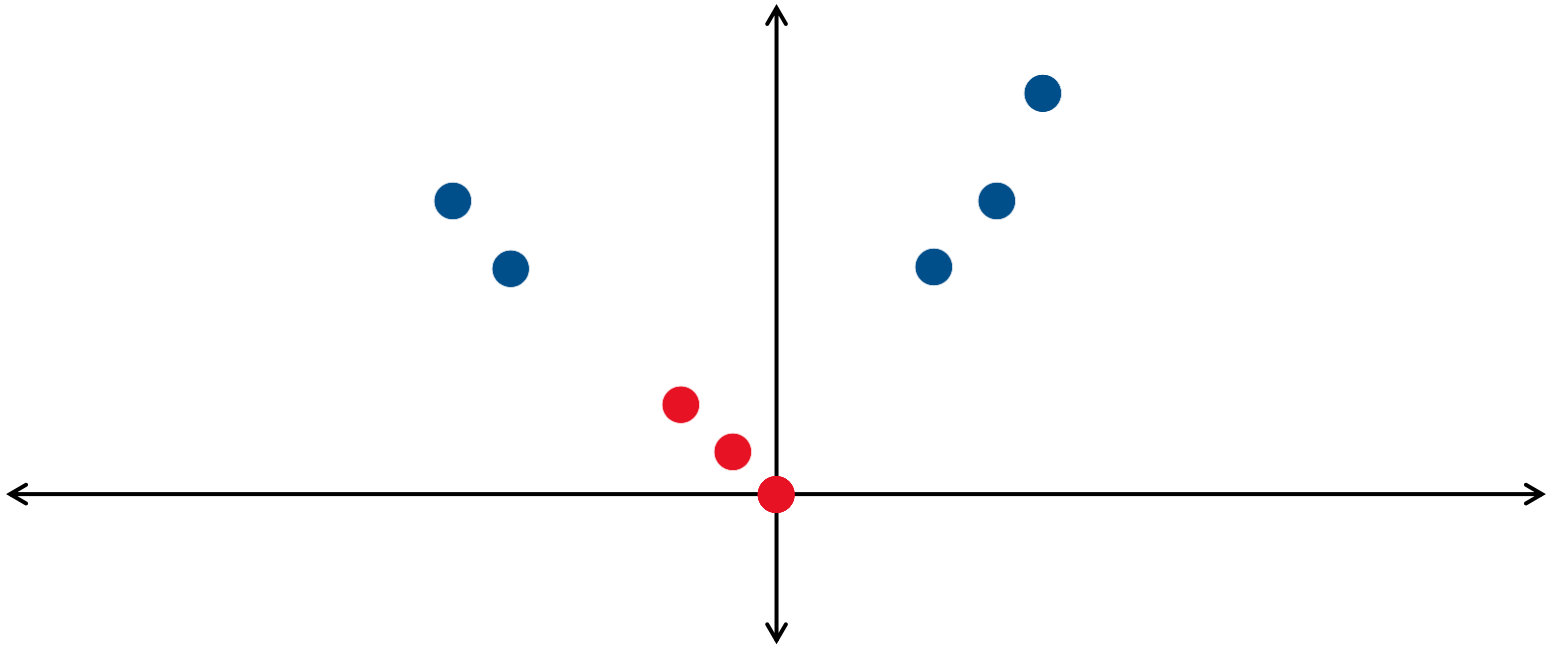


Decision Boundaries in Higher Dimensions

Consider the following dataset of feature label pairs $(x^{(i)}, y^{(i)})$ where $x^{(i)} \in \mathbb{R}$ (the features are 1-dimensional) and $y^{(i)} \in \{-1, 1\}$.

- Further, notice that the **data are NOT linearly separable**.

Decision Boundaries in Higher Dimensions

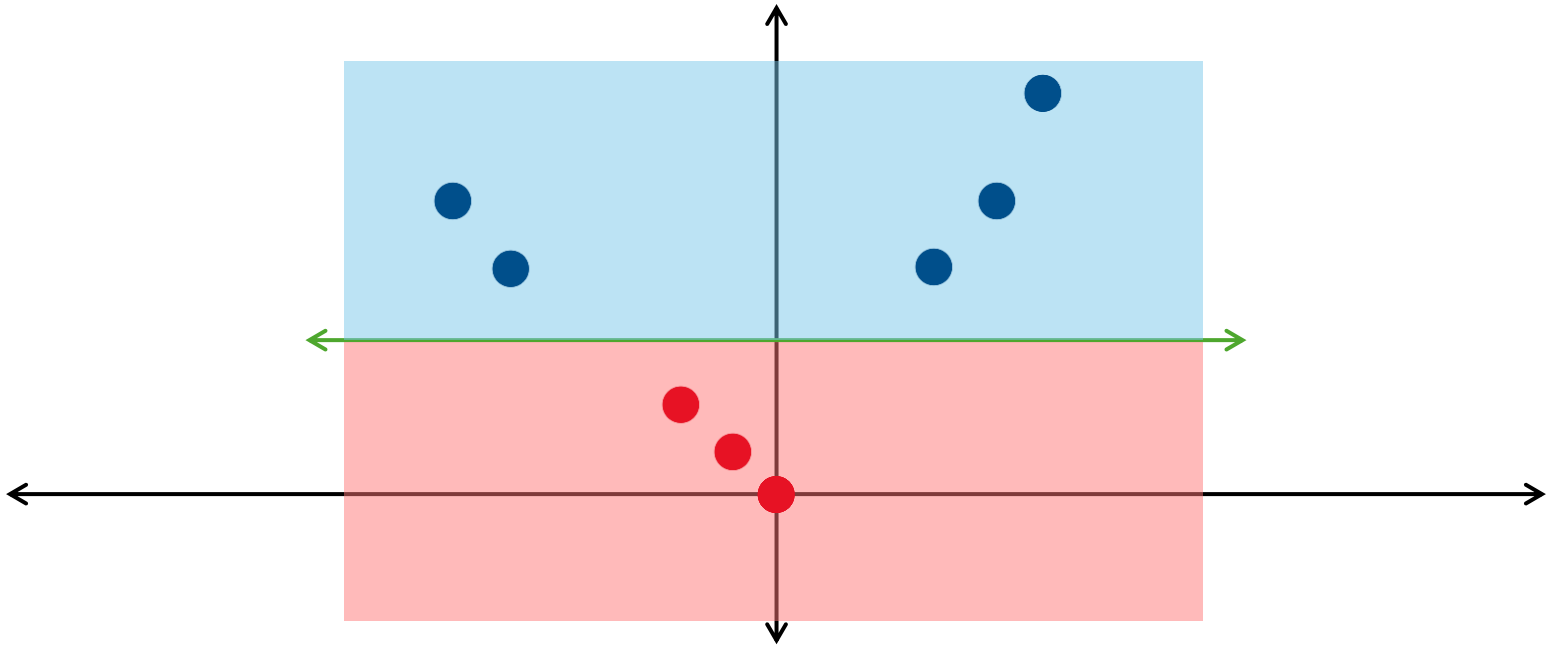


Consider the following dataset of feature label pairs $(x^{(i)}, y^{(i)})$ where $x^{(i)} \in \mathbb{R}$ (the features are 1-dimensional) and $y^{(i)} \in \{-1,1\}$.

- Further, notice that the **data are NOT linearly separable**.

Now, applying the transformation function $\psi(x) \coloneqq (x, x^2) \in \mathbb{R}^2$ (maps the 1-D space to a 2-D space).

Decision Boundaries in Higher Dimensions



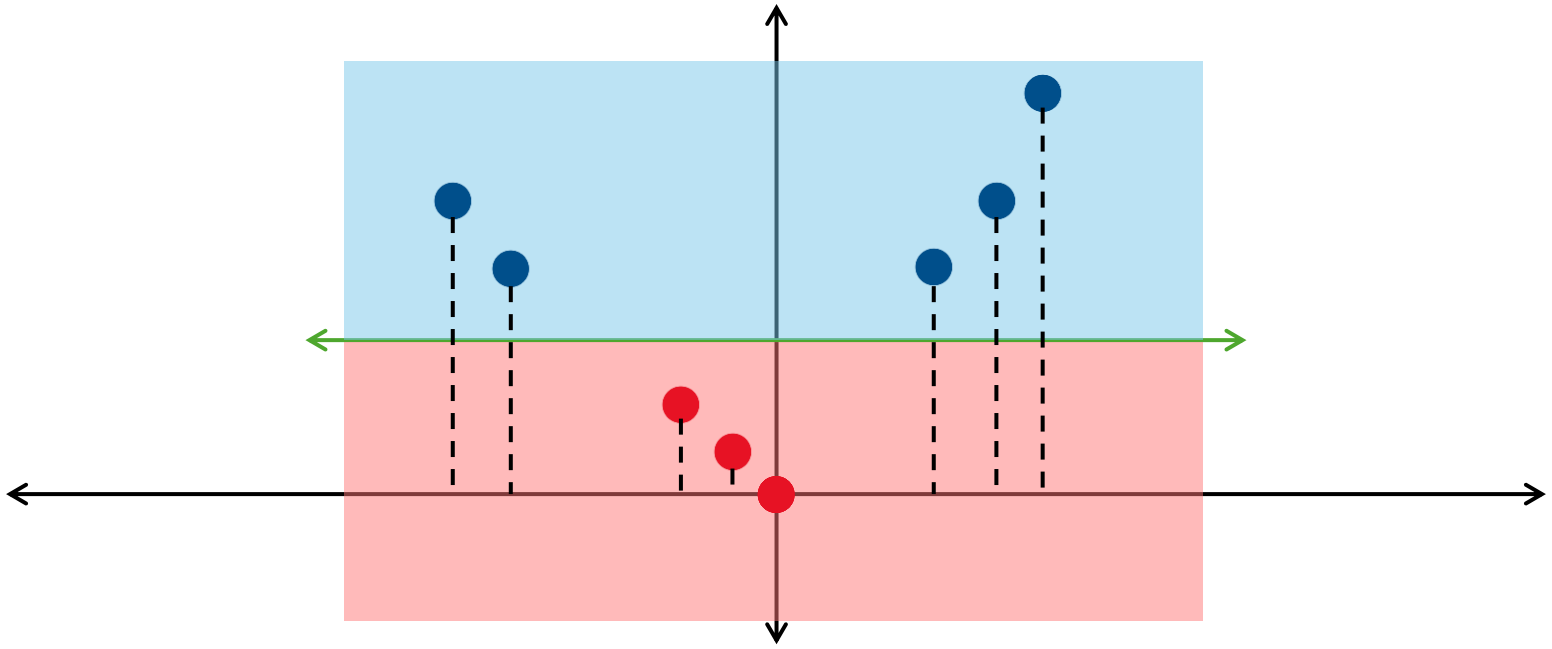
Consider the following dataset of feature label pairs $(x^{(i)}, y^{(i)})$ where $x^{(i)} \in \mathbb{R}$ (the features are 1-dimensional) and $y^{(i)} \in \{-1, 1\}$.

- Further, notice that the **data are NOT linearly separable**.

Now, applying the transformation function $\psi(x) \coloneqq (x, x^2) \in \mathbb{R}^2$ (maps the 1-D space to a 2-D space).

- This mapping yields a new feature space that **is indeed linearly separable**.

Decision Boundaries in Higher Dimensions



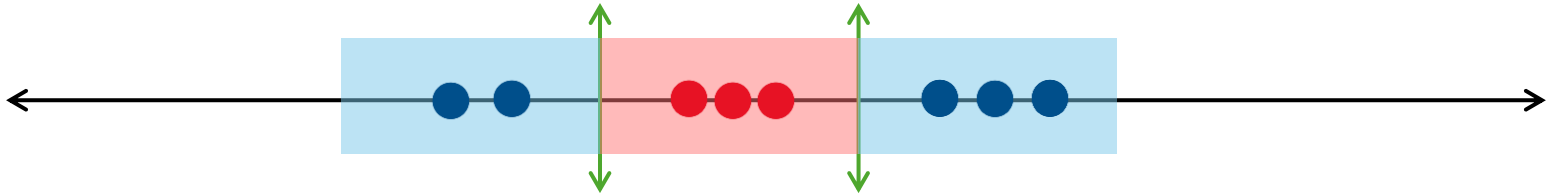
Consider the following dataset of feature label pairs $(x^{(i)}, y^{(i)})$ where $x^{(i)} \in \mathbb{R}$ (the features are 1-dimensional) and $y^{(i)} \in \{-1, 1\}$.

- Further, notice that the **data are NOT linearly separable**.

Now, applying the transformation function $\psi(x) \coloneqq (x, x^2) \in \mathbb{R}^2$ (maps the 1-D space to a 2-D space).

- This mapping yields a new feature space that **is indeed linearly separable**.
- When **projecting** the data back to the original feature space, the newly learned decision boundary can yield **nonlinearity**.

Decision Boundaries in Higher Dimensions



Consider the following dataset of feature label pairs $(x^{(i)}, y^{(i)})$ where $x^{(i)} \in \mathbb{R}$ (the features are 1-dimensional) and $y^{(i)} \in \{-1,1\}$.

- Further, notice that the **data are NOT linearly separable**.

Now, applying the transformation function $\psi(x) \coloneqq (x, x^2) \in \mathbb{R}^2$ (maps the 1-D space to a 2-D space).

- This mapping yields a new feature space that **is indeed linearly separable**.
- When **projecting** the data back to the original feature space, the newly learned decision boundary can yield **nonlinearity**.

Kernel Functions

A kernel function a measure of the similarity between two vectors in a (potentially high-dimensional) feature space. The definition of a kernel function can be defined as the following.

Kernel Function

Let $\psi: \mathbb{R}^n \to \mathbb{R}^k$ be a **mapping** from some feature space $\mathbb{R}^n$ to a different feature space $\mathbb{R}^k$ (typically of a higher dimension). Then, a **kernel** $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a function that computes a "**similarity**" score between two input vectors. Formally, the kernel K of two vectors $c \in \mathbb{R}^n$ and $d \in \mathbb{R}^n$ is defined as the dot-product of the two vectors in the new feature space obtained from the mapping ψ, i.e.,

$$K(c, d) \coloneqq \langle \psi(c), \psi(d) \rangle = \psi(c)^T \psi(d).$$

The reason that the kernel can be interpreted as a **measure of similarity** between two datapoints can be discerned by recalling the definition of the dot-product (i.e., $\psi(c)^T \psi(d) = \|\psi(c)\| \cdot \|\psi(d)\| \cdot \cos\theta$ where θ is the angle between $\psi(c)$ and $\psi(d)$). The interpretation of the kernel value also depends on the type of kernel used (note that each of the following kernels have their own mapping functions ψ).

- **Linear Kernel:** This kernel is simply the dot-product of the two vectors and measures how aligned (or similar) the datapoints are in the original space, i.e., $K(c, d) = c^T d$.
- **p-th Degree Polynomial Kernel:** This kernel measures the similarity in a higher-dimensional space created by a polynomial expansion, i.e., $K(c, d) = (c^T d)^p$.
- **Radial Basis Function (RBF) or Gaussian Kernel:** This kernel measures the actual distance between two vectors, with closer vectors having values near 1 and distant vectors having values near 0, i.e., $K(c, d) \coloneqq \exp\left(\frac{-\|c-d\|^2}{2\sigma^2}\right)$.

The Kernel Trick

- We've introduced the ides of utilizing a function ψ to map a feature vector $x^{(i)}$ into a higher dimensional space (possibly infinitely-dimensional) to aid in identifying a linear separation boundary between target classes.

- However, explicitly computing the mapping $\psi\left(x^{(i)}\right)$, for all $i \in \{1,2,\dots,m\}$, can be VERY computationally expensive…

- This is where kernel functions com in, which can perform this higher-dimensional mapping implicitly without requiring to explicitly compute the mappings $\psi\left(x^{(i)}\right)$. To see how this is the case, consider the following example:
  - **Example:** Let $K(c,d) = (c^T d)^2$ denote the quadratic polynomial kernel, where $c \in \mathbb{R}^2$ and $d \in \mathbb{R}^2$. Further, we can see that $\psi(c)^T = [c_1c_1, c_1c_2, c_2c_1, c_2c_2]$ denotes the polynomial mapping corresponding to his kernel function:

$$K(c,d) = (c^T d)^2 = \left(\sum_{i=1}^{2} c_i d_i\right)^2 = \sum_{i=1}^{2}\sum_{j=1}^{2}(c_i c_j)(d_i d_j) = \langle \psi(c), \psi(d)\rangle = \psi(c)^T \psi(d).$$

    Therefore, one can simply use the equation $(c^T d)^2$ instead of having to explicitly perform the mapping $\psi(c)^T\psi(d)$ (although they will both return the same value, using the kernel function is much cheaper).

This technique of utilizing a kernel function instead of explicitly computing the dot-product of vectors in a higher dimensional space obtained from ψ is called the "**Kernel Trick**".

Kernel Tricks for the Dual SVM Formulations

The kernel trick requires the computation of a dot-product between two vectors. This is where the dual formulations of SVM come into play. Recall that the dual objective functions for both the Hard and Soft formulations of SVM are equivalent, given as

$$\hat{\mathcal{L}}(\alpha) \coloneqq \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} \left({x^{(i)}}^T x^{(j)} \right).$$

Notice that this objective function requires the computation of the dot-product between the two feature vectors $x^{(i)}$ and $x^{(j)}$. Since we are interested in learning a decision boundary in a higher dimensional feature space, we would be required to explicitly compute the mapping $\psi\left(x^{(i)}\right)$ for all $i \in \{1,2, \dots, m\}$. This would amount to altering the objective function to

$$\hat{\mathcal{L}}(\alpha) \coloneqq \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} \left\langle \psi\left(x^{(i)}\right), \psi\left(x^{(j)}\right) \right\rangle.$$

However, this is where the **kernel trick can be applied**, instead of computing the dot-product $\left\langle \psi\left(x^{(i)}\right), \psi\left(x^{(j)}\right) \right\rangle$ by explicitly computing the mapping ψ, we can use an expression for the corresponding kernel function. Letting $K_{ij} \coloneqq K\left(x^{(i)}, x^{(j)}\right) = \left\langle \psi\left(x^{(i)}\right), \psi\left(x^{(j)}\right) \right\rangle$, we can simply write the **dual objective** for these SVM formulations as

$$\boldsymbol{\hat{\mathcal{L}}(\alpha) \coloneqq \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y^{(i)} y^{(j)} K_{ij}}.$$

References

- https://scikit-learn.org/stable/modules/svm.html
- https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
- https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py
- https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d