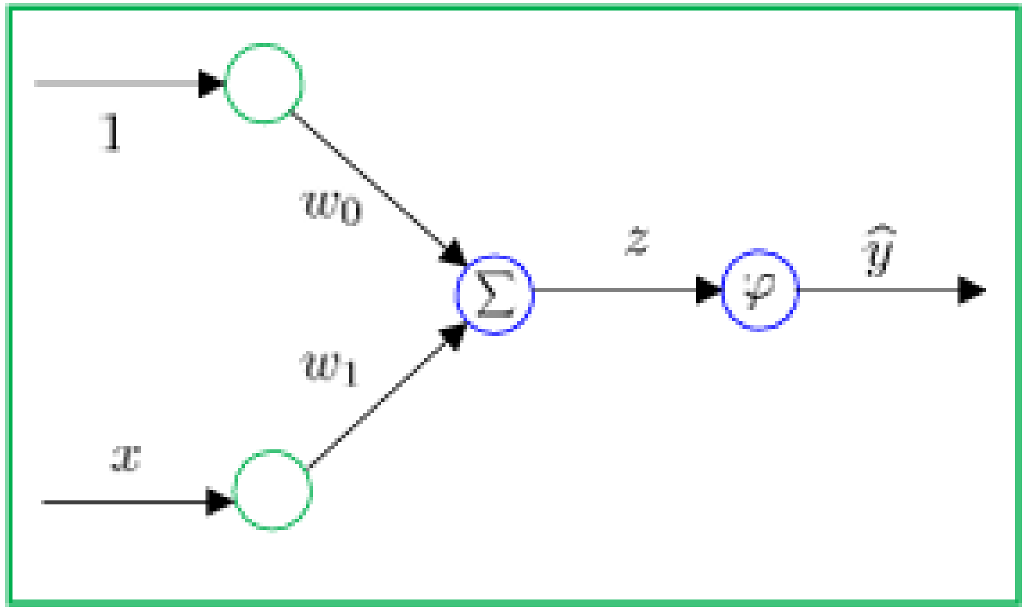
Perceptron with 1 input

Basic unit of a neural network

perceptron output $\hat{y} = \varphi(z)$ $\quad z = w_0 + w_1 x$

bias

weight



Perceptron solves **two-class classification** problems $(\mathcal{C}_1, \mathcal{C}_2)$

Activation function $\varphi(z) = \begin{cases} 1 & z \geq 0 \\ 0 & \text{otherwise} \end{cases}$

For certain input $x \in \mathcal{C}_1 \cup \mathcal{C}_2$

if $z \geq 0$ then $x \in \mathcal{C}_1$

if $z < 0$ then $x \in \mathcal{C}_2$

The equal to (=) assignment is arbitrary

dividing line (point, in this case) $w_0 + w_1 x = 0$

$x = -w_1 / w_0$

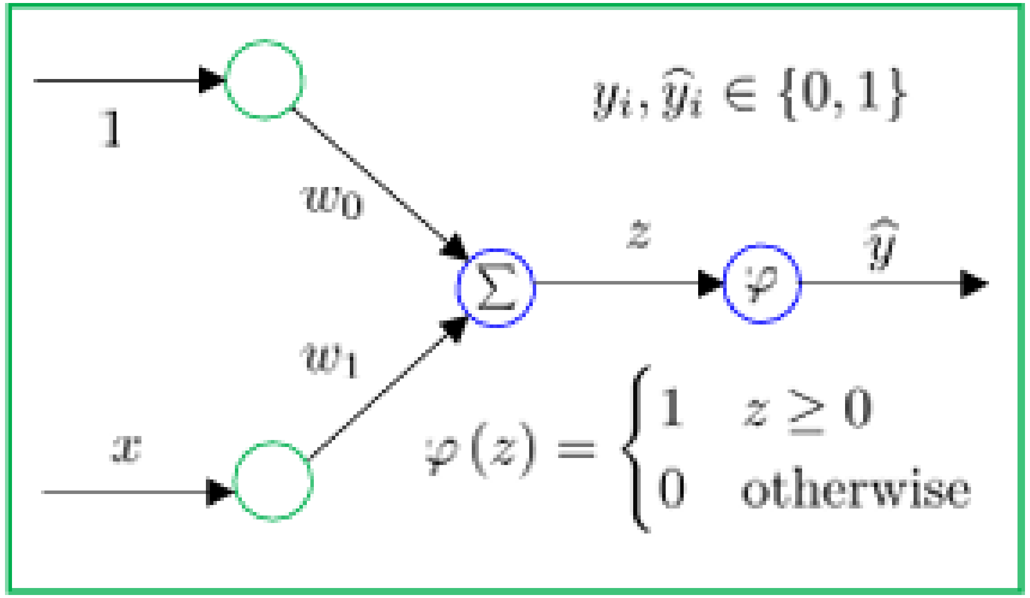
Learning of Perceptron

Computing optimum w_0, w_1

using the training data $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{n}$

An appropriate cost function may be optimized to find w_0, w_1

However, perceptron has a specific learning algorithm that works very well



learning rate $r \in (0, 1)$ (user defined)

$\mathbf{w}$ after k-th iteration $\mathbf{w}^{(k)}$

For input $\mathbf{x}_i$ $\quad \widehat{y}_i = \varphi\left(\mathbf{w}^{(k)T}\mathbf{x}_i\right) \quad i = 1, 2, \cdots, n$

Loss function $L_i = y_i - \widehat{y}_i \quad L_i \in \{0, 1, -1\}$

$\mathbf{w}^{(k)} \leftarrow \mathbf{w}^{(k)} + rL_i\mathbf{x}_i$

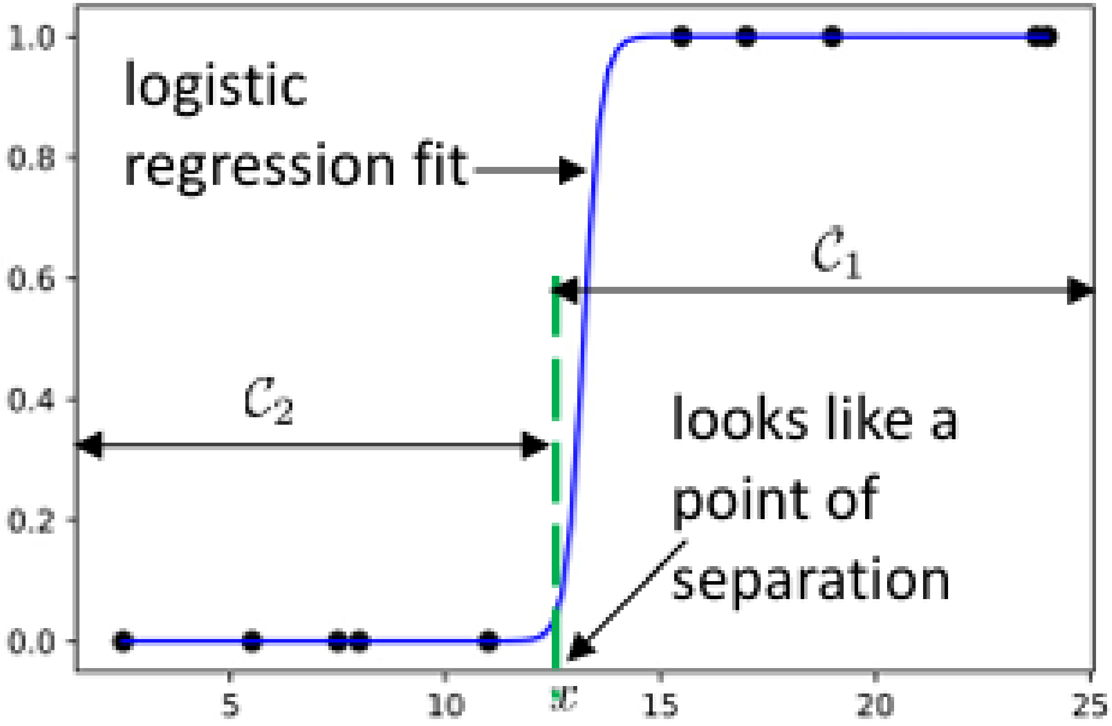
$\mathbf{w}^{(k+1)} \leftarrow \mathbf{w}^{(k)}$

The algorithm converges (to correct solution) after finite no of steps, provided the dataset is linearly separable (proof beyond the scope of this course)

Recall the problem of propellant problem

The data here is 1D, linearly separable

Data can be classified by a point in 1D, or by a straight line in 2D etc.



Test	Propellant age (Weeks) x	Shear strength test results y
1	15.5	fail = 1
2	23.75	fail = 1
3	8	pass = 0
4	17	fail = 1
5	5.5	pass = 0
6	19	fail = 1
7	24	fail = 1
8	2.5	pass = 0
9	7.5	pass = 0
10	11	pass = 0

A perceptron should be able to classify the data

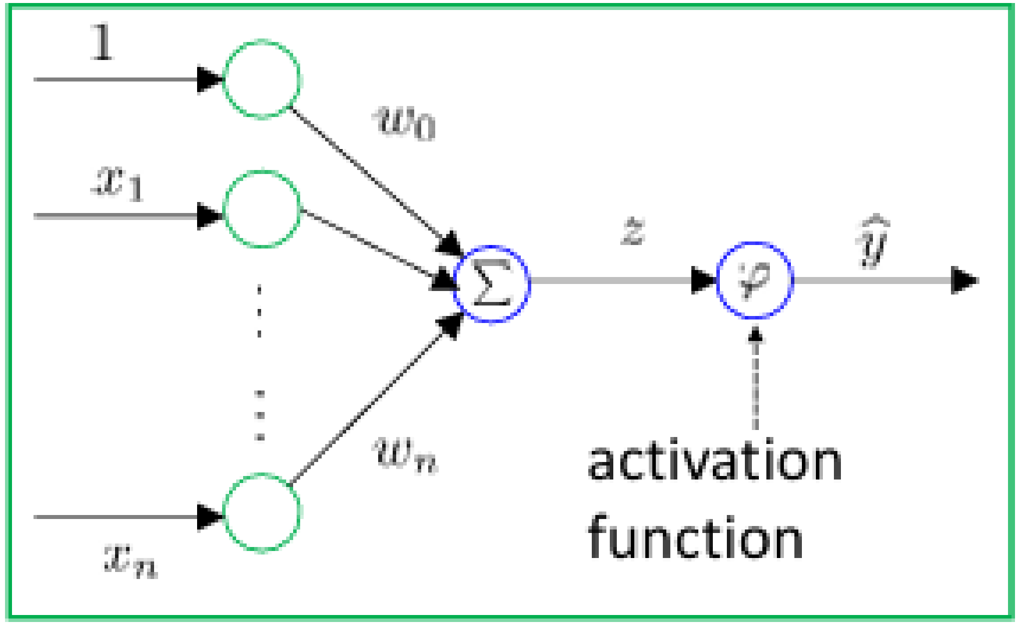
In general, a perceptron can accept vector input of any dimension

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T$$

w_0

and computes the necessary weights and bias

$$\mathbf{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix}^T$$

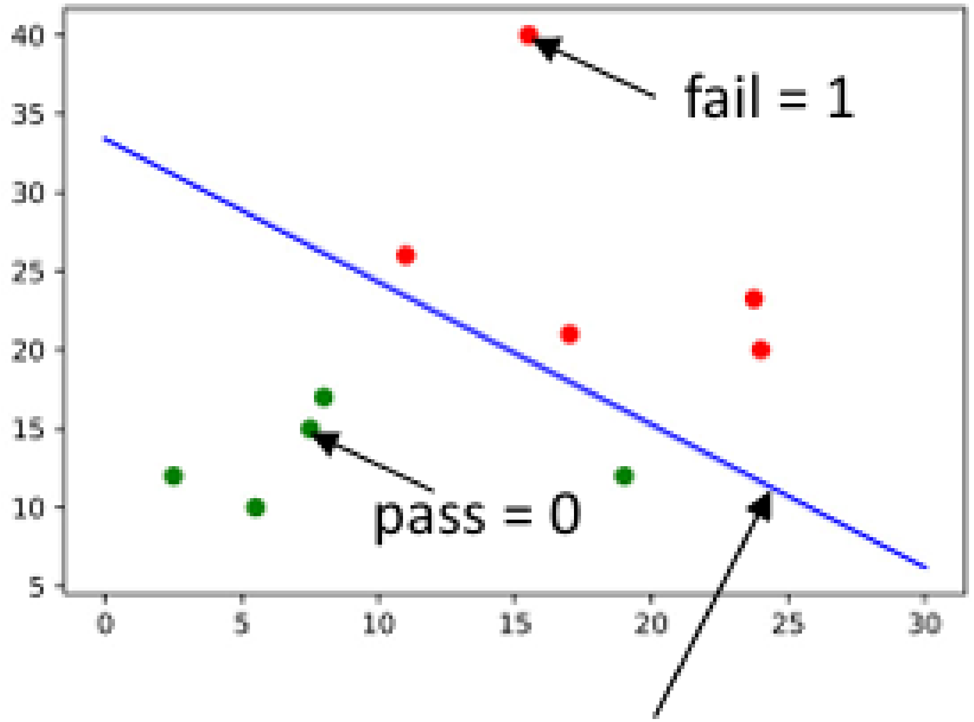


Activation function $\varphi(z) = \begin{cases} 1 & z \geq 0 \\ 0 & \text{otherwise} \end{cases}$

The perceptron learning algorithm, discussed here, is known as **online learning** where we use one training data after another

In **offline learning** we use all training data together, as we have seen in case of least square regression/classification

Propellant shear strength experiments at various ages and storage temperatures



linear decision boundary $w_0 + \mathbf{w}^T\mathbf{x} = 0$

$$\Rightarrow -3.59 + 0.0975x_1 + 0.1075x_2 = 0$$

Test	Propellant age (Weeks) x_1	Storage temperature (°C) x_2	Shear strength test y
1	15.5	40	fail = 1
2	23.75	23.25	fail = 1
3	8	17	pass = 0
4	17	21	fail = 1
5	5.5	10	pass = 0
6	19	12	pass = 0
7	24	20	fail = 1
8	2.5	12	pass = 0
9	7.5	15	pass = 0
10	11	26	fail = 1

The data are linearly separable

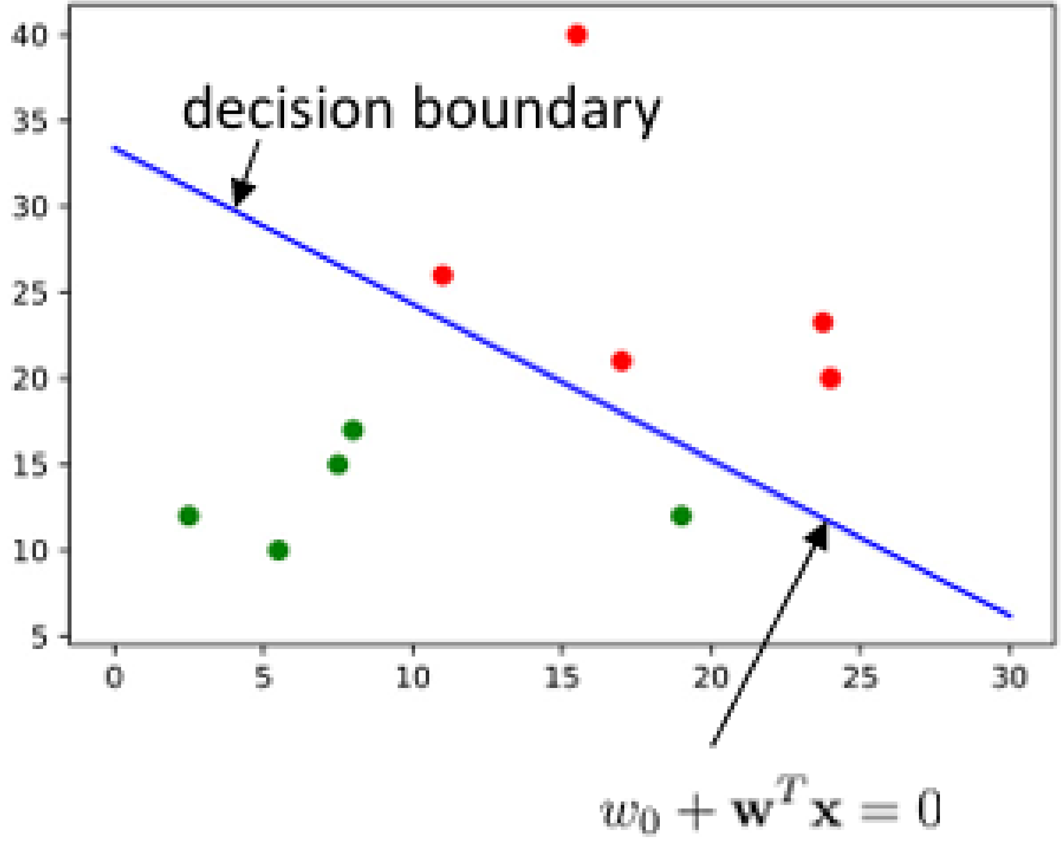
Beyond Perceptron

Perceptron usually have multiple (infinitely many) solutions

Not reliable for test points near the decision boundary

Sensitive to new data, outliers

Perceptron works with linearly separable data; not all data are linearly separable



Possible remedies: Artificial Neural Network (ANN)

Support Vector Machines (SVM) ← classifies with margin

classifies data that are not linearly separable

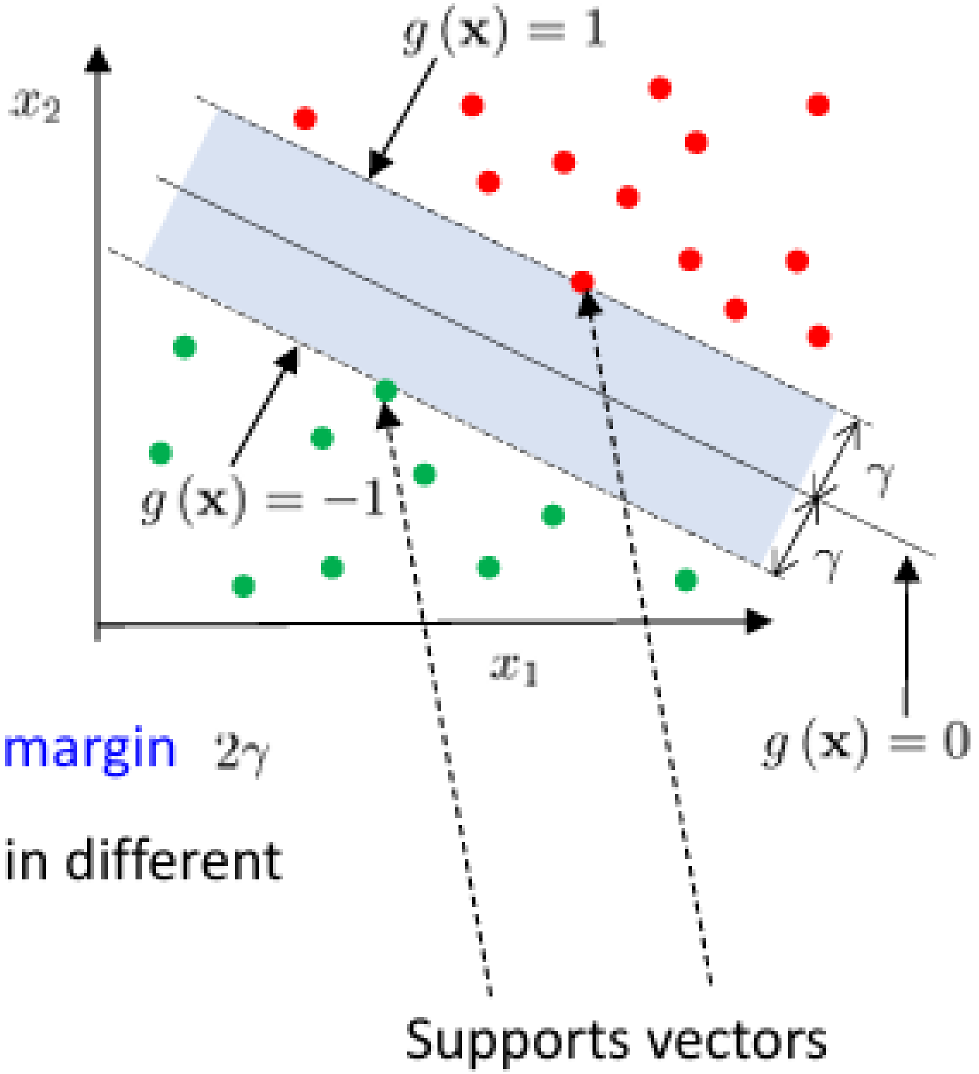
insensitive to outliers

Consider a classification problem with linear decision boundary

linear decision boundary $g(\mathbf{x}) = w_0 + \mathbf{w}^T\mathbf{x} = 0$

bias: w_0 $\mathbf{x} = [x_1 \quad x_2]^T$

weight: $\mathbf{w} = [w_1 \quad w_2]^T$



Support vector machines (SVM) maximizes the margin 2γ

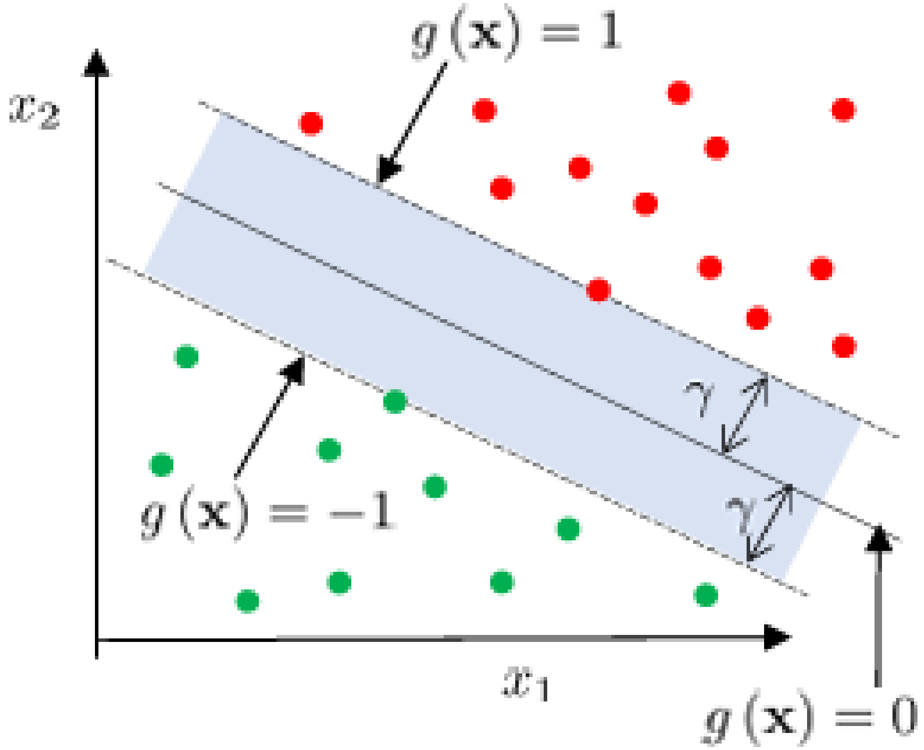
Support vectors are the closest vectors (points), in different classes, to the decision boundary

No of support vectors: two or more

Instead of a decision line, SVM creates a decision margin, that depends only on the points of different classes close to each other; insensitive to outlier

$$g(\mathbf{x}) = w_0 + \mathbf{w}^T\mathbf{x} = 0 \qquad \mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$$

$$\mathbf{w} = \begin{bmatrix} w_1 & w_2 \end{bmatrix}^T$$



Support vector machines (SVM)

maximizes the margin 2γ

$\downarrow$

distance between $g(\mathbf{x}) = 1$

and $g(\mathbf{x}) = -1$

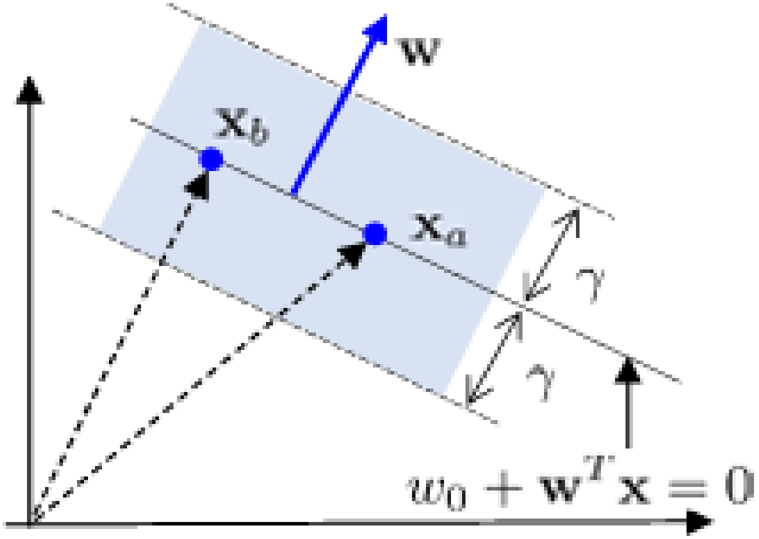
Consider two points $\mathbf{x}_a, \mathbf{x}_b$ on the decision boundary

$$w_0 + \mathbf{w}^T\mathbf{x}_a = 0 = w_0 + \mathbf{w}^T\mathbf{x}_b \quad \Rightarrow \mathbf{w}^T(\mathbf{x}_a - \mathbf{x}_b) = 0$$

$\mathbf{x}_a - \mathbf{x}_b$: decision boundary

$\mathbf{w}$: normal to decision boundary

$\dfrac{\mathbf{w}}{\|\mathbf{w}\|}$: unit vector normal to decision boundary

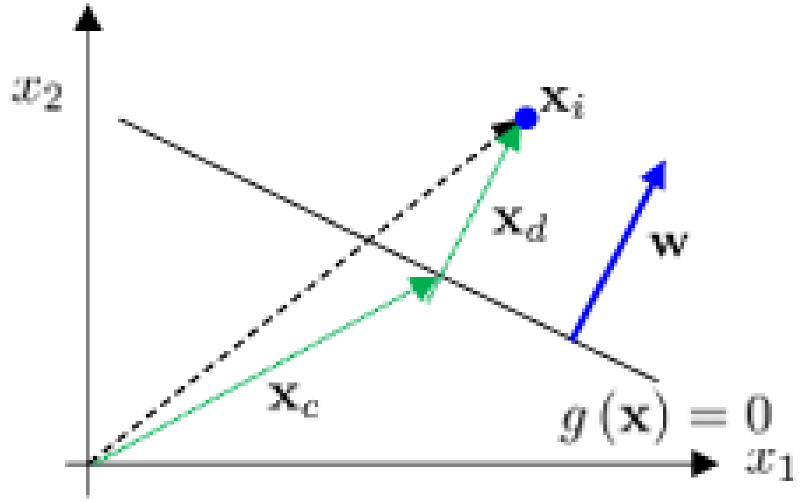


Consider an arbitrary point $\mathbf{x}_i$ $\quad \mathbf{x}_i = \mathbf{x}_c + \mathbf{x}_d$

$$\mathbf{x}_i = \mathbf{x}_c + \frac{r\mathbf{w}}{\|\mathbf{w}\|} \qquad r = \|\mathbf{x}_d\|$$

along $\mathbf{w}$

$$g(\mathbf{x}_i) = \mathbf{w}^T\left(\mathbf{x}_c + \frac{r\mathbf{w}}{\|\mathbf{w}\|}\right) + w_0$$



$$g(\mathbf{x}) = w_0 + \mathbf{w}^T\mathbf{x} = 0$$

since $\mathbf{x}_c$ is on the decision boundary $w_0 + \mathbf{w}^T\mathbf{x}_c = 0$

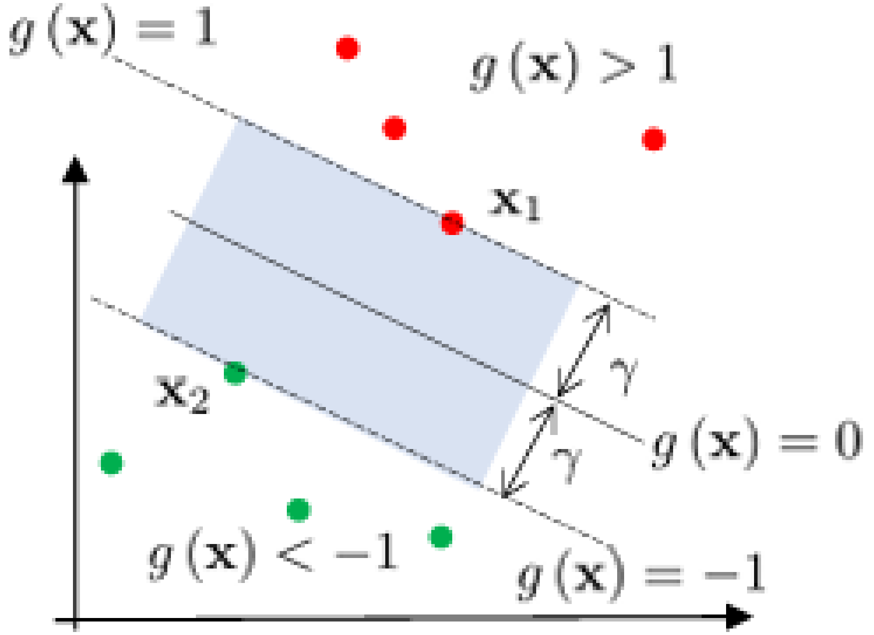
Thus $\quad g(\mathbf{x}_i) = \dfrac{\mathbf{w}^T r\mathbf{w}}{\|\mathbf{w}\|} = r\|\mathbf{w}\| \;\Rightarrow r = \dfrac{g(\mathbf{x}_i)}{\|\mathbf{w}\|}$

For point on the margin $\mathbf{x}_1, \mathbf{x}_2$ $\quad \gamma = \dfrac{1}{\|\mathbf{w}\|}$

To maximize the margin, we maximize $\quad \gamma = \dfrac{1}{\|\mathbf{w}\|}$

Defining labels $\quad y = 1, -1 \qquad yg(\mathbf{x}) \geq 1$

for all training points

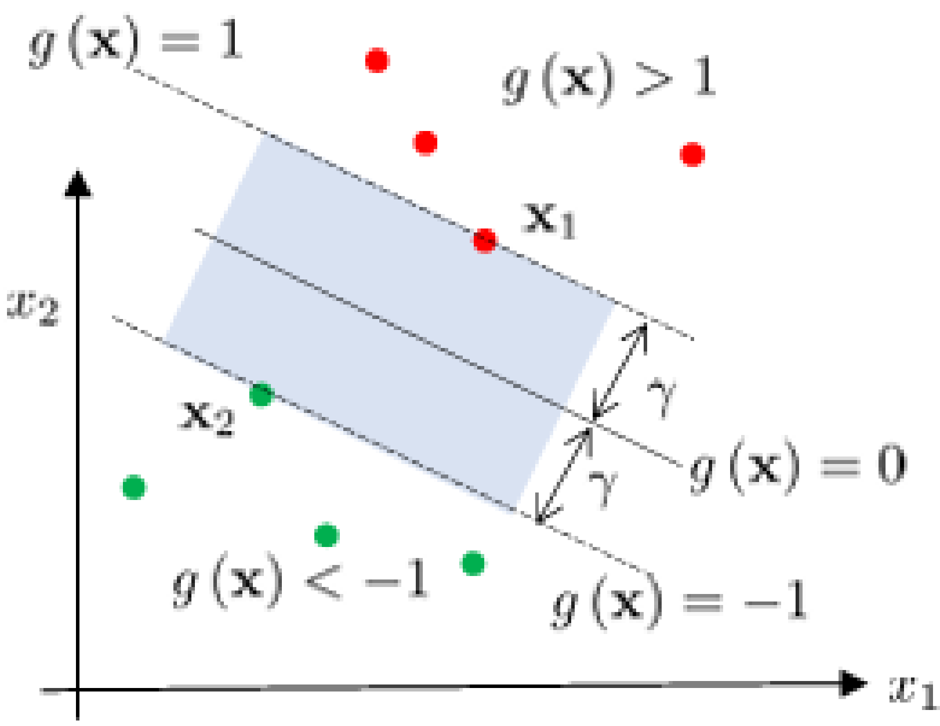


SVM maximizes margin $\gamma = \frac{1}{\|\mathbf{w}\|}$

where $yg(\mathbf{x}) \geq 1$ $\quad g(\mathbf{x}) = w_0 + \mathbf{w}^T\mathbf{x}$ $\quad y = \{-1, 1\}$

maximizing $\frac{1}{\|\mathbf{w}\|}$ is equivalent to minimizing $\|\mathbf{w}\|$

same as minimizing $\frac{1}{2}\mathbf{w}^T\mathbf{w}$



SVM learning problem for training dataset $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$

Constrained optimization

$$\begin{cases} \text{minimize} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \ y_i\left(w_0 + \mathbf{w}^T\mathbf{x}_i\right) \geq 1 \quad i = 1, 2, \cdots, n \end{cases}$$

Compared to perceptron, SVM learning has a mathematically sound background