Machine Learning
Support Vector Machine

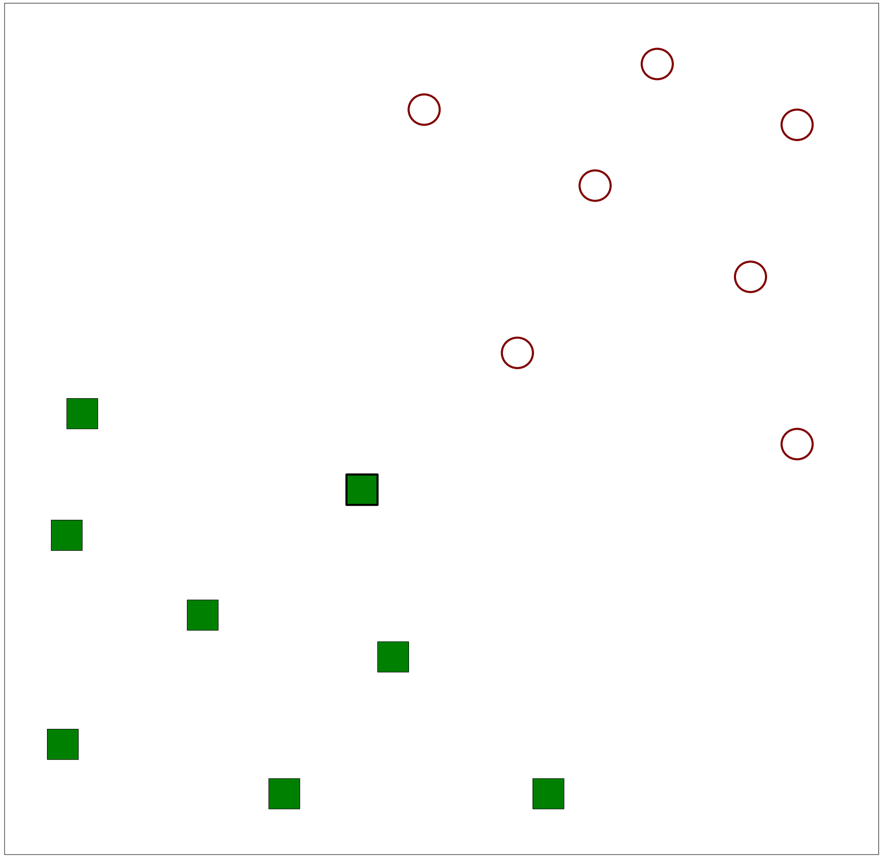
Dr. Pratyay Kuila

Dept. of Computer Science & Engineering
NIT Sikkim, Ravangla-737139

Introduction

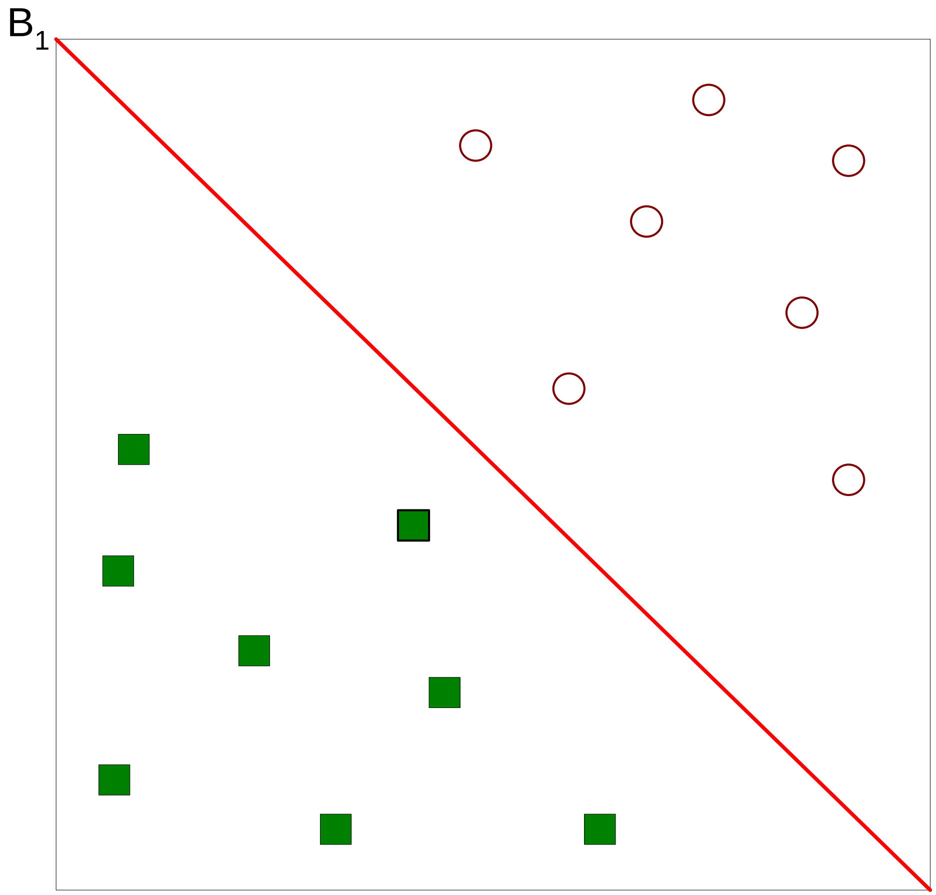
- *Support vector machines* (*SVMs*) are naturally defined for binary classification of numeric data.
- The binary-class problem can be generalized to the multiclass case by using a variety of tricks.
- Categorical feature variables can also be addressed.
- It is assumed that the class labels are drawn from $\{-1, 1\}$.
- With all linear models, SVMs use separating hyperplanes as the decision boundary between the two classes.
- The optimization problem of determining these hyperplanes is set up with the notion of ***margin***.
- Intuitively, a ***maximum margin hyperplane*** is one that cleanly separates the two classes.

SVM: Linearly Separable Data



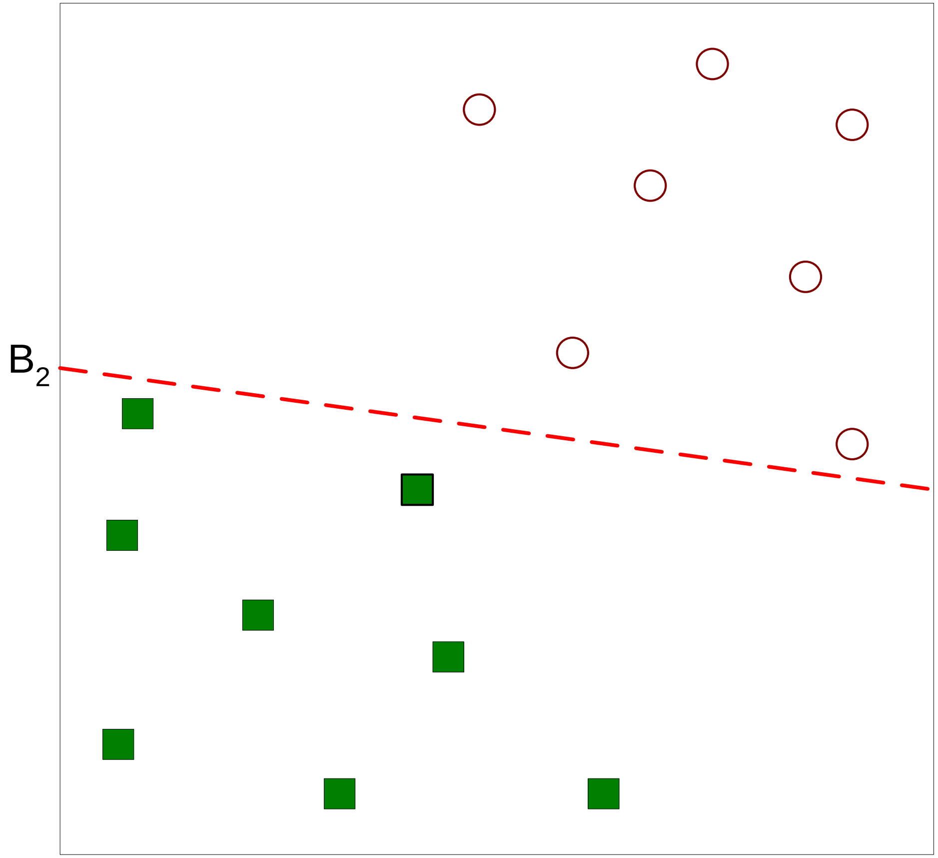
- In *linearly separable data*, it is possible to construct a linear hyperplane which cleanly separates data points belonging to the two classes.
- Find a *linear hyperplane* (decision boundary) that will separate the data.

SVM: Linearly Separable Data



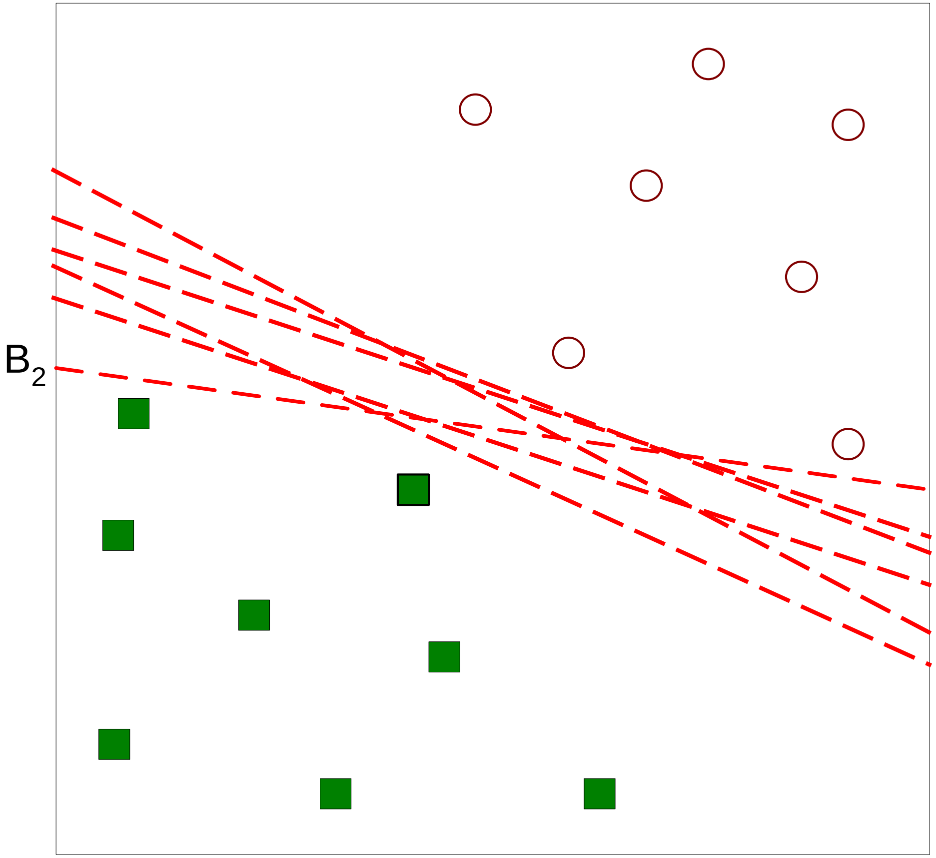
- One Possible Solution.

SVM: Linearly Separable Data



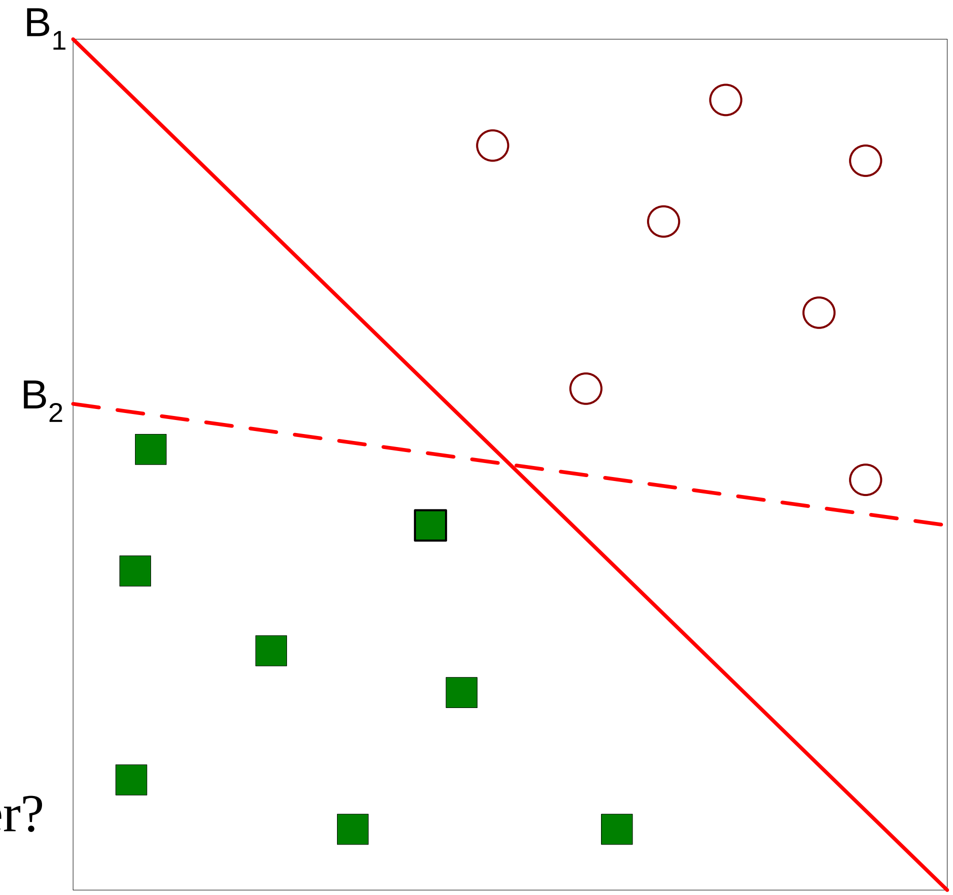
➢ Another possible solution.

SVM: Linearly Separable Data



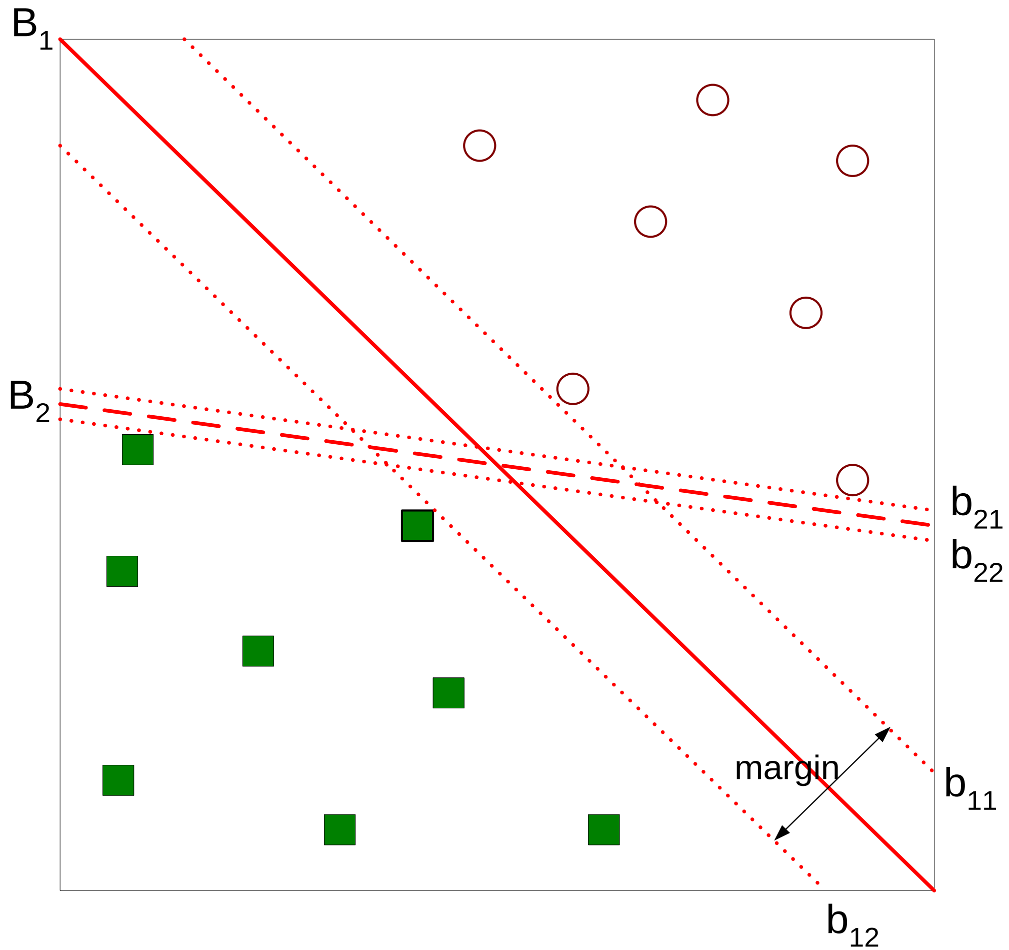
➢ Other possible solutions.

SVM: Linearly Separable Data



- Which one is better?
- B_1 or B_2?
- Both decision boundaries can separate the training examples into their respective classes without committing any misclassification errors.
- How do you define better?

SVM: Linearly Separable Data



➢ Find the hyperplane that *maximizes* the *margin*. B_1 is better than B_2.

SVM: Linearly Separable Data

Rationale for Maximum Margin

- Decision boundaries with *large margins* tend to have better generalization errors than those with small margins.
- Intuitively, if the margin is small, then any slight perturbations to the decision boundary can have quite a significant impact on its classification.
- Classifiers that produce decision boundaries with small margins are therefore more susceptible to model overfitting and tend to generalize poorly on previously unseen examples.
- The capacity of a linear model is inversely related to its margin. Models with small margins have higher capacities because they are more flexible and can fit many training sets, unlike models with large margins.
- However, according to the structural risk minimization (SRM) principle, as the capacity increases, the generalization error bound will also increase. Therefore, it is desirable to design linear classifiers that maximize the margins of their decision boundaries in order to ensure that their worst-case generalization errors are minimized.
- One such classifier is the **linear SVM**.

Linear SVM

- A linear SVM is a classifier that searches for a hyperplane with the largest margin, which is why it is often known as a *maximal margin classifier*.

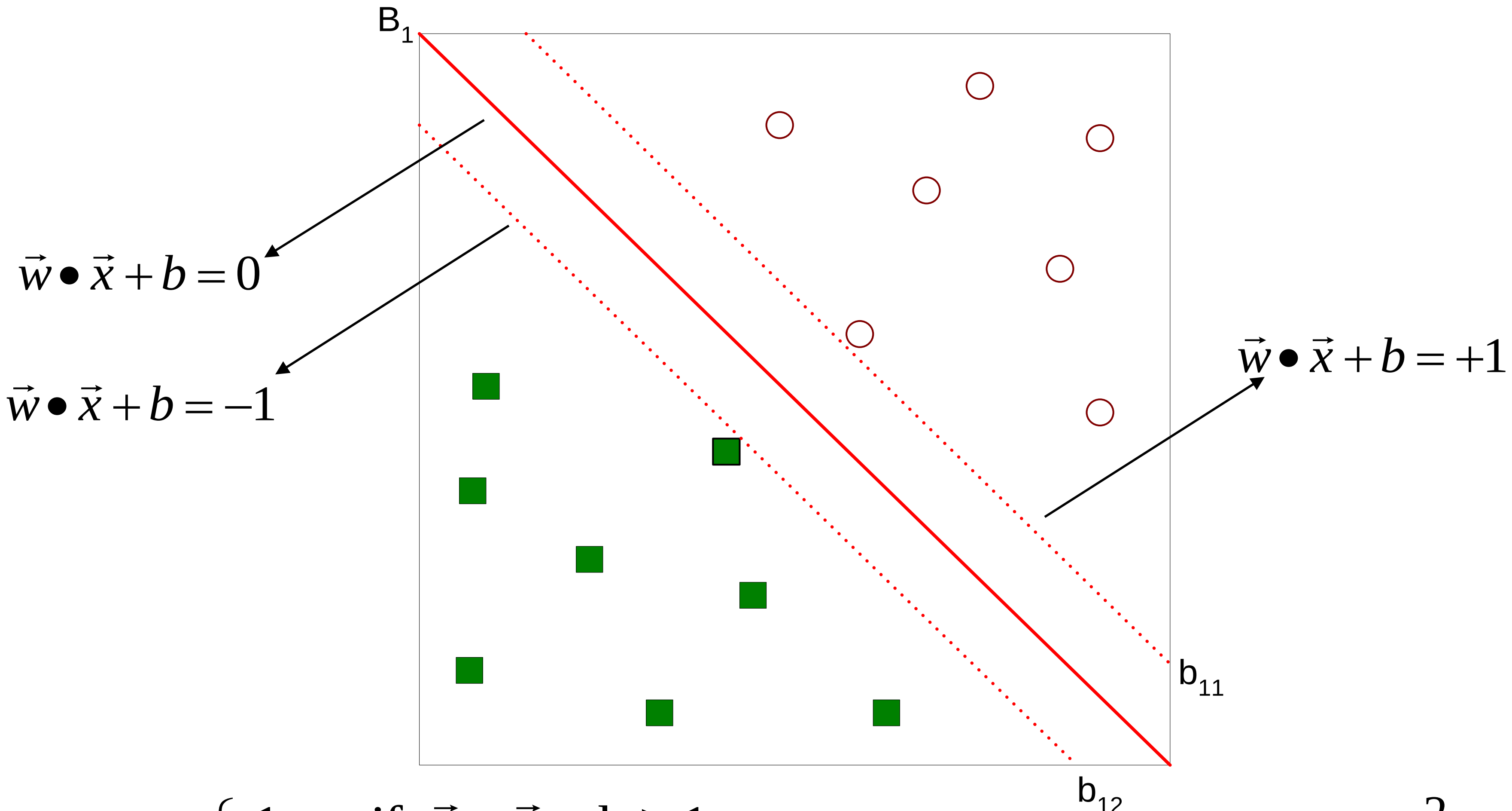
- ***Linear Decision Boundary*:** Consider a binary classification problem consisting of N training examples. Each example is denoted by a tuple $(\mathbf{x}_i, y_i)$ $(i = 1, 2, \ldots, N)$, where the vector $\mathbf{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}^T$ corresponds to the attribute set for the i^{th} example. By convention, let $y_i \in \{-1,+1\}$ denote its class label. The decision boundary of a *linear classifier* can be written in the following form:

$$(w_1 x_1 + w_2 x_2 + \ldots + w_D x_D) + b = 0$$
$$\text{Or, } \vec{\mathbf{w}}.\vec{\mathbf{x}} + b = 0 \qquad (a)$$

where $\vec{\mathbf{w}} = \{w_1, w_2, \ldots, w_D\}$ and b are parameters of the model.

- A decision boundary bisects the training examples into their respective.
- Any example located along the decision boundary must satisfy the Equation (a).

Linear SVM



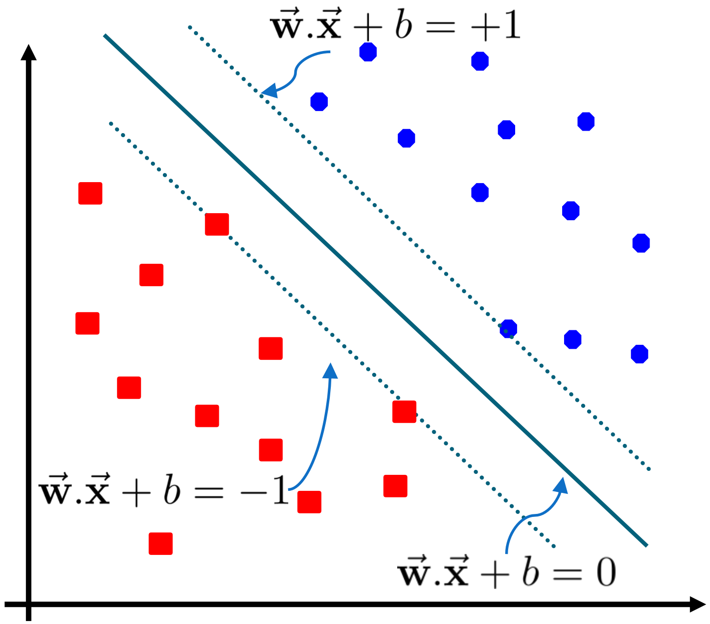
$$f(\vec{x})=\begin{cases}1 & \text{if } \vec{w}\bullet\vec{x}+b\geq 1\\ -1 & \text{if } \vec{w}\bullet\vec{x}+b\leq -1\end{cases}$$

$$\text{Margin}=\frac{2}{\|\vec{w}\|}$$

Linear SVM

- For example, if $\vec{\mathbf{x}_a}$ and $\vec{\mathbf{x}_b}$ are two points located on the decision boundary, then

$$\vec{\mathbf{w}}.\vec{\mathbf{x}_a} + b = 0$$
$$\vec{\mathbf{w}}.\vec{\mathbf{x}_b} + b = 0$$



- Subtracting the two equations will yield the following:

$$\vec{\mathbf{w}}.(\vec{\mathbf{x}_b} - \vec{\mathbf{x}_a}) = 0$$

where $(\vec{\mathbf{x}_b} - \vec{\mathbf{x}_a})$ is a vector parallel to the decision boundary and is directed from $\vec{\mathbf{x}_a}$ to $\vec{\mathbf{x}_b}$.

- Since the dot product is zero, the direction for $\vec{\mathbf{w}}$ must be perpendicular to the decision boundary (how?).
  - Let **u** and **v** be two vectors on the decision plane.
  - What is $\vec{\mathbf{w}}.(\vec{\mathbf{u}} - \vec{\mathbf{v}})$?
  - The vector **w** is also perpendicular to the plus and minus plane.

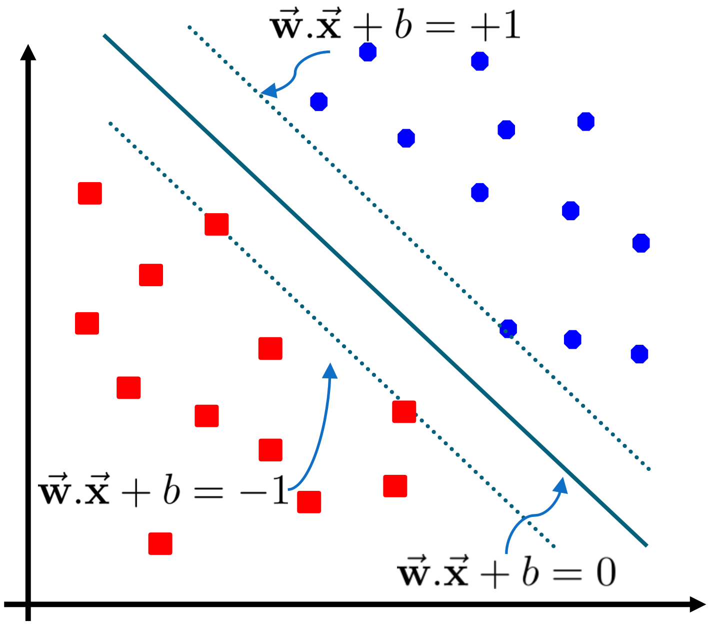
Linear SVM

- For any square $\vec{\mathbf{x}_s}$ located below the decision boundary, we can show that

$$\vec{\mathbf{w}}.\vec{\mathbf{x}_s} + b = k_1 \text{ where } k_1 < 0$$

- Similarly, for any circle $\vec{\mathbf{x}_c}$ located above the decision boundary, we can show that

$$\vec{\mathbf{w}}.\vec{\mathbf{x}_c} + b = k_2 \text{ where } k_2 > 0$$



- If we label all the squares as class +1 and all the circles as class −1, then we can predict the class label y for any test example $\mathbf{z}$ in the following way:

$$y = \begin{cases} +1, & \text{If } \vec{\mathbf{w}}.\vec{\mathbf{z}} + b < 0 \\ -1, & \text{If } \vec{\mathbf{w}}.\vec{\mathbf{z}} + b > 0 \end{cases}$$

Linear SVM: Margin of a Linear Classifier

- Consider the square and the circle that are closest to the decision boundary.

- Since the square is located below the decision boundary, it must satisfy the equation $\vec{\mathbf{w}}.\vec{\mathbf{x}_s} + b = k_1$ for some negative value k_1.

- Whereas the circle must satisfy the equation $\vec{\mathbf{w}}.\vec{\mathbf{x}_c} + b = k_2$ for some positive value k_2.

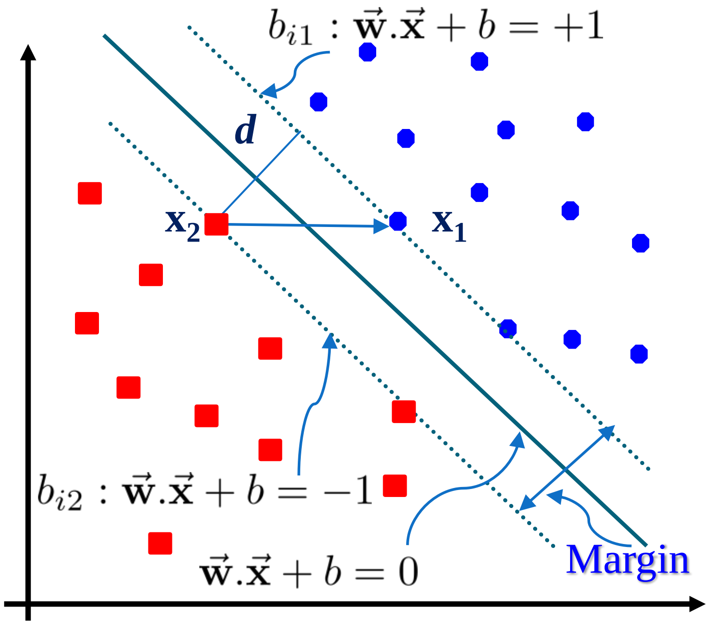
- We can rescale the parameters **w** and *b* of the decision boundary so that the two parallel hyperplanes b_{i1} and b_{i2} can be expressed as follows:

$$b_{i1} : \vec{\mathbf{w}}.\vec{\mathbf{x}} + b = +1 \qquad \text{(H1)}$$

$$b_{i2} : \vec{\mathbf{w}}.\vec{\mathbf{x}} + b = -1 \qquad \text{(H2)}$$

Linear SVM: Margin of a Linear Classifier

- The *margin* of the decision boundary is given by the distance between these two hyperplanes. To compute the *margin* (d), let $\mathbf{x}_1$ vector be a data point located on b_{i1} and $\mathbf{x}_2$ vector be a data point on b_{i2} as shown in the figure.



- *Claim*: $\vec{\mathbf{x}}_1 = \vec{\mathbf{x}}_2 + \lambda\vec{\mathbf{w}}$ for some value of λ. (Why? The line from $\mathbf{x}_2$ to $\mathbf{x}_1$ is perpendicular to the planes. So to get from $\mathbf{x}_2$ to $\mathbf{x}_1$, travel some distance in direction $\mathbf{w}$.)

- Therefore, we have,

1. $\vec{\mathbf{w}}.\vec{\mathbf{x}}_1 + b = 1$
2. $\vec{\mathbf{w}}.\vec{\mathbf{x}}_2 + b = -1$
3. $\vec{\mathbf{x}}_1 = \vec{\mathbf{x}}_2 + \lambda\vec{\mathbf{w}}$
4. $||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2|| = d$

Linear SVM: Margin of a Linear Classifier

- Therefore, we have, 1. $\vec{\mathbf{w}}.\vec{\mathbf{x}}_1 + b = 1$ 2. $\vec{\mathbf{w}}.\vec{\mathbf{x}}_2 + b = -1$

  3. $\vec{\mathbf{x}}_1 = \vec{\mathbf{x}}_2 + \lambda\vec{\mathbf{w}}$ 4. $||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2|| = d$

- It is now easy to get *d* in terms of **w** and *b*.

- From (1) and (3), $\vec{\mathbf{w}}.(\vec{\mathbf{x}}_2 + \lambda\vec{\mathbf{w}}) + b = 1$

$$\text{Or, } \vec{\mathbf{w}}.\vec{\mathbf{x}}_2 + b + \lambda\vec{\mathbf{w}}.\vec{\mathbf{w}} = 1$$

$$\text{Or, } -1 + \lambda\vec{\mathbf{w}}.\vec{\mathbf{w}} = 1$$

$$\text{Or, } \lambda = \frac{2}{\vec{\mathbf{w}}.\vec{\mathbf{w}}} = \frac{2}{||\vec{\mathbf{w}}||^2}$$

- From (3) and (4), $d = ||\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2|| = ||\lambda.\vec{\mathbf{w}}|| = \lambda.||\vec{\mathbf{w}}||$

$$\text{Or, } d = \frac{2}{||\vec{\mathbf{w}}||^2}||\vec{\mathbf{w}}|| = \frac{2}{\vec{\mathbf{w}}.\vec{\mathbf{w}}}\sqrt{\vec{\mathbf{w}}.\vec{\mathbf{w}}} = \frac{2}{\sqrt{\vec{\mathbf{w}}.\vec{\mathbf{w}}}}$$

$$\text{Or, } d = \frac{2}{\sqrt{\sum_{i=1}^{D} w_i^2}} = \frac{2}{||\vec{\mathbf{w}}||}$$

Learning a Linear SVM

- The training phase of SVM involves estimating the parameters **w** vector and *b* of the decision boundary from the training data.
- The parameters must be chosen in such a way that the following two conditions are met:

$$\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b \geq +1 \text{ if } y_i = 1$$
$$\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b \leq -1 \text{ if } y_i = -1$$

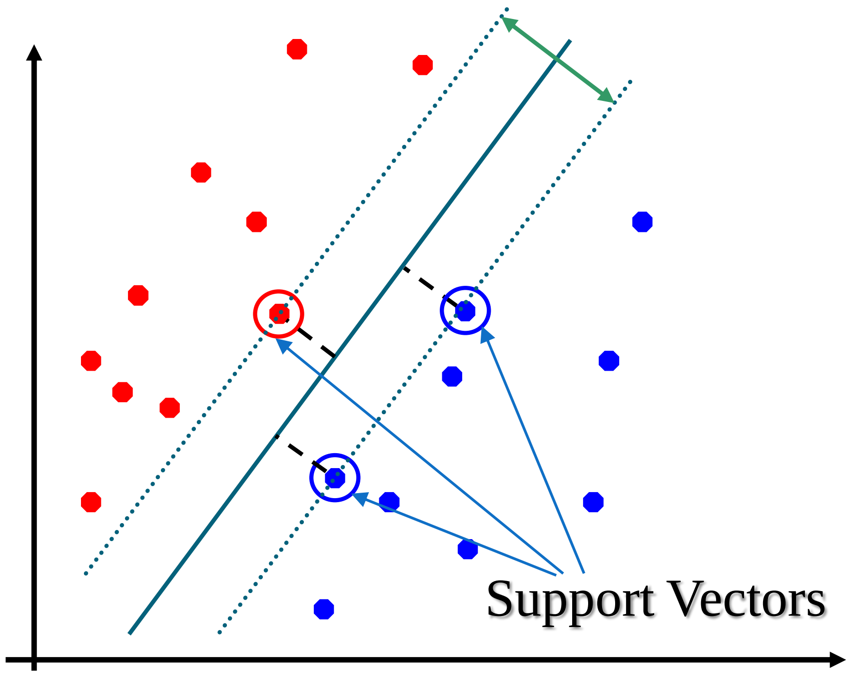
- These conditions impose the requirements that all training instances from class y = 1 (i.e., the squares) must be located on or below the hyperplane $\vec{\mathbf{w}}.\vec{\mathbf{x}} + b = -1$.

- While those instances from class y = −1 (i.e., the circles) must be located on or above the hyperplane $\vec{\mathbf{w}}.\vec{\mathbf{x}} + b = +1$.

Linear SVM

- The particular data points ($\mathbf{x}_i$, y_i) for which the first or second hyperplane is satisfied with the equality sign are called *support vectors*—hence the name "support vector machine."

- All the remaining examples in the training sample are completely irrelevant.



- In conceptual terms, the support vectors are those data points that lie closest to the optimal hyperplane and are therefore the most difficult to classify.

Learning a Linear SVM

- Both the following inequalities;

$$\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b \geq 1 \text{ if } y_i = 1$$
$$\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b \leq -1 \text{ if } y_i = -1$$

can be summarized in a more compact form as follows:

$$y_i(\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b) \geq 1,\ i = 1, 2, ..., N$$

- Although the conditions are also applicable to any linear classifiers (including perceptrons), SVM imposes an additional requirement that the margin of its decision boundary must be maximal.

- Objective is to maximize the margin $d = \frac{2}{||\vec{\mathbf{w}}||}$

- It is equivalent to minimize $f(\vec{\mathbf{w}}) = \dfrac{||\vec{\mathbf{w}}||^2}{2}$

Learning a Linear SVM: Separable Case

- **Separable Case**: The learning task in SVM can be formalized as the following constrained optimization problem:

$$\text{Minimize } f(\vec{\mathbf{w}}) = \frac{||\vec{\mathbf{w}}||^2}{2}$$

$$\text{Subject to } y_i(\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b) \geq 1,\ i = 1, 2, ..., N$$

- The objective function is quadratic and the constraints are linear in the parameters **w** and *b*, this is known as a *convex optimization problem*.

- It can be solved using the standard *Lagrange multiplier method*.

Learning a Linear SVM: Separable Case

➢ Solution to the Optimization Problem:

- First, we must rewrite the objective function in a form that takes into account the constraints imposed on its solutions.
- The new objective function is known as the Lagrangian for the optimization problem:

$$L_P = \frac{1}{2}||\vec{\mathbf{w}}||^2 - \sum_{i=1}^{N} \lambda_i \Big(y_i(\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b) - 1 \Big)$$

where the parameters λ_i, $1 \leq i \leq N$ are called the Lagrange multipliers.

- The first term in the Lagrangian is the same as the original objective function, while the second term captures the inequality constraints.
- It is easy to show that the function is minimized when $\vec{\mathbf{w}} = 0$, a null vector whose components are all zeros. Such a solution, violates the constraints because there is no feasible solution for b.

Learning a Linear SVM: Separable Case

- **Solution to the Optimization Problem:**
  - The solutions for **w** and b are infeasible if they violate the inequality constraints; i.e., if $y_i(\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b) - 1 < 0$.
  - The Lagrangian given in the above equation incorporates this constraint by subtracting the term from its original objective function.
  - Assuming that $\lambda_i \geq 0$, it is clear that any infeasible solution may only increase the value of the Lagrangian.
  - To minimize the Lagrangian, we must take the derivative of L_P with respect to w_j and b and set them to zero:

$$\text{Condition 1: } \frac{\partial L_P}{\partial \vec{\mathbf{w}}} = 0 \Longrightarrow \vec{\mathbf{w}} = \sum_{i=1}^{N} \lambda_i y_i \vec{\mathbf{x}}_i$$

$$\text{Condition 2: } \frac{\partial L_P}{\partial b} = 0 \Longrightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

  - Above condition (1) can be re-written as:

$$\boxed{\frac{\partial L_P}{\partial w_j} = 0 \Longrightarrow w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij}, \forall j \in [1, D]}$$

Learning a Linear SVM: Separable Case

➢ **Solution to the Optimization Problem**:

- The solution vector **w** is defined in terms of an expansion that involves the N training examples. Note, however, that although this solution is unique by virtue of the convexity of the Lagrangian, the same cannot be said about the Lagrange multipliers λ_i, $1 \leq i \leq N$.
- As the Lagrange multipliers λ_i, $1 \leq i \leq N$ are unknown, we still cannot solve for **w** and b.
- The solutions for **w** and b are infeasible if they violate the inequality.
- It is also important to note that for all the constraints that are not satisfied as equalities, the corresponding multiplier λ_i must be zero.

$$\lambda_i[y_i(\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b) - 1] = 0, \forall i \in [1, N]$$
$$\text{i.e., } \lambda_i = 0 \text{ or } y_i(\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b) - 1 = 0$$

In other words, only those multipliers that exactly satisfy the condition can assume nonzero values. This property is a statement of the *Karush–Kuhn–Tucker (KKT) conditions*.

Learning a Linear SVM: Separable Case

➢ **Solution to the Optimization Problem (dual form):**

- The primal problem deals with a convex cost function and linear constraints. Given such a constrained-optimization problem, it is possible to construct another problem called the ***dual problem***.

- To derive the dual form of the constrained optimization problem, we plug into the Lagrangian (L_P in slide 21) the value of **w** in terms of the dual variables as expressed in *condition* 1 (slide 22). This yields,

$$
\begin{aligned}
L_D &= \frac{1}{2}||\vec{\mathbf{w}}||^2 - \sum_{i=1}^{N} \lambda_i \Big(y_i(\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b) - 1 \Big) \\
&= \frac{1}{2}(\vec{\mathbf{w}}.\vec{\mathbf{w}}) - \sum_{i=1}^{N} \lambda_i y_i \vec{\mathbf{w}}.\vec{\mathbf{x}}_i - b\sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i \\
&= \frac{1}{2}\sum_{i=1}^{N} \lambda_i y_i \vec{\mathbf{w}}.\vec{\mathbf{x}}_i - \sum_{i=1}^{N} \lambda_i y_i \vec{\mathbf{w}}.\vec{\mathbf{x}}_i - b\sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i \\
&= \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N} \lambda_i y_i \vec{\mathbf{w}}.\vec{\mathbf{x}}_i \;\; [\because \text{Condition 2: } \sum_{i=1}^{N} \lambda_i y_i = 0]
\end{aligned}
$$

$$
L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j (\vec{\mathbf{x}}_i.\vec{\mathbf{x}}_j)
$$

Learning a Linear SVM: Separable Case

➢ **Solution to the Optimization Problem (dual form):**

- We may now state the dual problem as follows:

Given the training sample, $\{\vec{\mathbf{x}}_i, y_i\}_{i=1}^{N}$ *find the Lagrange multipliers* $\{\lambda_i\}_{i=1}^{N}$ *with following objective:*

$$\text{Maximize } f(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j (\vec{\mathbf{x}}_i . \vec{\mathbf{x}}_j)$$

Subject to,

$$1.\ \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$2.\ \lambda_i \geq 0, \forall i \in [1, N]$$

- Unlike the primal optimization problem based on the Lagrangian (L_P in slide 21), the dual problem is cast entirely in terms of the training data.
- Moreover, the objective function to be maximized depends *only* on the input patterns in the form of a set of *dot products* $(\vec{\mathbf{x}}_i . \vec{\mathbf{x}}_i)$.

Learning a Linear SVM: Separable Case

➢ **Solution to the Optimization Problem (dual form)**:

- The support vectors constitute a subset of the training sample, which means that the solution vector is *sparse*.
- That is to say, constraint (2) of the dual problem is satisfied with the inequality sign for all the support vectors for which the λ_i, $1 \le i \le N$ are nonzero, and with the equality sign for all the other data points in the training sample, for which the λ_i are all zero.
- The Lagrangian dual L_D may be optimized by using the *gradient ascent* technique. The corresponding gradient-based update equation is as follows:

$$\lambda_i = \lambda_i + \eta \frac{\partial L_D}{\partial \lambda_i}$$

$$\text{where, } \frac{\partial L_D}{\partial \lambda_i} = 1 - y_i \sum_{j=1}^{N} \lambda_j y_j (\vec{\mathbf{x}}_i . \vec{\mathbf{x}}_j)$$

- The learning rate η may be chosen to maximize the improvement in objective function. The initial solution can be chosen to be the vector of zeros, which is also a feasible solution for λ.

Learning a Linear SVM: Separable Case

- **Solution to the Optimization Problem:**
  - After determining the optimum Lagrange multipliers, we may compute the optimum weight vector as follow:

$$w_j = \sum_{i=1}^{N_S} \lambda_i y_i x_{ij}, \forall j \in [1, D]$$

where N_S be the number of support vectors.

- Since support vectors lie on the marginal hyperplanes, for any support vector $\vec{\mathbf{x}}_i$, $(\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b) = y_i$ and thus b can be obtained:

$$b = y_i - (\vec{\mathbf{w}}.\vec{\mathbf{x}}_i)$$

Linear SVM: Example

***Example* 1.** Consider the two dimensional data set shown in the table, which contains eight training instances. We can solve the dual optimization problem to obtain the Lagrange multiplier λ_i for each training instance. The Lagrange multipliers are depicted in the last column of the table. Notice that only the first two instances have non-zero Lagrange multipliers. These instances correspond to the support vectors for this data set.

x_1	x_2	y	**Lagrange multiplier**
0.3858	0.4687	1	65.5261
0.4871	0.611	–1	65.5261
0.9218	0.4103	–1	0
0.7382	0.8936	–1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	–1	0
0.2146	0.0099	1	0

Let $\mathbf{w} = (w_1,w_2)$ and b denote the parameters of the decision boundary. We can solve for w_1 and w_2 for each support vector in the following way:

$$w_1 = \sum_{i=1}^{2} \lambda_i y_i x_{i1} = 65.5621 \times 1 \times 0.3858 + 65.5621 \times -1 \times 0.4871 = -6.64$$

$$w_2 = \sum_{i=1}^{2} \lambda_i y_i x_{i2} = 65.5621 \times 1 \times 0.4687 + 65.5621 \times -1 \times 0.611 = -9.32$$

Linear SVM: Example

***Example* 1 (Cont…):**
The bias term b can be computed for each support vector:

x_1	x_2	y	Lagrange multiplier
0.3858	0.4687	1	65.5261
0.4871	0.611	–1	65.5261
0.9218	0.4103	–1	0
0.7382	0.8936	–1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	–1	0
0.2146	0.0099	1	0

$$\begin{aligned} b^{(1)} &= 1 - \vec{\mathbf{w}}.\vec{\mathbf{x}}_1 \\ &= 1 - (-6.64)(0.3858) - (-9.32)(0.4687) \\ &= 7.9300 \\ b^{(2)} &= 1 - \vec{\mathbf{w}}.\vec{\mathbf{x}}_2 \\ &= -1 - (-6.64)(0.4871) - (-9.32)(0.611) \\ &= 7.9289 \end{aligned}$$

- Averaging these values, we obtain b = 7.93.
- The decision boundary corresponding to these parameters is follow:

$$-6.64x_1 - 9.32x_2 + 7.93 = 0$$

- Once the parameters of the decision boundary are found, a test instance **z** is classified as follows:

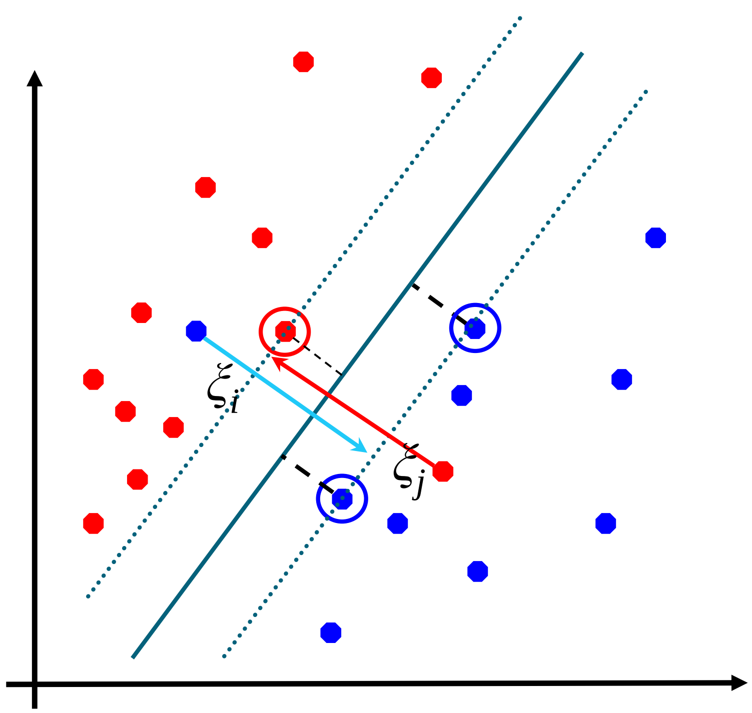
$$f(\vec{\mathbf{z}}) = sign(\vec{\mathbf{w}}.\vec{\mathbf{z}} + b) = sign\left(\sum_{i=1}^{N} \lambda_i y_i x_i.\vec{\mathbf{z}} + b\right)$$

- If f(**z**) = 1, then the test instance is classified as a positive class; otherwise, it is classified as a negative class.

Linear SVM: Non-separable Case

- If the training data is not *linearly separable*, *slack variables* ξ_i (> 0) can be added to allow misclassification of difficult or noisy examples.

$$\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b \geq +1 - \xi_i \text{ if } y_i = 1$$
$$\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b \leq -1 + \xi_i \text{ if } y_i = -1$$



- *Soft margin approach*: allow some errors. Let some points be moved to where they belong.
- The modified objective function is given by the following equation:

$$\text{Minimize } f(\vec{\mathbf{w}}) = \frac{||\vec{\mathbf{w}}||^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right)^k$$

- where C and k are user-specified parameters representing the penalty of misclassifying the training instances.

Linear SVM: Non-separable Case

- We assume $k = 1$ to simplify the problem. The parameter C can be chosen based on the model's performance on the validation set.
- The Lagrangian for this constrained optimization problem can be written as follows:

$$L_P = \frac{1}{2}||\vec{\mathbf{w}}||^2 + C\sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \lambda_i \Big(y_i(\vec{\mathbf{w}}.\vec{\mathbf{x}}_i + b) - 1 + \xi_i \Big) - \sum_{i=1}^{N} \mu_i \xi_i$$

where λ and μ are Lagrange multiplier.

- The constraints are,
  1. $\xi_i \geq 0,\ \lambda_i \geq 0,\ \mu_i \geq 0$
  2. $\lambda_i \Big(y_i(\vec{\mathbf{w}}\vec{\mathbf{x}}_i + b) - 1 + \xi_i \Big) = 0$
  3. $\mu_i \xi_i = 0$
- The Lagrange multiplier λ_i given in equation (2) is non-vanishing only if the training instance resides along the lines $\mathbf{w} \cdot \mathbf{x}_i + b = \pm 1$ or has $\xi_i > 0$. On the other hand, the Lagrange multipliers μ_i given in equation (3) are zero for any training instances that are misclassified (i.e., having $\xi_i > 0$).

Linear SVM: Non-separable Case

- Setting the first-order derivative of L_P with respect to **w**, b, and ξ_i to zero would result in the following equations:

$$\frac{\partial L_P}{\partial w_j} = w_j - \sum_{i=1}^{N} \lambda_i y_i x_{ij} = 0 \Longrightarrow w_j = \sum_{i=1}^{N} \lambda_i y_i x_{ij}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{N} \lambda_i y_i = 0 \Longrightarrow \sum_{i=1}^{N} \lambda_i y_i = 0$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \lambda_i - \mu_i = 0 \Longrightarrow \lambda_i + \mu_i = C$$

- The dual Lagrangian can be derived as it is done previously.

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j (\vec{\mathbf{x}}_i . \vec{\mathbf{x}}_j)$$

Linear SVM: Non-separable Case

- It turns out to be identical to the dual Lagrangian for linearly separable data. Nevertheless, the constraints imposed on the Lagrange multipliers λ_i's are slightly different those in the linearly separable case.

- In the linearly separable case, the Lagrange multipliers must be non-negative, i.e., $\lambda i \geq 0$. On the other hand, the equation ($\lambda_i + \mu_i = C$) suggests that λ_i should not exceed C (since both μ_i and λ_i are non-negative).

- Therefore, the Lagrange multipliers for nonlinearly separable data are restricted to $0 \leq \lambda i \leq C$.

- The dual problem can then be solved to obtain the Lagrange multipliers λ_i. These multipliers can be replaced into the obtained value od w_j to obtain the parameters of the decision boundary.

Gradient (or Steepest) Descent Algorithm for SVM

Learning an SVM has been formulated as a *constrained optimization* problem over **w** and ξ.

$$\text{Min } f(\vec{\mathbf{w}}) = \frac{||\vec{\mathbf{w}}||^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right) \text{ subject to } y_i(\vec{\mathbf{w}}\vec{\mathbf{x}}_i + b) \geq 1 - \xi_i, \forall i \in [1, N]$$

The constraint $y_i(\vec{\mathbf{w}}\vec{\mathbf{x}}_i + b) \geq 1 - \xi_i$, can be written more concisely as

$$y_i f(\vec{\mathbf{x}}_i) \geq 1 - \xi_i$$

which, together with $\xi_i \geq 0$, is equivalent to

$$\xi_i = \text{Max}(0, 1 - y_i f(\vec{\mathbf{x}}_i))$$

Hence, the learning problem is equivalent to the *unconstrained optimization* problem over **w**.

$$\text{Min } f(\vec{\mathbf{w}}) = \frac{||\vec{\mathbf{w}}||^2}{2} + C\left(\sum_{i=1}^{N} \text{Max}(0, 1 - y_i f(\vec{\mathbf{x}}_i))\right)$$

where $\text{Max}(0, 1 - y_i f(\vec{\mathbf{x}}_i))$ be the loss function.

Gradient (or Steepest) Descent Algorithm for SVM

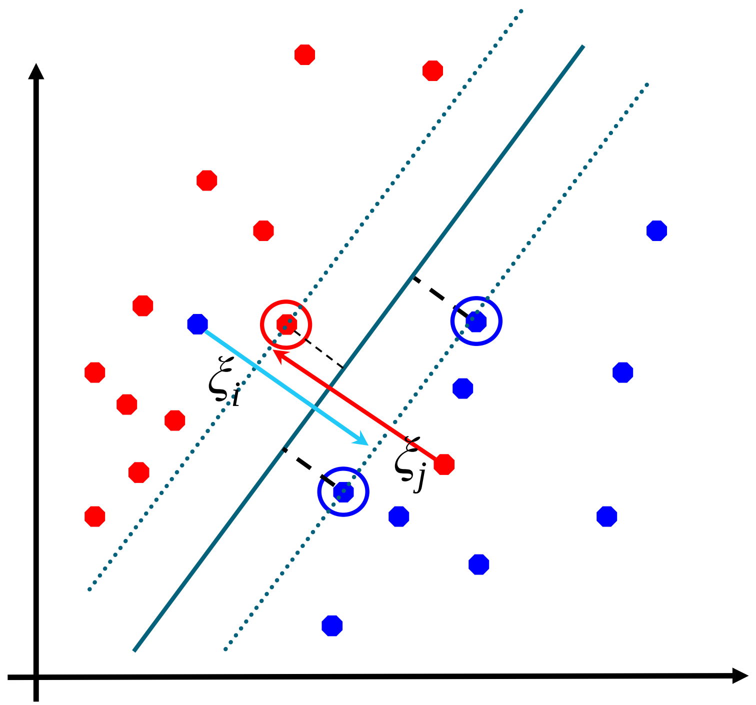
The *unconstrained optimization* problem is.

$$\text{Min } f(\vec{\mathbf{w}}) = \frac{||\vec{\mathbf{w}}||^2}{2} + C\left(\sum_{i=1}^{N} \text{Max}(0, 1 - y_i f(\vec{\mathbf{x}}_i))\right)$$

where $\text{Max}(0, 1 - y_i f(\vec{\mathbf{x}}_i))$ be the loss function. It is "hinge" loss, an approximation to 0-1 loss.

Points are in three categories:

- $y_i f(\mathbf{x}_i) > 1$:
  - Point is outside margin.
  - No contribution to loss
- $y_i f(\mathbf{x}_i) = 1$:
  - Point is on margin.
  - No contribution to loss.
  - As in hard margin case.
- $y_i f(\mathbf{x}_i) < 1$:
  - Point violates margin constraint.
  - Contributes to loss



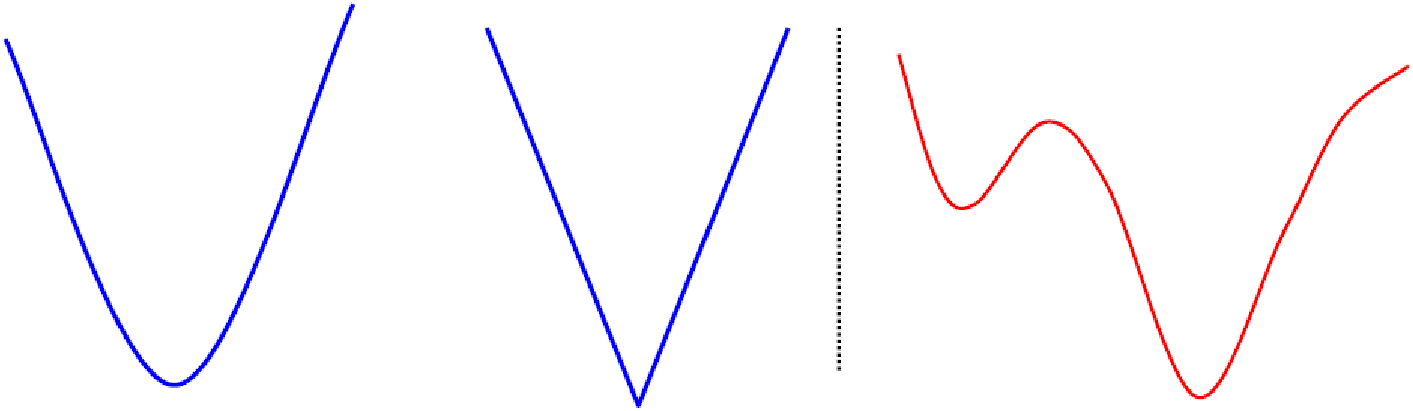
Gradient (or Steepest) Descent Algorithm for SVM

The *unconstrained optimization* problem is.

$$\text{Min } f(\vec{\mathbf{w}}) = \frac{||\vec{\mathbf{w}}||^2}{2} + C\left(\sum_{i=1}^{N} \text{Max}(0, 1 - y_i f(\vec{\mathbf{x}}_i))\right)$$

This is a *convex* function.

A non-negative sum of convex functions is convex.



Convex

Non-convex

Gradient (or Steepest) Descent Algorithm for SVM

The *unconstrained optimization* problem is.

$$\text{Min } f(\vec{\mathbf{w}}) = \frac{||\vec{\mathbf{w}}||^2}{2} + C\left(\sum_{i=1}^{N} \text{Max}(0, 1 - y_i f(\vec{\mathbf{x}}_i))\right)$$

To minimize a cost function *f*(**w**) use the iterative update as per the gradient (steepest) descent rule.

$$\vec{\mathbf{w}}_{t+1} = \vec{\mathbf{w}}_t - \eta \nabla_{\vec{\mathbf{w}}} f(\vec{\mathbf{w}}_t)$$

where η is the learning rate.
The iterative update is

$$\vec{\mathbf{w}}_{t+1} \leftarrow \vec{\mathbf{w}}_t - \eta \nabla_{\vec{\mathbf{w}}} f(\vec{\mathbf{w}}_t)$$

$$\text{where } \nabla_{\vec{\mathbf{w}}} f(\vec{\mathbf{w}}_t) = \frac{1}{\partial \vec{\mathbf{w}}}\left(\frac{||\vec{\mathbf{w}}||^2}{2} + C\sum_{i=1}^{N} \text{Max}(0, 1 - y_i f(\vec{\mathbf{x}}_i))\right)$$

Gradient of the SVM objective requires summing over the entire *N* number of training data. It slows and does not really scale. Instead we will take derivative of single data only.

Gradient (or Steepest) Descent Algorithm for SVM

Therefore,

$$\nabla_{\vec{\mathbf{w}}} f(\vec{\mathbf{w}}_t) = \frac{1}{\partial \vec{\mathbf{w}}}\left(\frac{||\vec{\mathbf{w}}||^2}{2} + C.\text{Max}(0, 1 - y_i f(\vec{\mathbf{x}}_i))\right)$$

$$= ||\vec{\mathbf{w}}|| + C.\frac{1}{\partial \vec{\mathbf{w}}}(\text{Max}(0, 1 - y_i f(\vec{\mathbf{x}}_i)))$$

$$= \begin{cases} ||\vec{\mathbf{w}}|| - C y_i x_i, & \text{If } (1 - y_i f(\vec{\mathbf{x}}_i)) > 0 \\ ||\vec{\mathbf{w}}||, & \text{Otherwise} \end{cases}$$

Therefore, $w_j^{(t+1)} \leftarrow \begin{cases} w_j^{(t)} - \eta(w_j^{(t)} - C y_i x_{ij}), & \text{If } (1 - y_i f(\vec{\mathbf{x}}_i)) > 0 \\ w_j^{(t)} - \eta(w_j^{(t)}), & \text{Otherwise} \end{cases}$

Gradient (or Steepest) Descent Algorithm for SVM

Algorithm 1 : Gradient Descent for SVM

Input:(1) Training data $S = \{\vec{\mathbf{x}}_i, y_i\}_{i=1}^{N}, y_i \in \{+1, -1\}$.

Initialize $\vec{\mathbf{w}} = 0, C, \eta$. $//\vec{\mathbf{w}} = \{w_1, w_2, ..., w_D\}$ is initialized as null.
for $t = 1$ to T
  for each training eaxmple $\{\vec{\mathbf{x}}_i, y_i\} \in S$
    for $j = 1$ to D
      if $((1 - y_i f(\vec{\mathbf{x}}_i)) > 0)$ **then**
        $w_j^{(t+1)} = w_j^{(t)} - \eta(w_j^{(t)} - C y_i x_{ij})$
      else
        $w_j^{(t+1)} = w_j^{(t)} - \eta(w_j^{(t)})$
      end if
    end for
  end for
end for
Return $\vec{\mathbf{w}}$.

Assignments

***Assignment* SVM1:** Suppose that the following are a set of points in two classes:

$$\text{Class 1: } \begin{pmatrix}1\\1\end{pmatrix} \begin{pmatrix}1\\2\end{pmatrix} \begin{pmatrix}2\\1\end{pmatrix}$$

$$\text{Class 2: } \begin{pmatrix}0\\0\end{pmatrix} \begin{pmatrix}1\\0\end{pmatrix} \begin{pmatrix}0\\1\end{pmatrix}$$

Plot the points. What are the support vectors?

***Assignment* SVM2:** Write the pseudocode for stochastic and batch gradient descent algorithm for the SVM.

Further study:

- Non-linear SVM.
- Constrained Optimization: Quadratic programming (QP).
- Pegasos algorithm: Gradient descent for SVM.

Assignments

***Assignment* SVM3:** The hyperplane function for two variables is:

$$f(x) = \omega_1 x_1 + \omega_2 x_2 + \omega_0$$

Two hyperplanes are built for two different classifiers as:

$$f_1(x) = 2x_1 + 5x_2 + 5$$

$$f_2(x) = 20x_1 + 50x_2 + 5$$

Find the margin and which one is better?

$$\text{Hints: Margin } d = \frac{2}{\sqrt{\sum_{i=1}^{D} w_i^2}} = \frac{2}{||\vec{\mathbf{w}}||}$$

Assignments

***Assignment* SVM4:** Consider the following data set for SVM (hard-margin).

a) After computation, the Lagrangian multipliers are found as:

$$\lambda = [0, 0, 0, 0.25, 0, 0.25, 0, 0, 0, 0]$$

Identify the support vectors.

b) Using part (a), find out the decision hyperplane.

Record	x_1	x_2	Class (y_i)
1	1	1	-1
2	2	1	-1
3	1	2	-1
4	2	2	-1
5	1.5	1.5	-1
6	4	4	+1
7	4	5	+1
8	5	4	+1
9	5	5	+1
10	4.5	4.5	+1

Books:

1. Chapter 6: "Neural Networks and Learning Machines" by Simon Haykin, Pearson (3rd ed.)
2. Chapter 8: "Machine Learning an Algorithmic Perspective" *by* Stephen Marsland, CRC Press (2nd ed).
3. Chapter 5: "Introduction to Data Mining" by PN Tang, M Steinbach, V Kumar, Pearson.
4. Chapter 10: "Data Mining: The Textbook" by Charu C. Aggarwal, Springer.